USB Power Delivery Software Framework(PSF) Configuration Options

Version 0.91

Table of Contents

1 Introduction

USB Power Delivery Software Framework(PSF) is software based Power Delivery Stack along with UPD350 Type-C Port Controller provides USB-PD functionality.

This documents serves as a guide to configure PSF to various PD functionalities which includes Port specific configurations (PowerRole, PDOs, etc.,), PD features, Timers, UPD350 PIOs configuration for DC-DC control and System configurations.

1.1 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

1.2 Terms and Abbreviations

Term	Definition
USB-PD	USB Power Delivery
UPD350	Microchip UPD350 Power Delivery Port Controller
Port Partner	Remote port which is connected to UPD350’s local port
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

2 SW License Agreement

Software License Agreement

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.

3 Configure the stack

PSF has two 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_APIHook.h -> To port and integrate to any new SOC platform

This document elaborates on Configurations available with PSF_Config.h. which contains,

- Inclusion and Exclusion of Features
- Power Delivery IDs Configuration
- System Level Configuration
- Port Specific Configuration
- Power Fault and Timers
- UPD350 PIO configuration for DC-DC control

PSF_Config.h shall be defined as per PD application and included as part of User Application Directory.

It is recommended to include this file path as one of preprocessor include path.

3.1 Configure Include/Exclude Features

This parameter is used to configure USB-PD features that is to be included or excluded in stack. User can choose to include/exclude any of the listed features based on the functional requirements. Based on this definition, part of code will be included/excluded for compilation process. This can be used effectively to reduce PSF code size if specific features aren't required.

3.1.1 INCLUDE_PD_3_0

C

```
#define INCLUDE_PD_3_0 1
```

Description

Setting the INCLUDE_PD_3_0 as 1, enables the stack to include USB Power delivery 3.0 specification features Collision Avoidance, Extended message support via chunking along with PD 2.0 features at the compile. User can set this define to 0 to reduce code size of the stack, if none of the PD enabled ports require PD 3.0 specific features and operates only at PD 2.0 specification.

Remarks

Recommended default value is '1'.

Example

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

3.1.2 INCLUDE_PD_SOURCE

C

```
#define INCLUDE_PD_SOURCE 1
```

Description

Setting the INCLUDE_PD_SOURCE as 1, enables the stack to include the USB PD Source functionality at compile time. User can set this define to 0 to reduce code size of the stack, if none of the PD enabled ports in the system are configured for Source operation.

Remarks

Recommended default value is '1' for Source Application.

Example

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

3.1.3 INCLUDE_PD_SINK

C

```
#define INCLUDE_PD_SINK 1
```

Description

Setting the INCLUDE_PD_SINK as 1, enables the stack to include USB PD Sink functionality at the compile time. User can set this define to 0 to reduce code size of the stack, if none of the PD enabled ports are configured for Sink operation.

Remarks

Recommended default value is '1' for Sink Application.

Example

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

3.1.4 INCLUDE_VCONN_SWAP_SUPPORT

C

```
#define INCLUDE_VCONN_SWAP_SUPPORT 1
```

Description

Setting the INCLUDE_VCONN_SWAP_SUPPORT as 1, enables the stack to include the VCONN Swap functionality at the compile time. User can set this define to 0 to reduce code size of the stack, if none of the PD enabled ports requires VCONN Swap functionality.

Remarks

Recommended default value is 1.

Example

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

3.1.5 INCLUDE_POWER_FAULT_HANDLING

C

```
#define INCLUDE_POWER_FAULT_HANDLING 1
```

Description

Setting the INCLUDE_POWER_FAULT_HANDLING as 1, enables the stack to handle Power faults (Source & Sink over voltage, Source OCS, Sink under voltage) as per Power Delivery specification Rev3.0 as applicable.User can set this define to 0 to reduce code size of the stack, if stack based power fault handling is not required.

Remarks

Recommended default value is 1.

Example

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

3.1.6 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 stack used this feature to disable EN_VBUS instantly on detection of a Power Fault Condition. Setting the INCLUDE_UPD_PIO_OVERRIDE_SUPPORT as 1 enables this feature. User can set this define to 0 to reduce code size of the stack, if PIO override based power faulting is not required.

Remarks

To use this feature, EN_VBUS and FAULT_IN Pin 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 and FAULT_IN.

Example

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

3.1.7 INCLUDE_POWER_MANAGEMENT_CTRL

C

```
#define INCLUDE_POWER_MANAGEMENT_CTRL 1
```

Description

Setting the INCLUDE_POWER_MANAGEMENT_CTRL as 1, enables the stack to include the functionality that puts the UPD350 into low power mode if UPD350 is inactive for CONFIG_PORT_UPD_IDLE_TIMEOUT_MS (⧉) time and stack notifies the same via the stack call back MCHP_PSF_NOTIFY_CALL_BACK. User can set this define to 0 to reduce code size of the stack, if low power mode operation of UPD350 is not required for the application.

Remarks

Recommended 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)
```

3.1.8 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. User can set this define to 0 to reduce code size of the stack, application doesnot use Firmware update feature.

Remarks

Recommended default value is 0 unless Firmware update feature is used.

Example

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

3.2 Power Delivery IDs Configuration

This section explains the configurable Power delivery IDs and versions mostly used in PDFU descriptors.

3.2.1 CONFIG_VENDOR_ID

C

```
#define CONFIG_VENDOR_ID
```

Description

CONFIG_VENDOR_ID field defines Vendor Identifier value. 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 defined as '1'.It should always be two byte width. It should always be two byte width.

Example

```
    #define CONFIG_VENDOR_ID    0x0424u
```

3.2.2 CONFIG_PRODUCT_ID

C

```
#define CONFIG_PRODUCT_ID
```

Description

CONFIG_PRODUCT_ID is the Product Identifier value. 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 defined as '1'.It should always be two byte width.

Example

```
#define CONFIG_PRODUCT_ID 0x301Cu
```

3.2.3 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
```

3.2.4 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
```

3.2.5 CONFIG_SILICON_VERSION

C

```
#define CONFIG_SILICON_VERSION
```

Description

CONFIG_SILICON_VERSION is UPD301 Silicon Base Version. 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 defined as '1'. It should be a byte width.

Example

```
#define CONFIG_SILICON_VERSION    0x01u
```

3.3 System Level configuration

This section explain PSF configurability available at overall system level.

3.3.1 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 stack can be configured is '4'.

Remarks

The max and min value for CONFIG_PD_PORT_COUNT is '4' and '1' respectively. Stack 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.

Example

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

3.3.2 CONFIG_DEFINE_UPD350_HW_INTF_SEL

C

```
#define CONFIG_DEFINE_UPD350_HW_INTF_SEL
```

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 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. Different part for each port, for example SPI supported UPD350 for Port 1 and I2C supported UPD350 part for Port 2 is not supported.

Example

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

3.4 Port Specific Configuration

This section cover PD and Type C configuration specific to each port.

3.4.1 Port basic configuration

This section covers basic port configurations like Power role, Type C current configuration option available in PSF stack.

3.4.1.1 CONFIG_PORT_n_ENDIS

C

```
#define CONFIG_PORT_n_ENDIS 1
```

Description

CONFIG_PORT_n_ENDIS defines whether a port to be enabled or disabled. n can take values between 0 and (CONFIG_PD_PORT_COUNT (⎚)-1). If CONFIG_PORT_n_ENDIS defined as 1, port is enabled. If CONFIG_PORT_n_ENDIS defined as 0, port is disabled.

Remarks

By default, all ports are enabled.

Example

```
#define CONFIG_PORT_0_ENDIS  0 (Type-C Port 0 Disabled)
#define CONFIG_PORT_0_ENDIS  1 (Type-C Port 0 Enabled)
```

3.4.1.2 CONFIG_PORT_n_POWER_ROLE

C

```
#define CONFIG_PORT_n_POWER_ROLE
```

Description

CONFIG_PORT_n_POWER_ROLE defines the Power role of nth port. n can take values between 0 and (CONFIG_PD_PORT_COUNT (⎚)-1). Setting CONFIG_PORT_n_POWER_ROLE as 1, configures the nth port as Source or Setting CONFIG_PORT_n_POWER_ROLE as 0, configures the nth port as Sink.

Remarks

The default Data Role for a port is determined based on the Power role configured via this through this define. Data role is configured as 'DFP' for Source and 'UFP' for Sink respectively. By default, all the ports are configured as Source i.e. CONFIG_PORT_0_POWER_ROLE defined as 1.

Example

```
#define CONFIG_PORT_0_POWER_ROLE    1 (Configuring the Port 0 as Source)
#define CONFIG_PORT_0_POWER_ROLE    0 (Configuring the Port 0 as Sink)
```

3.4.1.3 CONFIG_PORT_n_RP_CURRENT_VALUE

C

```
#define CONFIG_PORT_n_RP_CURRENT_VALUE
```

Description

CONFIG_PORT_n_RP_CURRENT_VALUE defines the Rp Value of nth port. n can take values between 0 and (CONFIG_PD_PORT_COUNT ()-1). CONFIG_PORT_n_RP_CURRENT_VALUE can take following values 0 - Rp termination is disabled; For Sink, CONFIG_PORT_n_RP_CURRENT_VALUE should be set '0' 1 - Rp termination is set to default USB Power 2 - Rp termination is set to 1.5A current 3 - Rp termination is set to 3A current

Remarks

If CONFIG_PORT_n_POWER_ROLE () set as 0 (Sink), CONFIG_PORT_n_RP_CURRENT_VALUE should be defined as 0. If CONFIG_PORT_n_POWER_ROLE () set as 1 (Source), CONFIG_PORT_n_RP_CURRENT_VALUE should take value other than 0. By default, all the ports are configured as Source & Rp termination set to 3A.

Example

```
#define CONFIG_PORT_0_RP_CURRENT_VALUE  0 (Configuring the Port 0 Rp Value as Disabled)
#define CONFIG_PORT_0_RP_CURRENT_VALUE  1 (Configuring the Port 0 Rp Value as DEFAULT)
#define CONFIG_PORT_0_RP_CURRENT_VALUE  2 (Configuring the Port 0 Rp Value as CURRENT_15)
#define CONFIG_PORT_0_RP_CURRENT_VALUE  3 (Configuring the Port 0 Rp Value as CURRENT_30)
```

3.4.2 Port Source configuration

This section covers all the configuration option available to configure the Source PDOs.

3.4.2.1 CONFIG_PORT_n_SOURCE_NUM_OF_PDOS

C

```
#define CONFIG_PORT_n_SOURCE_NUM_OF_PDOS
```

Description

CONFIG_PORT_n_SOURCE_NUM_OF_PDOS refers to the number PDOs supported by the nth source port. n can take values between 0 and (CONFIG_PD_PORT_COUNT () - 1).

Remarks

CONFIG_PORT_n_SOURCE_NUM_OF_PDOS can take only values from 1 to 7. Default value for CONFIG_PORT_n_SOURCE_NUM_OF_PDOS is 1.

Example

```
#define CONFIG_PORT_0_SOURCE_NUM_OF_PDOS       4 (Port 0 has 4 Source PDOs)
```

3.4.2.2 CONFIG_PORT_n_SOURCE_UNCONSTRAINED_PWR

C

```
#define CONFIG_PORT_n_SOURCE_UNCONSTRAINED_PWR
```

Description

CONFIG_PORT_n_SOURCE_UNCONSTRAINED_PWR defines the Unconstrained Power bit in fixed PDO of nth source

port. As per PD specification, this field is exposed for PDO1 alone for rest of the fixed PDOs it is Zero. CONFIG_PORT_n_SOURCE_UNCONSTRAINED_PWR can be configured as 0 or 1. n can take values between 0 and (CONFIG_PD_PORT_COUNT (🔗) - 1).

Remarks

By default, this define is set to 1.

Example

```
 #define CONFIG_PORT_0_SOURCE_UNCONSTRAINED_PWR  1 (Port 0 is unconstrained power capable)
 #define CONFIG_PORT_0_SOURCE_UNCONSTRAINED_PWR  0 (Port 0 is not unconstrained power
capable)
```

3.4.2.3 CONFIG_PORT_n_SOURCE_USB_COM

C

```
#define CONFIG_PORT_n_SOURCE_USB_COM
```

Description

CONFIG_PORT_n_SOURCE_USB_COM defines the USB communication enable bit in PDO of nth source port. As per PD specification, this field is exposed for PDO1 alone for rest of the fixed PDOs it is Zero. CONFIG_PORT_n_SOURCE_USB_COM can be configured as 0 or 1. n can take values between 0 and (CONFIG_PD_PORT_COUNT (🔗) - 1).

Remarks

By default, this define is set to 0.

Example

```
 #define CONFIG_PORT_0_SOURCE_USB_COM       1 (Port 0 is USB communication capable)
 #define CONFIG_PORT_0_SOURCE_USB_COM       0 (Port 0 is not USB communication capable)
```

3.4.2.4 CONFIG_PORT_n_SOURCE_USB_SUSPEND

C

```
#define CONFIG_PORT_n_SOURCE_USB_SUSPEND
```

Description

CONFIG_PORT_n_SOURCE_USB_SUSPEND defines the USB Suspend supported bit in fixed PDO of nth source port. As per PD specification, this field is exposed for PDO1 alone for rest of the fixed PDOs it is Zero. CONFIG_PORT_n_SOURCE_PDO_1_USB_SUSPEND can be configured as 0 or 1. n can take values between 0 and (CONFIG_PD_PORT_COUNT (🔗) - 1).

Remarks

By default, it is defined as '0'.

Example

```
 #define CONFIG_PORT_0_SOURCE_USB_SUSPEND       0 (Port 0 is not USB suspend capable)
 #define CONFIG_PORT_0_SOURCE_USB_SUSPEND       0 (Port 0 is USB suspend capable)
```

3.4.2.5 CONFIG_PORT_n_SOURCE_PDO_x_CURRENT

C

```
#define CONFIG_PORT_n_SOURCE_PDO_x_CURRENT
```

Description

CONFIG_PORT_n_SOURCE_PDO_x_CURRENT defines the maximum current value in xth PDO of nth source port. As per PD specification there can be 7 PDOs, thus x takes value from 1 to 7. n can take values between 0

and(CONFIG_PD_PORT_COUNT (🔗) - 1). This define is expressed in mA units.

Remarks

By default, PDO 1 of all the port is defined as 3000 mA and rest as 0 mA.

Example

```
#define CONFIG_PORT_0_SOURCE_PDO_1_CURRENT      3000 (Maximum current value is configured
                                                    as 3A for PDO1 of Port-0)
```

3.4.2.6 CONFIG_PORT_n_SOURCE_PDO_x_VOLTAGE

C

```
#define CONFIG_PORT_n_SOURCE_PDO_x_VOLTAGE
```

Description

CONFIG_PORT_n_SOURCE_PDO_1_VOLTAGE defines the voltage supported in xth PDO of nth source port. As per PD specification there can be 7 PDOs, So x takes value from 1 to 7. n can take values between 0 and (CONFIG_PD_PORT_COUNT (🔗) - 1).

Remarks

Units are expressed in milliVolts(mV). It is mandatory to define PDO1 as vSafe5V (5000). By default, PDO 1 of all the port is defined as 5000mV and rest of PDOs voltage as 0 mV.

Example

```
#define CONFIG_PORT_0_SOURCE_PDO_1_VOLTAGE     5000 (PDO1 voltage of Port 1 is 5V)
```

3.4.3 Port Sink configuration

This section covers all the configuration option available to configure the Sink PDOs.

3.4.3.1 CONFIG_PORT_n_SINK_NUM_OF_PDOS

C

```
#define CONFIG_PORT_n_SINK_NUM_OF_PDOS
```

Description

CONFIG_PORT_n_SINK_NUM_OF_PDOS defines the number PDOs supported by the nth sink port. n can take values between 0 and (CONFIG_PD_PORT_COUNT (🔗) - 1).

Remarks

CONFIG_PORT_n_SINK_NUM_OF_PDOS can be configured from 1 to 7. Default value for CONFIG_PORT_n_SINK_NUM_OF_PDOS is 1.

Example

```
#define CONFIG_PORT_0_SINK_NUM_OF_PDOS     4 (Port 0 has 4 Sink PDOs)
```

3.4.3.2 CONFIG_PORT_n_SINK_HIGHER_CAPABILITY

C

```
#define CONFIG_PORT_n_SINK_HIGHER_CAPABILITY
```

Description

CONFIG_PORT_n_SINK_HIGHER_CAPABILITY defines the Higher Capability bit in fixed PDO in nth sink port. As per PD

specification, this field is exposed for PDO1 alone for rest of the fixed PDOs it is Zero. CONFIG_PORT_0_SINK_HIGHER_CAPABILITY can be configured as 0 or 1. n can take values between 0 and (CONFIG_PD_PORT_COUNT (🔗) - 1).

Remarks

If the define is set as '1', Sink is higher capability capable & if it is set as '0', Sink is not higher capability capable. The default value is '1'.

Example

```
#define CONFIG_PORT_0_SINK_HIGHER_CAPABILITY       1
```

3.4.3.3 CONFIG_PORT_n_SINK_USB_SUSPEND

C

```
#define CONFIG_PORT_n_SINK_USB_SUSPEND
```

Description

CONFIG_PORT_n_SINK_USB_SUSPEND defines the USB Suspend supported bit in fixed PDO of nth sink port. As per PD specification, this field is exposed for PDO1 alone for rest of the fixed PDOs it is Zero. CONFIG_PORT_n_SINK_PDO_1_USB_SUSPEND can be configured as 0 or 1. n can take values between 0 and (CONFIG_PD_PORT_COUNT (🔗) - 1).

Remarks

By default, it is defined as '0'.

Example

```
#define CONFIG_PORT_0_SINK_USB_SUSPEND     0 (Port 0 is not USB suspend capable)
#define CONFIG_PORT_0_SINK_USB_SUSPEND     0 (Port 0 is USB suspend capable)
```

3.4.3.4 CONFIG_PORT_n_SINK_UNCONSTRAINED_PWR

C

```
#define CONFIG_PORT_n_SINK_UNCONSTRAINED_PWR
```

Description

CONFIG_PORT_n_SINK_UNCONSTRAINED_PWR defines the Unconstrained Power bit in fixed PDO of nth sink port. As per PD specification, this field is exposed for PDO1 alone for rest of the fixed PDOs it is Zero. CONFIG_PORT_n_SINK_UNCONSTRAINED_PWR can be configured as 0 or 1. n can take values between 0 and (CONFIG_PD_PORT_COUNT (🔗) - 1).

Remarks

By default, this define is set to 0.

Example

```
#define CONFIG_PORT_0_SINK_UNCONSTRAINED_PWR    1 (Port 0 is unconstrained power capable)
#define CONFIG_PORT_0_SINK_UNCONSTRAINED_PWR    0 (Port 0 is not unconstrained power
capable)
```

3.4.3.5 CONFIG_PORT_n_SINK_USB_COM

C

```
#define CONFIG_PORT_n_SINK_USB_COM 0
```

Description

CONFIG_PORT_n_SINK_USB_COM defines the USB communication enable bit in PDO of nth sink port. As per PD specification, this field is exposed for PDO1 alone for rest of the fixed PDOs it is Zero. CONFIG_PORT_n_SINK_USB_COM

can be configured as 0 or 1. n can take values between 0 and (CONFIG_PD_PORT_COUNT (🔗) - 1).

Remarks

By default, this define is set to 0.

Example

```
#define CONFIG_PORT_0_SINK_USB_COM     1 (Port 0 is USB communication capable)
#define CONFIG_PORT_0_SINK_USB_COM     0 (Port 0 is not USB communication capable)
```

3.4.3.6 CONFIG_PORT_n_SINK_PDO_x_CURRENT

C

```
#define CONFIG_PORT_n_SINK_PDO_x_CURRENT
```

Description

CONFIG_PORT_n_SINK_PDO_x_CURRENT defines the maximum current value in xth PDO of nth Sink port. As per PD specification there can be 7 PDOs, thus x takes value from 1 to 7. n can take values between 0 and(CONFIG_PD_PORT_COUNT (🔗) - 1). This define is expressed in mA units.

Remarks

By default, PDO 1 of all the port is defined as 3000 mA and rest as 0 mA.

Example

```
#define CONFIG_PORT_0_SINK_PDO_1_CURRENT        3000 (Maximum current value is configured
                                                      as 3A for PDO1 of sink Port-0)
```

3.4.3.7 CONFIG_PORT_n_SINK_PDO_x_VOLTAGE

C

```
#define CONFIG_PORT_n_SINK_PDO_x_VOLTAGE
```

Description

CONFIG_PORT_n_SINK_PDO_1_VOLTAGE defines the voltage supported in xth PDO of nth sink port. As per PD specification there can be 7 PDOs, So x takes value from 1 to 7. n can take values between 0 and (CONFIG_PD_PORT_COUNT (🔗) - 1).

Remarks

Units are expressed in milliVolts(mV). It is mandatory to define PDO1 as vSafe5V (5000). By default, PDO 1 of all the port is defined as 5000mV and rest of PDOs voltage as 0 mV.

Example

```
#define CONFIG_PORT_0_SINK_PDO_1_VOLTAGE       5000 (PDO1 voltage of sink Port 1 is 5V)
```

3.5 Power Fault configuration

This section explains configurations required for Power fault handling by stack.

3.5.1 CONFIG_OVER_VOLTAGE_FACTOR

C

```
#define CONFIG_OVER_VOLTAGE_FACTOR 1.15
```

Description

CONFIG_OVER_VOLTAGE_FACTOR is percentage of PDO voltage to be considered as Over Voltage for that PDO. As per PD specification desired range for fixed PDO voltage is (0.95 * PDO Voltage) to (1.05 * PDO Voltage), So CONFIG_OVER_VOLTAGE_FACTOR should be greater than the desired range.

Remarks

If 115% of the PDO voltage has to be considered as overvoltage for that PDO voltage, then define CONFIG_OVER_VOLTAGE_FACTOR as 1.15. It is mandatory to define CONFIG_OVER_VOLTAGE_FACTOR when INCLUDE_POWER_FAULT_HANDLING () is defined as '1'. Default value for this macro is 1.15 indicating 115%.

Example

```
#define CONFIG_OVER_VOLTAGE_FACTOR          1.15
(CONFIG_PORT_0_SOURCE_PDO_1_VOLTAGE is 5000, then for PDO 1 Over voltage is 5750mV)
```

3.5.2 CONFIG_UNDER_VOLTAGE_FACTOR

C

```
#define CONFIG_UNDER_VOLTAGE_FACTOR 0.85
```

Description

CONFIG_UNDER_VOLTAGE_FACTOR is percentage of PDO voltage to be considered as under Voltage for that PDO. As per PD specification desired range for fixed PDO voltage is (0.95 * PDO Voltage) to (1.05 * PDO Volatge), So CONFIG_OVER_VOLTAGE_FACTOR () should be less than the desired range.

Remarks

If 85% of the PDO voltage has to be considered as under voltage for that PDO voltage, then define CONFIG_UNDER_VOLTAGE_FACTOR as 0.85. CONFIG_UNDER_VOLTAGE_FACTOR must be defined when INCLUDE_POWER_FAULT_HANDLING () is defined. As an exceptional case this factor is not considered for VSafe5V.

For Source VSafe5V, CONFIG_VSINKDISCONNECT_VOLTAGE () is considered as Vsafe5V undervoltage instead of (CONFIG_UNDER_VOLTAGE_FACTOR * TYPEC_VBUS_5V). For Sink, VSafe5V under voltage is not applicable as when voltage is less than or equal to CONFIG_VSINKDISCONNECT_VOLTAGE (), sink becomes disconnected. The default value for CONFIG_UNDER_VOLTAGE_FACTOR is 0.85 indicating 85%.

Example

```
#define CONFIG_UNDER_VOLTAGE_FACTOR         0.85
(CONFIG_PORT_0_SOURCE_PDO_2_VOLTAGE is 9000, then for PDO 2 Over voltage is 7650mV)
```

3.5.3 CONFIG_FAULT_IN_OCS_DEBOUNCE_MS

C

```
#define CONFIG_FAULT_IN_OCS_DEBOUNCE_MS 5
```

Description

CONFIG_FAULT_IN_OCS_DEBOUNCE_MS is debounce timer value in terms of milliseconds for VBUS overcurrent fault conditions before reacting and entering fault recovery routine. It is applicable only for OCS detection via FAULT_IN configured UPD350 pin.

Remarks

The default debounce for Fault IN OCS detection is 5ms.

Example

```
#define CONFIG_FAULT_IN_OCS_DEBOUNCE_MS           5 (Debounce is 5ms)
```

3.5.4 CONFIG_POWER_GOOD_TIMER_MS

C

```
#define CONFIG_POWER_GOOD_TIMER_MS MILLISECONDS_TO_TICKS(10000)
```

Description

After an automatic fault recovery, a CONFIG_POWER_GOOD_TIMER_MS is ran 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.

For power Source, it is the time a power source must consistently provide power without a power 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.

Remarks

It shall be expressed in MILLISECONDS_TO_TICKS defines. By default, it is configured to 10Seconds.

Example

```
#define CONFIG_POWER_GOOD_TIMER_MS          MILLISECONDS_TO_TICKS(10000)
```

3.5.5 CONFIG_VCONN_OCS_ENABLE

C

```
#define CONFIG_VCONN_OCS_ENABLE 1
```

Description

PSF uses UPD350 internal comparator to detect VCONN Overcurrent fault. CONFIG_VCONN_OCS_ENABLE is to enable or disable the internal VCONN OCS detection logic. Setting CONFIG_VCONN_OCS_ENABLE as '1' enables the VCONN OCS detection and setting as '0' disables the VCONN OCS detection.

Remarks

The default value for CONFIG_VCONN_OCS_ENABLE is '1'.

Example

```
#define CONFIG_VCONN_OCS_ENABLE    1(Enables VCONN OCS detection)
#define CONFIG_VCONN_OCS_ENABLE    0(Disbales VCONN OCS detection)
```

3.5.6 CONFIG_VCONN_OCS_DEBOUNCE_IN_MS

C

```
#define CONFIG_VCONN_OCS_DEBOUNCE_IN_MS 2
```

Description

CONFIG_VCONN_OCS_DEBOUNCE_IN_MS is debounce timer value in terms of milliseconds for VCONN overcurrent fault conditions before reacting and entering fault recovery routine.

Remarks

The default value for CONFIG_VCONN_OCS_DEBOUNCE_IN_MS is 2ms.

Example

```
#define CONFIG_VCONN_OCS_DEBOUNCE_IN_MS            2 (Debounce is 2ms)
```

3.5.7 CONFIG_MAX_VCONN_FAULT_COUNT

C

```
#define CONFIG_MAX_VCONN_FAULT_COUNT 3
```

Description

CONFIG_MAX_VCONN_POWER_FAULT_COUNT is the maximum number of back-to-back VCONN faults allowed before it disables the VCONN. A back-to-back fault is a second fault which occurs within the CONFIG_POWER_GOOD_TIMER_MS () after a port is automatically re-enabled from a previous fault condition. If VCONN disabled due to occurrent VCONN power fault, VCONN will be enabled only after a physical detach and re-attach.

Remarks

By default, it is configured to count 3.

Example

```
#define CONFIG_MAX_VCONN_FAULT_COUNT             3
```

3.5.8 CONFIG_MAX_VBUS_POWER_FAULT_COUNT

C

```
#define CONFIG_MAX_VBUS_POWER_FAULT_COUNT 3
```

Description

CONFIG_MAX_VBUS_POWER_FAULT_COUNT is the maximum number of back-to-back VBUS faults allowed before shut down of the port. A back-to-back fault is a second fault which occurs within the CONFIG_POWER_GOOD_TIMER_MS () after a port is automatically re-enabled from a previous fault condition. During port shutdown due to occurrent fault, the device removes its CC termination and wait for port partner to get detached physically from the port to resume its normal operation.

Remarks

By default, it is configured to count 3.

Example

```
#define CONFIG_MAX_VBUS_POWER_FAULT_COUNT           3
```

3.6 Vsafe5V Configuration for Source and Sink

This section explains the configurability available to define VSafe5v threshold range.

3.6.1 CONFIG_SRC_VSAFE5V_DESIRED_MAX_VOLTAGE

C

```
#define CONFIG_SRC_VSAFE5V_DESIRED_MAX_VOLTAGE 5500
```

Description

CONFIG_SRC_VSAFE5V_DESIRED_MAX_VOLTAGE is maximum voltage acceptable for VSafe5V expressed in terms of millivolts for source. The voltage will be considered as valid Vsafe5V only if it is equal to or greater than CONFIG_SRC_VSAFE5V_DESIRED_MIN_VOLTAGE () & less than CONFIG_SRC_VSAFE5V_DESIRED_MAX_VOLTAGE. CONFIG_OVER_VOLTAGE_FACTOR () * 5000mV will be considered as overvoltage for Vsafe5V for Source.

Valid Vsafe5V condition: CONFIG_SRC_VSAFE5V_DESIRED_MIN_VOLTAGE () <= Valid Vsafe5V < CONFIG_SRC_VSAFE5V_DESIRED_MAX_VOLTAGE Vsafe5V overvoltage condition: VBUS >= CONFIG_OVER_VOLTAGE_FACTOR () * 5000mV

Remarks

It is mandatory to define CONFIG_SRC_VSAFE5V_DESIRED_MAX_VOLTAGE. It must be defined in such a way that following condition is met. CONFIG_SRC_VSAFE5V_DESIRED_MAX_VOLTAGE < CONFIG_OVER_VOLTAGE_FACTOR () * TYPEC_VBUS_5V. By default, it is defined as 5500 mV.

Example

```
#define CONFIG_SRC_VSAFE5V_DESIRED_MAX_VOLTAGE          5500
```

3.6.2 CONFIG_SRC_VSAFE5V_DESIRED_MIN_VOLTAGE

C

```
#define CONFIG_SRC_VSAFE5V_DESIRED_MIN_VOLTAGE 4750
```

Description

CONFIG_SRC_VSAFE5V_DESIRED_MIN_VOLTAGE is minimum voltage acceptable for VSafe5V expressed in terms of millivolts for source. The voltage will be considered as valid Vsafe5V only if it is equal to or greater than CONFIG_SRC_VSAFE5V_DESIRED_MIN_VOLTAGE & less than CONFIG_SRC_VSAFE5V_DESIRED_MAX_VOLTAGE ().

Valid Vsafe5V condition: CONFIG_SRC_VSAFE5V_DESIRED_MIN_VOLTAGE <= Valid Vsafe5V < CONFIG_SRC_VSAFE5V_DESIRED_MAX_VOLTAGE ()

Remarks

It is mandatory to define CONFIG_SRC_VSAFE5V_DESIRED_MIN_VOLTAGE. It must be defined in such a way that following condition is met: CONFIG_SRC_VSAFE5V_DESIRED_MIN_VOLTAGE > CONFIG_VSINKDISCONNECT_VOLTAGE (). The default value for this macro is 4750mV.

Example

```
#define CONFIG_SRC_VSAFE5V_DESIRED_MIN_VOLTAGE          4750
```

3.6.3 CONFIG_SNK_VSAFE5V_DESIRED_MAX_VOLTAGE

C

```
#define CONFIG_SNK_VSAFE5V_DESIRED_MAX_VOLTAGE 5500
```

Description

CONFIG_SNK_VSAFE5V_DESIRED_MAX_VOLTAGE is maximum voltage acceptable for VSafe5V expressed in terms of millivolts for sink. The voltage will be considered as valid Vsafe5V only if it is equal to or greater than CONFIG_SNK_VSAFE5V_DESIRED_MIN_VOLTAGE () & less than CONFIG_SNK_VSAFE5V_DESIRED_MAX_VOLTAGE. CONFIG_OVER_VOLTAGE_FACTOR () * 5000mV will be considered as overvoltage for Vsafe5V for sink.

Valid Vsafe5V condition: CONFIG_SNK_VSAFE5V_DESIRED_MAX_VOLTAGE > Valid Vsafe5V <= CONFIG_SNK_VSAFE5V_DESIRED_MIN_VOLTAGE (🔗) Overvoltage condition: Vsafe5V >= CONFIG_OVER_VOLTAGE_FACTOR (🔗) * 5000

Remarks

It is mandatory to define CONFIG_SNK_VSAFE5V_DESIRED_MAX_VOLTAGE. It must be defined in such a way that following condition is met. CONFIG_SNK_VSAFE5V_DESIRED_MAX_VOLTAGE < CONFIG_OVER_VOLTAGE_FACTOR (🔗) * TYPEC_VBUS_5V. The default value for this macro is 5500mV.

Example

```
#define CONFIG_SNK_VSAFE5V_DESIRED_MAX_VOLTAGE          5500
```

3.6.4 CONFIG_SNK_VSAFE5V_DESIRED_MIN_VOLTAGE

C

```
#define CONFIG_SNK_VSAFE5V_DESIRED_MIN_VOLTAGE 4400
```

Description

CONFIG_SNK_VSAFE5V_DESIRED_MIN_VOLTAGE is minimum voltage acceptable for VSafe5V expressed in terms of millivolts for Sink. The voltage will be considered as valid Vsafe5V only if it is equal to or greater than CONFIG_SNK_VSAFE5V_DESIRED_MIN_VOLTAGE & less than CONFIG_SNK_VSAFE5V_DESIRED_MAX_VOLTAGE (🔗). Valid Vsafe5V condition: CONFIG_SNK_VSAFE5V_DESIRED_MAX_VOLTAGE (🔗) > Valid Vsafe5V <= CONFIG_SNK_VSAFE5V_DESIRED_MIN_VOLTAGE

Remarks

It is mandatory to define CONFIG_SRC_VSAFE5V_DESIRED_MIN_VOLTAGE (🔗). It must be defined in such a way that following condition is met. CONFIG_SNK_VSAFE5V_DESIRED_MIN_VOLTAGE > CONFIG_VSINKDISCONNECT_VOLTAGE (🔗). By default, it is defined as 4400mV.

Example

```
#define CONFIG_SNK_VSAFE5V_DESIRED_MIN_VOLTAGE          4400
```

3.6.5 CONFIG_VSINKDISCONNECT_VOLTAGE

C

```
#define CONFIG_VSINKDISCONNECT_VOLTAGE 3670
```

Description

CONFIG_VSINKDISCONNECT_VOLTAGE is the vSinkDisconnect mentioned in Type c specification v1.3. Specification defines it as threshold used for transition from Attached.SNK to Unattached.SNK. In PSF, CONFIG_VSINKDISCONNECT_VOLTAGE is considered as undervoltage for Vsafe5V in case of source. For Sink, if the voltage is below CONFIG_VSINKDISCONNECT_VOLTAGE, it is considered as VBUS disconnect.

For Sink: If Voltage <= CONFIG_VSINKDISCONNECT_VOLTAGE -> Sink disconnected For Source: If Voltage <= CONFIG_VSINKDISCONNECT_VOLTAGE -> Source undervoltage

Remarks

By default, it is defined as 3.67V

Example

```
#define CONFIG_VSINKDISCONNECT_VOLTAGE          3670
```

3.7 DC_DC Configuration

The section explains the option to configure DC-DC control.

3.7.1 CONFIG_DCDC_CTRL

C

```
#define CONFIG_DCDC_CTRL PWRCTRL_DEFAULT_PSF_GPIO_CONFIG
```

Description

CONFIG_DCDC_CTRL is to define the default DC-DC control provided by the PSF stack. If CONFIG_DCDC_CTRL defined as PWRCTRL_DEFAULT_PSF_GPIO_CONFIG, default GPIO based DC-DC controller is used. If left undefined, default stack's DC-DC control option is not used. User has to config via Power control APIs provided by the stack.

Remarks

None.

Example

```
    #define CONFIG_DCDC_CTRL    PWRCTRL_DEFAULT_PSF_GPIO_CONFIG
        (Uses default GPIO based DC-DC contol)
    #define CONFIG_DCDC_CTRL
        (If undefined, Default DC DC control provided by stack is not used)
```

3.7.2 GPIO Based DC-DC control configuration

The section describes the various configurable option available GPIO based DC-DC control.

3.7.2.1 eFAULT_IN_MODE_TYPE Enumeration

C

```
typedef enum {
  eFAULT_IN_ACTIVE_LOW = 0x20U,
  eFAULT_IN_ACTIVE_HIGH = 0x10U,
  eFAULT_IN_ACTIVE_LOW_PU = 0xA0U,
  eFAULT_IN_ACTIVE_HIGH_PD = 0x50U
} eFAULT_IN_MODE_TYPE;
```

Description

eFAULT_IN_MODE_TYPE enum defines the various combination modes applicable for UPD350 GPIO in input mode.

Members

Members	Description
eFAULT_IN_ACTIVE_LOW = 0x20U	Active low input signal
eFAULT_IN_ACTIVE_HIGH = 0x10U	Active high input signal
eFAULT_IN_ACTIVE_LOW_PU = 0xA0U	Active low signal with internal pull up
eFAULT_IN_ACTIVE_HIGH_PD = 0x50U	Active high signal with internal pull down

Remarks

None

3.7.2.2 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.

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

3.7.2.3 CONFIG_PORT_n_UPD_DC_DC_EN_PIO_NO

C

```
#define CONFIG_PORT_n_UPD_DC_DC_EN_PIO_NO eUPD_PIO6
```

Description

CONFIG_PORT_n_UPD_DC_DC_EN_PIO_NO defines the UPD350 PIO to enable DC-DC controller. It is asserted as per CONFIG_PORT_n_UPD_DC_DC_EN_PIO_MODE () during initialization and de-asserted during error condition to reset the DC-DC controller n can take values between 0 and (CONFIG_PD_PORT_COUNT ()-1). It takes value from 0 to 15 and to disable the pin functionality from stack, user can define it as 0xFF. It is applicable only when CONFIG_DCDC_CTRL () is defined as PWRCTRL_DEFAULT_PSF_GPIO_CONFIG.

Remarks

By default, it is configured to PIO6. User can also use stack's enum eUPD_PIO_NUM_TYPE to define this.

Example

```
#define CONFIG_PORT_0_UPD_VBUS_DIS_PIO_NO          6 (DC_DC_EN is PIO6)
#define CONFIG_PORT_0_UPD_VBUS_DIS_PIO_NO          0xFF (DC_DC_EN functinality disabled)
```

3.7.2.4 CONFIG_PORT_n_UPD_DC_DC_EN_PIO_MODE

C

```
#define CONFIG_PORT_n_UPD_DC_DC_EN_PIO_MODE ePUSH_PULL_ACTIVE_HIGH
```

Description

CONFIG_PORT_n_UPD_DC_DC_EN_PIO_MODE defines the PIO mode of the UPD350 PIO DC_DC_EN defined in CONFIG_PORT_n_UPD_DC_DC_EN_PIO_NO (). It takes only values from enum eUPD_OUTPUT_PIN_MODES_TYPE (). n can take values between 0 and (CONFIG_PD_PORT_COUNT ()-1).

Remarks

By default, it is configured to ePUSH_PULL_ACTIVE_HIGH.

Example

```
#define CONFIG_PORT_0_UPD_DC_DC_EN_PIO_MODE          ePUSH_PULL_ACTIVE_HIGH
(DC_DC_EN is configured as Active signal in output mode)
```

3.7.2.5 CONFIG_PORT_n_UPD_EN_VBUS

C

```
#define CONFIG_PORT_n_UPD_EN_VBUS eUPD_PIO3
```

Description

CONFIG_PORT_n_EN_VBUS_UPD_PIO refers to the UPD350 PIO number used for EN_VBUS pin functionality for the nth Port. 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 CONFIG_PORT_n_UPD_EN_VBUS_PIO_MODE () configuration mode whenever stack requires VBUS to driven high as well as low. n can take values between 0 and CONFIG_PD_PORT_COUNT () - 1. It takes value from 0 to 15 and to disable the pin functionality from stack, user can define it as 0xFF. It is applicable only when CONFIG_DCDC_CTRL () is defined as PWRCTRL_DEFAULT_PSF_GPIO_CONFIG and for Source operation only. By defining INCLUDE_UPD_PIO_OVERRIDE_SUPPORT () as '1', The PIO Override feature of the UPD350 shall be utilized in this pin to ensure that fast and autonomous action is taken by the UPD350 in a fault condition.

Remarks

By default, it is configured to PIO3. User can also use stack's enum eUPD_PIO_NUM_TYPE to define this.

Example

```
#define CONFIG_PORT_0_UPD_EN_VBUS       3    (EN_VBUS pin is PIO3)
#define CONFIG_PORT_0_UPD_EN_VBUS       0xFF (EN_VBUS functinality disabled)
```

3.7.2.6 CONFIG_PORT_n_UPD_EN_VBUS_PIO_MODE

C

```
#define CONFIG_PORT_n_UPD_EN_VBUS_PIO_MODE ePUSH_PULL_ACTIVE_HIGH
```

Description

CONFIG_PORT_n_UPD_EN_VBUS_PIO_MODE defines the PIO mode of the UPD350 PIO EN_VBUS defined in CONFIG_PORT_n_UPD_EN_VBUS (). It takes only values from enum eUPD_OUTPUT_PIN_MODES_TYPE (). n can take values between 0 and CONFIG_PD_PORT_COUNT () - 1.

Remarks

By default, it is configured to ePUSH_PULL_ACTIVE_HIGH.

Example

```
#define CONFIG_PORT_0_UPD_EN_VBUS_PIO_MODE          ePUSH_PULL_ACTIVE_HIGH
```

```
(EN_VBUS is configured as Active signal in output mode)
```

3.7.2.7 CONFIG_PORT_n_UPD_VBUS_DIS_PIO_NO

C

```
#define CONFIG_PORT_n_UPD_VBUS_DIS_PIO_NO eUPD_PIO4
```

Description

CONFIG_PORT_n_UPD_VBUS_DIS_PIO_NO defines the UPD350 PIO for VBUS discharge functionality. It is a control for discharging VBUS (connecting VBUS to GND). It asserts as per CONFIG_PORT_n_UPD_VBUS_DIS_PIO_MODE (🔗) whenever VBUS voltage must transition from a high voltage to a lower voltage state and when VBUS is disabled. n can take values between 0 and (CONFIG_PD_PORT_COUNT (🔗)-1). It takes value from 0 to 15 and to disable the pin functionality from stack, user can define it as 0xFF. It is applicable only when CONFIG_DCDC_CTRL (🔗) is defined as PWRCTRL_DEFAULT_PSF_GPIO_CONFIG.

Remarks

By default, it is configured to PIO4. User can also use stack's enum eUPD_PIO_NUM_TYPE to define this.

Example

```
#define CONFIG_PORT_0_UPD_VBUS_DIS_PIO_NO          eUPD_PIO4 (VBUS_DIS is PIO4)
#define CONFIG_PORT_0_UPD_VBUS_DIS_PIO_NO          0xFF (EN_VBUS functinality disabled)
```

3.7.2.8 CONFIG_PORT_n_UPD_VBUS_DIS_PIO_MODE

C

```
#define CONFIG_PORT_n_UPD_VBUS_DIS_PIO_MODE ePUSH_PULL_ACTIVE_HIGH
```

Description

CONFIG_PORT_n_UPD_VBUS_DIS_PIO_MODE defines the PIO mode of the UPD350 PIO VBUS_DIS defined in CONFIG_PORT_n_UPD_VBUS_DIS_PIO_NO (🔗). It takes only values from enum eUPD_OUTPUT_PIN_MODES_TYPE (🔗). n can take values between 0 and (CONFIG_PD_PORT_COUNT (🔗)-1).

Remarks

By default, it is configured to ePUSH_PULL_ACTIVE_HIGH.

Example

```
#define CONFIG_PORT_0_UPD_VBUS_DIS_PIO_MODE          ePUSH_PULL_ACTIVE_HIGH
(VBUS_DIS is configured as Active signal in output mode)
```

3.7.2.9 CONFIG_PORT_n_UPD_VSELx_PIO_NO

C

```
#define CONFIG_PORT_n_UPD_VSELx_PIO_NO eUPD_PIO7
```

Description

CONFIG_PORT_n_UPD_VSELx_PIO_NO defines the UPD350 PIO as voltage selector pins(VSEL[2:0]). PSF stack provides provision for three Voltage selector pin VSEL[2:0]. It is used to control the output voltage of the DC/DC controller. In a typical application, these pins are used to switch in different resistors into the feedback loop to vary the output voltage. n can take values between 0 and CONFIG_PD_PORT_COUNT (🔗) - 1. x takes vlaue between 0 to 2. This define takes value from 0 to 15 and to disable the pin functionality from stack, user can define it as 0xFF. It is applicable only when CONFIG_DCDC_CTRL (🔗) is defined as PWRCTRL_DEFAULT_PSF_GPIO_CONFIG.

Remarks

By default, VSEL0, VSEL1, VSEL2 is configured to PIO7, PIO8 and PIO9 respectively. User can also use stack's enum eUPD_PIO_NUM_TYPE to define this. It is applicable only for source operation.

Example

```
#define CONFIG_PORT_0_UPD_VSEL0_PIO_NO          7 (VSEL0 for port 0 is PIO7)
#define CONFIG_PORT_0_UPD_VSEL1_PIO_NO          8 (VSEL1 for port 0 is PIO8)
#define CONFIG_PORT_0_UPD_VSEL2_PIO_NO          9 (VSEL2 for port 0 is PIO9)
#define CONFIG_PORT_0_UPD_VSEL0_PIO_NO          0xFF (VSEL0 for port 0 is disabled)
```

3.7.2.10 CONFIG_PORT_n_UPD_VSELx_PIO_MODE

C

```
#define CONFIG_PORT_n_UPD_VSELx_PIO_MODE ePUSH_PULL_ACTIVE_HIGH
```

Description

CONFIG_PORT_n_UPD_VSELx_PIO_MODE defines the PIO mode of the UPD350 PIO VSELx defined in CONFIG_PORT_n_UPD_VSELx_PIO_NO (). It takes only values from enum eUPD_OUTPUT_PIN_MODES_TYPE (). n can take values between 0 and (CONFIG_PD_PORT_COUNT ()-1). x takes vlaue between 0 to 2.

Remarks

By default, it is configured to ePUSH_PULL_ACTIVE_HIGH.

Example

```
#define CONFIG_PORT_0_UPD_VSEL0_PIO_MODE          ePUSH_PULL_ACTIVE_HIGH
(VSEL0 is configured as Active signal in output mode)
```

3.7.2.11 CONFIG_PORT_n_VSAFE0V_VSEL_MAPPING

C

```
#define CONFIG_PORT_n_VSAFE0V_VSEL_MAPPING 0x00
```

Description

CONFIG_PORT_n_VSAFE0V_VSEL_MAPPING defines the assertion and de-assertion to be driven on VSEL[2:0] pins(defined in CONFIG_PORT_n_UPD_VSELx_PIO_NO ()) by the PSF stack as per CONFIG_PORT_n_UPD_VSELx_PIO_MODE () to have a output voltage of VSafe0V out of DC-Dc controller. n can take values between 0 and (CONFIG_PD_PORT_COUNT ()-1).

Remarks

By default, it is configured to 0x00. It is applicable only for source operation.

Example

```
#define CONFIG_PORT_0_VSAFE0V_VSEL_MAPPING          0x00
```

3.7.2.12 CONFIG_PORT_n_PDO_x_VSEL_MAPPING

C

```
#define CONFIG_PORT_n_PDO_x_VSEL_MAPPING 0x00
```

Description

CONFIG_PORT_n_VSAFE0V_VSEL_MAPPING () defines the assertion and de-assertion to be driven on VSEL[2:0] pins(defined in CONFIG_PORT_n_UPD_VSELx_PIO_NO ()) by the PSF stack as per CONFIG_PORT_n_UPD_VSELx_PIO_MODE () to have a output voltage of PDO voltage defined in CONFIG_PORT_n_SOURCE_PDO_x_VOLTAGE () out of DC-DC controller. n can take values between 0 and (CONFIG_PD_PORT_COUNT ()-1). x takes value between 1 to 7 as by PD specification, there can only be 7 PDOs. It is applicable only for source.

Remarks

By default, a 1 pin per voltage implementation is implemented. VSEL[2:0]: ?000? - 5V (No pins asserted) ?001? - 9V (VSEL0

asserted) ?010? - 15V (VSEL1 asserted) ?100? - 20V (VSEL2 asserted)

Example

```
#define CONFIG_PORT_0_PDO_1_VSEL_MAPPING    0x00
#define CONFIG_PORT_0_PDO_2_VSEL_MAPPING    0x01
#define CONFIG_PORT_0_PDO_3_VSEL_MAPPING    0x02
#define CONFIG_PORT_0_PDO_4_VSEL_MAPPING    0x04
#define CONFIG_PORT_0_PDO_5_VSEL_MAPPING    0x00
#define CONFIG_PORT_0_PDO_6_VSEL_MAPPING    0x00
#define CONFIG_PORT_0_PDO_7_VSEL_MAPPING    0x00
```

3.7.2.13 CONFIG_PORT_n_UPD_FAULT_IN_PIO_NO

C

```
#define CONFIG_PORT_n_UPD_FAULT_IN_PIO_NO eUPD_PIO5
```

Description

CONFIG_PORT_n_UPD_FAULT_IN_PIO_NO defines the UPD PIO used as FAULT_IN pin. FAULT_IN detects over-current fault or under/over-voltage fault from external sensing device based on its configuration CONFIG_PORT_n_UPD_FAULT_IN_MODE (). It takes value from 0 to 15 and to disable the pin functionality from stack, user can define it as 0xFF. It is applicable only when CONFIG_DCDC_CTRL () is defined as PWRCTRL_DEFAULT_PSF_GPIO_CONFIG and INCLUDE_POWER_FAULT_HANDLING () defined as '1'.

Remarks

By default, it is defined as 5.n can take values between 0 and (CONFIG_PD_PORT_COUNT ()-1).

Example

```
#define CONFIG_PORT_0_UPD_FAULT_IN          5 (FAULT_IN is PIO5)
#define CONFIG_PORT_0_UPD_FAULT_IN          0xFF (FAULT_IN functinality disabled)
```

3.7.2.14 CONFIG_PORT_n_UPD_FAULT_IN_MODE

C

```
#define CONFIG_PORT_n_UPD_FAULT_IN_MODE eFAULT_IN_ACTIVE_LOW
```

Description

CONFIG_PORT_n_UPD_FAULT_IN_MODE defines the PIO mode of the UPD350 PIO FAULT_IN defined in CONFIG_PORT_n_UPD_FAULT_IN_PIO_NO (). It takes only values from enum eFAULT_IN_MODE_TYPE (). n can take values between 0 and (CONFIG_PD_PORT_COUNT ()-1).

Remarks

By default, it is configured to ePUSH_PULL_ACTIVE_HIGH.

Example

```
#define CONFIG_PORT_n_UPD_FAULT_IN_MODE          eFAULT_IN_ACTIVE_LOW
(FAULT_IN is configured as Active low in input mode)
```

3.8 PDFU Configuration

This section explains the configurable options available for PDFU.

3.8.1 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 TRUE. 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
```

3.8.2 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 TRUE. 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
```

3.8.3 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)
```

3.8.4 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 TRUE. It can have values from 0x00 to 0xFF. By default, it is defined as '0x03'.

Example

```
#define CONFIG_VALIDATION_PHASE_WAITTIME   0x03u
```

3.8.5 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
```

3.8.6 CONFIG_PDFU_VIA_USBPD_SUPPORTED

C

```
#define CONFIG_PDFU_VIA_USBPD_SUPPORTED 1
```

Description

CONFIG_PDFU_VIA_USBPD_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_USBPD_SUPPORTED  1
```

3.8.7 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(32KB).

Example

```
 #define CONFIG_MAX_FIRMWARE_IMAGESIZE 38*1024 (38*1024 bytes for 38KB Updatable
application).
```

3.9 MCU Idle Timeout Configuration

3.9.1 CONFIG_PORT_UPD_IDLE_TIMEOUT_MS

C

```
#define CONFIG_PORT_UPD_IDLE_TIMEOUT_MS MILLISECONDS_TO_TICKS(15000)
```

Description

CONFIG_PORT_UPD_IDLE_TIMEOUT_MS is the idle time after which UPD350 is put to low power mode by the power management control if there is no activity or interrupt in UPD350.

Remarks

It shall be expressed in MILLISECONDS_TO_TICKS define. CONFIG_PORT_UPD_IDLE_TIMEOUT_MS is valid only if INCLUDE_POWER_MANAGEMENT_CTRL (🔗) set as 1. By default, this define is set to to 15seconds.

Example

```
 #define CONFIG_PORT_UPD_IDLE_TIMEOUT_MS      MILLISECONDS_TO_TICKS(15000) (Timeout is 15
seconds)
```

3.10 Type-C and PD Specification defined Timeout configuration

PSF provides defines for the user to configure various Timeouts specified by Type-C and USB Power Delivery Specification.

3.10.1 CONFIG_TYPEC_TCCDEBOUNCE_TIMEOUT_MS

C

```
#define CONFIG_TYPEC_TCCDEBOUNCE_TIMEOUT_MS MILLISECONDS_TO_TICKS(150)
```

Description

CONFIG_TYPEC_TCCDEBOUNCE_TIMEOUT_MS defines the tCCDebounce timeout specified in the USB Type C Specification. Default value of CONFIG_TYPEC_TCCDEBOUNCE_TIMEOUT_MS is set as 150 milliseconds.

Remarks

CONFIG_TYPEC_TCCDEBOUNCE_TIMEOUT_MS can be configured depending on the microcontroller platform platform used, for the device to be USB Type C Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_TYPEC_TCCDEBOUNCE_TIMEOUT_MS      MILLISECONDS_TO_TICKS(150)
```

3.10.2 CONFIG_TYPEC_TPDEBOUNCE_TIMEOUT_MS

C

```
#define CONFIG_TYPEC_TPDEBOUNCE_TIMEOUT_MS MILLISECONDS_TO_TICKS(10)
```

Description

CONFIG_TYPEC_TPDEBOUNCE_TIMEOUT_MS defines the tPDDebounce timeout specified in the USB Type C Specification. Default value of this macro is set as 10 milliseconds.

Remarks

CONFIG_TYPEC_TPDEBOUNCE_TIMEOUT_MS can can be configured depending on the microcontroller platform platform used, for the device to be USB Type C Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_TYPEC_TPDEBOUNCE_TIMEOUT_MS      MILLISECONDS_TO_TICKS(10)
```

3.10.3 CONFIG_TYPEC_VBUS_OFF_TIMER_MS

C

```
#define CONFIG_TYPEC_VBUS_OFF_TIMER_MS MILLISECONDS_TO_TICKS(650)
```

Description

CONFIG_TYPEC_VBUS_OFF_TIMER_MS defines the tVBUSOFF specified in the USB-TypeC Specification. Default value of CONFIG_TYPEC_VBUS_OFF_TIMER_MS is set as 650 milliseconds.

Remarks

CONFIG_TYPEC_VBUS_OFF_TIMER_MS can be configured depending on the microcontroller platform used, for the device to be USB Type C Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_TYPEC_VBUS_OFF_TIMER_MS                MILLISECONDS_TO_TICKS(650)
```

3.10.4 CONFIG_TYPEC_VBUS_ON_TIMER_MS

C

```
#define CONFIG_TYPEC_VBUS_ON_TIMER_MS MILLISECONDS_TO_TICKS(275)
```

Description

CONFIG_TYPEC_VBUS_ON_TIMER_MS defines the tVBUSON specified in the USB-TypeC Specification. Default value of CONFIG_TYPEC_VBUS_ON_TIMER_MS is set as 275 milliseconds.

Remarks

CONFIG_TYPEC_VBUS_ON_TIMER_MS can be configured depending on the microcontroller platform used, for the device to be USB Type C Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_TYPEC_VBUS_ON_TIMER_MS                    MILLISECONDS_TO_TICKS(275)
```

3.10.5 CONFIG_TYPEC_VCONNOFF_TIMEOUT_MS

C

```
#define CONFIG_TYPEC_VCONNOFF_TIMEOUT_MS MILLISECONDS_TO_TICKS(25)
```

Description

CONFIG_TYPEC_VCONNOFF_TIMEOUT_MS defines the tVCONNOFF specified in the USB-Type C Specification. Default value of CONFIG_TYPEC_VCONNOFF_TIMEOUT_MS is set as 25 milliseconds.

Remarks

CONFIG_TYPEC_VCONNOFF_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_TYPEC_VCONNOFF_TIMEOUT_MS                MILLISECONDS_TO_TICKS(25)
```

3.10.6 CONFIG_TYPEC_VCONNON_TIMEOUT_MS

C

```
#define CONFIG_TYPEC_VCONNON_TIMEOUT_MS MILLISECONDS_TO_TICKS(10)
```

Description

CONFIG_TYPEC_VCONNON_TIMEOUT_MS defines the tVCONNON specified in the USB-Type C Specification. Default value of CONFIG_TYPEC_VCONNON_TIMEOUT_MS is set as 2 milliseconds.

Remarks

CONFIG_TYPEC_VCONNON_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB Type-C Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_TYPEC_VCONNON_TIMEOUT_MS              MILLISECONDS_TO_TICKS(10)
```

3.10.7 CONFIG_TYPEC_VCONNDISCHARGE_TIMEOUT_MS

C

```
#define CONFIG_TYPEC_VCONNDISCHARGE_TIMEOUT_MS MILLISECONDS_TO_TICKS(35)
```

Description

CONFIG_TYPEC_VCONNDISCHARGE_TIMEOUT_MS defines the tVCONNDischarge timeout specified in the USB Type C Specification. Default value of CONFIG_TYPEC_VCONNDISCHARGE_TIMEOUT_MS is set as 35 milliseconds.

Remarks

CONFIG_TYPEC_VCONNDISCHARGE_TIMEOUT_MS can be configured depending on the microcontroller platform platform used, for the device to be USB Type C Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_TYPEC_VCONNDISCHARGE_TIMEOUT_MS      MILLISECONDS_TO_TICKS(35)
```

3.10.8 CONFIG_TYPEC_ERRORRECOVERY_TIMEOUT_MS

C

```
#define CONFIG_TYPEC_ERRORRECOVERY_TIMEOUT_MS MILLISECONDS_TO_TICKS(500)
```

Description

CONFIG_TYPEC_ERRORRECOVERY_TIMEOUT_MS defines the tErrorRecovery timeout specified in the USB Type C Specification. Default value of CONFIG_TYPEC_ERRORRECOVERY_TIMEOUT_MS is set as 500 milliseconds.

Remarks

CONFIG_TYPEC_ERRORRECOVERY_TIMEOUT_MS can be configured depending on the microcontroller platform platform used, for the device to be USB Type C Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_TYPEC_ERRORRECOVERY_TIMEOUT_MS       MILLISECONDS_TO_TICKS(500)
```

3.10.9 CONFIG_PRL_CHUNKSENDERREQUEST_TIMEOUT_MS

C

```
#define CONFIG_PRL_CHUNKSENDERREQUEST_TIMEOUT_MS MILLISECONDS_TO_TICKS(26)
```

Description

CONFIG_PRL_CHUNKSENDERREQUEST_TIMEOUT_MS defines the ChunkSenderRequestTimer specified in the USB-PD Specification. Default value of CONFIG_PRL_CHUNKSENDERREQUEST_TIMEOUT_MS is set as 26 milliseconds.

Remarks

CONFIG_PRL_CHUNKSENDERREQUEST_TIMEOUT_MS can be configured depending on the microcontroller platform

used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_PRL_CHUNKSENDERREQUEST_TIMEOUT_MS        MILLISECONDS_TO_TICKS(26)
```

3.10.10 CONFIG_PRL_CHUNKSENDERRESPONSE_TIMEOUT_MS

C

```
#define CONFIG_PRL_CHUNKSENDERRESPONSE_TIMEOUT_MS MILLISECONDS_TO_TICKS(26)
```

Description

CONFIG_PRL_CHUNKSENDERRESPONSE_TIMEOUT_MS defines the ChunkSenderResponseTimer specified in the USB-PD Specification. Default value of CONFIG_PRL_CHUNKSENDERRESPONSE_TIMEOUT_MS is set as 26 milliseconds.

Remarks

CONFIG_PRL_CHUNKSENDERRESPONSE_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_PRL_CHUNKSENDERRESPONSE_TIMEOUT_MS    MILLISECONDS_TO_TICKS(26)
```

3.10.11 CONFIG_PRL_SINKTX_TIMEOUT_MS

C

```
#define CONFIG_PRL_SINKTX_TIMEOUT_MS MILLISECONDS_TO_TICKS(16)
```

Description

CONFIG_PRL_SINKTX_TIMEOUT_MS defines the SinkTxTimer specified in the USB-PD Specification. Default value of CONFIG_PRL_SINKTX_TIMEOUT_MS is set as 16 milliseconds.

Remarks

CONFIG_PRL_SINKTX_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_PRL_SINKTX_TIMEOUT_MS                    MILLISECONDS_TO_TICKS(16)
```

3.10.12 CONFIG_PRL_BIST_CONTMODE_TIMEOUT_MS

C

```
#define CONFIG_PRL_BIST_CONTMODE_TIMEOUT_MS MILLISECONDS_TO_TICKS(45)
```

Description

CONFIG_PRL_BIST_CONTMODE_TIMEOUT_MS defines the BISTContModeTimer specified in the USB-PD Specification. Default value of CONFIG_PRL_BIST_CONTMODE_TIMEOUT_MS is set as 45 milliseconds.

Remarks

CONFIG_PRL_BIST_CONTMODE_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_PRL_BIST_CONTMODE_TIMEOUT_MS              MILLISECONDS_TO_TICKS(45)
```

3.10.13 CONFIG_PE_PSHARDRESET_TIMEOUT_MS

C

```
#define CONFIG_PE_PSHARDRESET_TIMEOUT_MS MILLISECONDS_TO_TICKS(28)
```

Description

CONFIG_PE_PSHARDRESET_TIMEOUT_MS defines the PSHardResetTimer specified in the USB-PD Specification. Default value of CONFIG_PE_PSHARDRESET_TIMEOUT_MS is set as 28 milliseconds.

Remarks

CONFIG_PE_PSHARDRESET_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_PE_PSHARDRESET_TIMEOUT_MS              MILLISECONDS_TO_TICKS(28)
```

3.10.14 CONFIG_PE_PSTRANSITION_TIMEOUT_MS

C

```
#define CONFIG_PE_PSTRANSITION_TIMEOUT_MS MILLISECONDS_TO_TICKS(500)
```

Description

CONFIG_PE_PSTRANSITION_TIMEOUT_MS defines the PSTransitionTimer specified in the USB-PD Specification. Default value of CONFIG_PE_PSTRANSITION_TIMEOUT_MS is set as 500 milliseconds.

Remarks

CONFIG_PE_PSTRANSITION_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_PE_PSTRANSITION_TIMEOUT_MS              MILLISECONDS_TO_TICKS(500)
```

3.10.15 CONFIG_PE_SENDERRESPONSE_TIMEOUT_MS

C

```
#define CONFIG_PE_SENDERRESPONSE_TIMEOUT_MS MILLISECONDS_TO_TICKS(26)
```

Description

CONFIG_PE_SENDERRESPONSE_TIMEOUT_MS defines the SenderResponseTimer specified in the USB-PD Specification. Default value of CONFIG_PE_SENDERRESPONSE_TIMEOUT_MS is set as 26 milliseconds.

Remarks

CONFIG_PE_SENDERRESPONSE_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_PE_SENDERRESPONSE_TIMEOUT_MS              MILLISECONDS_TO_TICKS(26)
```

3.10.16 CONFIG_PE_SINKREQUEST_TIMEOUT_MS

C

```
#define CONFIG_PE_SINKREQUEST_TIMEOUT_MS MILLISECONDS_TO_TICKS(100)
```

Description

CONFIG_PE_SINKREQUEST_TIMEOUT_MS defines the SinkRequestTimer specified in the USB-PD Specification. Default value of CONFIG_PE_SINKREQUEST_TIMEOUT_MS is set as 100 milliseconds.

Remarks

CONFIG_PE_SINKREQUEST_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_PE_SINKREQUEST_TIMEOUT_MS                MILLISECONDS_TO_TICKS(100)
```

3.10.17 CONFIG_PE_SINKWAITCAP_TIMEOUT_MS

C

```
#define CONFIG_PE_SINKWAITCAP_TIMEOUT_MS MILLISECONDS_TO_TICKS(465)
```

Description

CONFIG_PE_SINKWAITCAP_TIMEOUT_MS defines the SinkWaitCapTimer specified in the USB-PD Specification. Default value of CONFIG_PE_SINKWAITCAP_TIMEOUT_MS is set as 465 milliseconds.

Remarks

CONFIG_PE_SINKWAITCAP_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_PE_SINKWAITCAP_TIMEOUT_MS                MILLISECONDS_TO_TICKS(465)
```

3.10.18 CONFIG_PE_SOURCECAPABILITY_TIMEOUT_MS

C

```
#define CONFIG_PE_SOURCECAPABILITY_TIMEOUT_MS MILLISECONDS_TO_TICKS(150)
```

Description

CONFIG_PE_SOURCECAPABILITY_TIMEOUT_MS defines the SourceCapabilityTimer specified in the USB-PD Specification. Default value of CONFIG_PE_SOURCECAPABILITY_TIMEOUT_MS is set as 150 milliseconds.

Remarks

CONFIG_PE_SOURCECAPABILITY_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_PE_SOURCECAPABILITY_TIMEOUT_MS        MILLISECONDS_TO_TICKS(150)
```

3.10.19 CONFIG_PE_SRCRECOVER_TIMEOUT_MS

C

```
#define CONFIG_PE_SRCRECOVER_TIMEOUT_MS MILLISECONDS_TO_TICKS(800)
```

Description

CONFIG_PE_SRCRECOVER_TIMEOUT_MS defines the tSrcRecover specified in the USB-PD Specification. Default value of CONFIG_PE_SRCRECOVER_TIMEOUT_MS is set as 800 milliseconds.

Remarks

CONFIG_PE_SRCRECOVER_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_PE_SRCRECOVER_TIMEOUT_MS                MILLISECONDS_TO_TICKS(800)
```

3.10.20 CONFIG_PE_VDMRESPONSE_TIMEOUT_MS

C

```
#define CONFIG_PE_VDMRESPONSE_TIMEOUT_MS MILLISECONDS_TO_TICKS(28)
```

Description

CONFIG_PE_VDMRESPONSE_TIMEOUT_MS defines the VDMResponseTimer specified in the USB-PD Specification. Default value of CONFIG_PE_VDMRESPONSE_TIMEOUT_MS is set as 28 milliseconds.

Remarks

CONFIG_PE_VDMRESPONSE_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_PE_VDMRESPONSE_TIMEOUT_MS               MILLISECONDS_TO_TICKS(28)
```

3.10.21 CONFIG_PE_VCONNON_TIMEOUT_MS

C

```
#define CONFIG_PE_VCONNON_TIMEOUT_MS MILLISECONDS_TO_TICKS(100)
```

Description

CONFIG_PE_VCONNON_TIMEOUT_MS defines the tVCONNSourceOn specified in the USB PD Specification. Default value of CONFIG_PE_VCONNON_TIMEOUT_MS is set as 100 milliseconds.

Remarks

CONFIG_PE_VCONNON_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_PE_VCONNON_TIMEOUT_MS                MILLISECONDS_TO_TICKS(100)
```

3.10.22 CONFIG_PE_SRCTRANSISTION_TIMEOUT_MS

C

```
#define CONFIG_PE_SRCTRANSISTION_TIMEOUT_MS MILLISECONDS_TO_TICKS(28)
```

Description

CONFIG_PE_SRCTRANSISTION_TIMEOUT_MS defines the tSrcTransistionTimer specified in the USB-PD Specification. By default it is set to 30 milliseconds.

Remarks

CONFIG_POWER_GOOD_TIMER_MS (⧉) can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_PE_SRCTRANSISTION_TIMEOUT_MS              MILLISECONDS_TO_TICKS(30)
```

3.10.23 CONFIG_PE_VCONNOFF_TIMEOUT_MS

C

```
#define CONFIG_PE_VCONNOFF_TIMEOUT_MS MILLISECONDS_TO_TICKS(35)
```

Description

CONFIG_PE_VCONNOFF_TIMEOUT_MS defines the tVCONNOFF specified in the USB-Type C Specification. Default value of CONFIG_PE_VCONNOFF_TIMEOUT_MS is set as 35 milliseconds.

Remarks

CONFIG_PE_VCONNOFF_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
 #define CONFIG_PE_VCONNOFF_TIMEOUT_MS               MILLISECONDS_TO_TICKS(35)
```

3.10.24 CONFIG_PE_VCONNON_SELF_TIMEOUT_MS

C

```
#define CONFIG_PE_VCONNON_SELF_TIMEOUT_MS MILLISECONDS_TO_TICKS(150)
```

Description

CONFIG_PE_VCONNON_SELF_TIMEOUT_MS defines the tVCONNSourceOn specified in the USB PD Specification. Default value of CONFIG_PE_VCONNON_SELF_TIMEOUT_MS is set as 100 milliseconds.

Remarks

CONFIG_PE_VCONNON_SELF_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
 #define CONFIG_PE_VCONNON_SELF_TIMEOUT_MS              MILLISECONDS_TO_TICKS(150)
```

3.10.25 CONFIG_PE_NORESPONSE_TIMEOUT_MS

C

```
#define CONFIG_PE_NORESPONSE_TIMEOUT_MS MILLISECONDS_TO_TICKS(5500)
```

Description

CONFIG_PE_NORESPONSE_TIMEOUT_MS defines the NoResponseTimer specified in the USB-PD Specification. Default value of CONFIG_PE_NORESPONSE_TIMEOUT_MS is set as 5.5 seconds.

Remarks

CONFIG_PE_NORESPONSE_TIMEOUT_MS an be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_PE_NORESPONSE_TIMEOUT_MS        MILLISECONDS_TO_TICKS(5500)
```

3.10.26 CONFIG_PE_SRC_READY_TIMEOUT_MS

C

```
#define CONFIG_PE_SRC_READY_TIMEOUT_MS MILLISECONDS_TO_TICKS(285)
```

Description

CONFIG_PE_SRC_READY_TIMEOUT_MS defines the tSrcReady specified in the PD 3.0 Specification. Default value of CONFIG_PE_SRC_READY_TIMEOUT_MS is set as 285 milliseconds.

Remarks

CONFIG_PE_SRC_READY_TIMEOUT_MS can be configured depending on the microcontroller platform used, for the device to be USB PD Compliant. It shall always be expressed in define MILLISECONDS_TO_TICKS.

Example

```
#define CONFIG_PE_SRC_READY_TIMEOUT_MS                MILLISECONDS_TO_TICKS(285)
```

4 Appendix-I

4.1 Data types

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
SIZEOF(x)	sizeof(x)