Xilinx Standalone Library Documentation

XilPM Library v3.2

UG1225 (v2020.2) December 16, 2020

XILINX®

Table of Contents

Chapter 1

XilPM Zynq UltraScale+ MPSoC APIs

Xilinx Power Management (XilPM) provides Embedded Energy Management Interface (EEMI) APIs for power management on Zynq UltraScale+ MPSoC. For more details about EEMI, see the Embedded Energy Management Interface (EEMI) API User Guide (UG1200).

Table 1: **Quick Function Reference**

Type	Name	Arguments
XStatus	XPm_InitXilpm	XIpiPsu * IpiInst
enum XPmBootStatus	XPm_GetBootStatus	void
void	XPm_SuspendFinalize	void
XStatus	pm_ipi_send	struct XPm_Master *const master u32 payload
XStatus	pm_ipi_buff_read32	struct XPm_Master *const master u32 * value1 u32 * value2 u32 * value3
XStatus	XPm_SelfSuspend	const enum XPmNodeId nid const u32 latency const u8 state const u64 address
XStatus	XPm_SetConfiguration	const u32 address
XStatus	XPm_InitFinalize	void
XStatus	XPm_RequestSuspend	const enum XPmNodeId target const enum XPmRequestAck ack const u32 latency const u8 state

Table 1: **Quick Function Reference** *(cont'd)*

Type	Name	Arguments
XStatus	XPm_RequestWakeUp	const enum XPmNodeId target const bool setAddress const u64 address const enum XPmRequestAck ack
XStatus	XPm_ForcePowerDown	const enum XPmNodeId target const enum XPmRequestAck ack
XStatus	XPm_AbortSuspend	const enum XPmAbortReason reason
XStatus	XPm_SetWakeUpSource	const enum XPmNodeId target const enum XPmNodeId wkup_node const u8 enable
XStatus	XPm_SystemShutdown	restart
XStatus	XPm_RequestNode	const enum XPmNodeId node const u32 capabilities const u32 qos const enum XPmRequestAck ack
XStatus	XPm_SetRequirement	const enum XPmNodeId nid const u32 capabilities const u32 qos const enum XPmRequestAck ack
XStatus	XPm_ReleaseNode	const enum XPmNodeId node
XStatus	XPm_SetMaxLatency	const enum XPmNodeId node const u32 latency
void	XPm_InitSuspendCb	const enum XPmSuspendReason reason const u32 latency const u32 state const u32 timeout
void	XPm_AcknowledgeCb	const enum XPmNodeId node const XStatus status const u32 oppoint

Table 1: **Quick Function Reference** *(cont'd)*

Type	Name	Arguments
void	XPm_NotifyCb	const enum XPmNodeId node const enum XPmNotifyEvent event const u32 oppoint
XStatus	XPm_GetApiVersion	u32 * version
XStatus	XPm_GetNodeStatus	const enum XPmNodeId node XPm_NodeStatus *const nodestatus
XStatus	XPm_GetOpCharacteristic	const enum XPmNodeId node const enum XPmOpCharType type u32 *const result
XStatus	XPm_ResetAssert	const enum XPmReset reset assert
XStatus	XPm_ResetGetStatus	const enum XPmReset reset u32 * status
XStatus	XPm_RegisterNotifier	XPm_Notifier *const notifier
XStatus	XPm_UnregisterNotifier	XPm_Notifier *const notifier
XStatus	XPm_MmioWrite	const u32 address const u32 mask const u32 value
XStatus	XPm_MmioRead	const u32 address u32 *const value
XStatus	XPm_ClockEnable	const enum XPmClock clk
XStatus	XPm_ClockDisable	const enum XPmClock clk
XStatus	XPm_ClockGetStatus	const enum XPmClock clk u32 *const status
XStatus	XPm_ClockSetOneDivider	const enum XPmClock clk const u32 divider const u32 divId

Table 1: **Quick Function Reference** *(cont'd)*

Type	Name	Arguments
XStatus	XPm_ClockSetDivider	const enum XPmClock clk const u32 divider
XStatus	XPm_ClockGetOneDivider	const enum XPmClock clk u32 *const divider
XStatus	XPm_ClockGetDivider	const enum XPmClock clk u32 *const divider
XStatus	XPm_ClockSetParent	const enum XPmClock clk const enum XPmClock parent
XStatus	XPm_ClockGetParent	const enum XPmClock clk enum XPmClock *const parent
XStatus	XPm_ClockSetRate	const enum XPmClock clk const u32 rate
XStatus	XPm_ClockGetRate	const enum XPmClock clk u32 *const rate
XStatus	XPm_PllSetParameter	const enum XPmNodeId node const enum XPmPllParam parameter const u32 value
XStatus	XPm_PllGetParameter	const enum XPmNodeId node const enum XPmPllParam parameter u32 *const value
XStatus	XPm_PllSetMode	const enum XPmNodeId node const enum XPmPllMode mode
XStatus	XPm_PllGetMode	const enum XPmNodeId node enum XPmPllMode *const mode
XStatus	XPm_PinCtrlAction	const u32 pin
XStatus	XPm_PinCtrlRequest	const u32 pin
XStatus	XPm_PinCtrlRelease	const u32 pin

Table 1: **Quick Function Reference** *(cont'd)*

Type	Name	Arguments
XStatus	XPm_PinCtrlSetFunction	const u32 pin const enum XPmPinFn fn
XStatus	XPm_PinCtrlGetFunction	const u32 pin enum XPmPinFn *const fn
XStatus	XPm_PinCtrlSetParameter	const u32 pin const enum XPmPinParam param const u32 value
XStatus	XPm_PinCtrlGetParameter	const u32 pin const enum XPmPinParam param u32 *const value

Functions

XPm_InitXilpm

Initialize xilpm library.

***Note*:** None

Prototype

```
XStatus XPm_InitXilpm(XIpiPsu *IpiInst);
```

Parameters

The following table lists the `XPm_InitXilpm` function arguments.

Table 2: **XPm_InitXilpm Arguments**

Type	Name	Description
XIpiPsu *	IpiInst	Pointer to IPI driver instance

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_GetBootStatus

This Function returns information about the boot reason. If the boot is not a system startup but a resume, power down request bitfield for this processor will be cleared.

Note: None

Prototype

```
enum
                    XPmBootStatus
                 XPm_GetBootStatus(void);
```

Returns

Returns processor boot status

- PM_RESUME : If the boot reason is because of system resume.
- PM_INITIAL_BOOT : If this boot is the initial system startup.

XPm_SuspendFinalize

This Function waits for PMU to finish all previous API requests sent by the PU and performs client specific actions to finish suspend procedure (e.g. execution of wfi instruction on A53 and R5 processors).

Note: This function should not return if the suspend procedure is successful.

Prototype

```
void XPm_SuspendFinalize(void);
```

Returns

pm_ipi_send

Sends IPI request to the PMU.

Note: None

Prototype

```
XStatus pm_ipi_send(struct XPm_Master *const master, u32
payload[PAYLOAD_ARG_CNT]);
```

Parameters

The following table lists the `pm_ipi_send` function arguments.

Table 3: **pm_ipi_send Arguments**

Type	Name	Description
struct `XPm_Master` *const	master	Pointer to the master who is initiating request
u32	payload	API id and call arguments to be written in IPI buffer

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

pm_ipi_buff_read32

Reads IPI response after PMU has handled interrupt.

Note: None

Prototype

```
XStatus pm_ipi_buff_read32(struct XPm_Master *const master, u32 *value1,
u32 *value2, u32 *value3);
```

Parameters

The following table lists the `pm_ipi_buff_read32` function arguments.

Table 4: **pm_ipi_buff_read32 Arguments**

Type	Name	Description
struct `XPm_Master` *const	master	Pointer to the master who is waiting and reading response
u32 *	value1	Used to return value from 2nd IPI buffer element (optional)
u32 *	value2	Used to return value from 3rd IPI buffer element (optional)
u32 *	value3	Used to return value from 4th IPI buffer element (optional)

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_SelfSuspend

This function is used by a CPU to declare that it is about to suspend itself. After the PMU processes this call it will wait for the requesting CPU to complete the suspend procedure and become ready to be put into a sleep state.

Note: This is a blocking call, it will return only once PMU has responded

Prototype

```
XStatus XPm_SelfSuspend(const enum XPmNodeId nid, const u32 latency, const
u8 state, const u64 address);
```

Parameters

The following table lists the `XPm_SelfSuspend` function arguments.

Table 5: **XPm_SelfSuspend Arguments**

Type	Name	Description
const enum `XPmNodeId`	nid	Node ID of the CPU node to be suspended.
const u32	latency	Maximum wake-up latency requirement in us(microsecs)
const u8	state	Instead of specifying a maximum latency, a CPU can also explicitly request a certain power state.
const u64	address	Address from which to resume when woken up.

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_SetConfiguration

This function is called to configure the power management framework. The call triggers power management controller to load the configuration object and configure itself according to the content of the object.

Note: The provided address must be in 32-bit address space which is accessible by the PMU.

Prototype

```
XStatus XPm_SetConfiguration(const u32 address);
```

Parameters

The following table lists the `XPm_SetConfiguration` function arguments.

Table 6: **XPm_SetConfiguration Arguments**

Type	Name	Description
const u32	address	Start address of the configuration object

Returns

XST_SUCCESS if successful, otherwise an error code

XPm_InitFinalize

This function is called to notify the power management controller about the completed power management initialization.

Note: It is assumed that all used nodes are requested when this call is made. The power management controller may power down the nodes which are not requested after this call is processed.

Prototype

```
XStatus XPm_InitFinalize(void);
```

Returns

XST_SUCCESS if successful, otherwise an error code

XPm_RequestSuspend

This function is used by a PU to request suspend of another PU. This call triggers the power management controller to notify the PU identified by 'nodeID' that a suspend has been requested. This will allow said PU to gracefully suspend itself by calling XPm_SelfSuspend for each of its CPU nodes, or else call XPm_AbortSuspend with its PU node as argument and specify the reason.

Note: If 'ack' is set to PM_ACK_NON_BLOCKING, the requesting PU will be notified upon completion of suspend or if an error occurred, such as an abort. REQUEST_ACK_BLOCKING is not supported for this command.

Prototype

```
XStatus XPm_RequestSuspend(const enum XPmNodeId target, const enum
XPmRequestAck ack, const u32 latency, const u8 state);
```

Parameters

The following table lists the `XPm_RequestSuspend` function arguments.

Table 7: **XPm_RequestSuspend Arguments**

Type	Name	Description
const enum XPmNodeId	target	Node ID of the PU node to be suspended
const enum XPmRequestAck	ack	Requested acknowledge type
const u32	latency	Maximum wake-up latency requirement in us(micro sec)
const u8	state	Instead of specifying a maximum latency, a PU can also explicitly request a certain power state.

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_RequestWakeUp

This function can be used to request power up of a CPU node within the same PU, or to power up another PU.

Note: If acknowledge is requested, the calling PU will be notified by the power management controller once the wake-up is completed.

Prototype

```
XStatus XPm_RequestWakeUp(const enum XPmNodeId target, const bool
setAddress, const u64 address, const enum XPmRequestAck ack);
```

Parameters

The following table lists the `XPm_RequestWakeUp` function arguments.

Table 8: **XPm_RequestWakeUp Arguments**

Type	Name	Description
const enum XPmNodeId	target	Node ID of the CPU or PU to be powered/woken up.
const bool	setAddress	Specifies whether the start address argument is being passed. • 0 : do not set start address • 1 : set start address
const u64	address	Address from which to resume when woken up. Will only be used if set_address is 1.
const enum XPmRequestAck	ack	Requested acknowledge type

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ForcePowerDown

One PU can request a forced poweroff of another PU or its power island or power domain. This can be used for killing an unresponsive PU, in which case all resources of that PU will be automatically released.

***Note*:** Force power down may not be requested by a PU for itself.

Prototype

```
XStatus XPm_ForcePowerDown(const enum XPmNodeId target, const enum
XPmRequestAck ack);
```

Parameters

The following table lists the `XPm_ForcePowerDown` function arguments.

Table 9: **XPm_ForcePowerDown Arguments**

Type	Name	Description
const enum `XPmNodeId`	target	Node ID of the PU node or power island/domain to be powered down.
const enum `XPmRequestAck`	ack	Requested acknowledge type

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_AbortSuspend

This function is called by a CPU after a XPm_SelfSuspend call to notify the power management controller that CPU has aborted suspend or in response to an init suspend request when the PU refuses to suspend.

***Note*:** Calling PU expects the PMU to abort the initiated suspend procedure. This is a non-blocking call without any acknowledge.

Prototype

```
XStatus XPm_AbortSuspend(const enum XPmAbortReason reason);
```

Parameters

The following table lists the `XPm_AbortSuspend` function arguments.

Table 10: **XPm_AbortSuspend Arguments**

Type	Name	Description
const enum `XPmAbortReason`	reason	Reason code why the suspend can not be performed or completed • ABORT_REASON_WKUP_EVENT : local wakeup-event received • ABORT_REASON_PU_BUSY : PU is busy • ABORT_REASON_NO_PWRDN : no external powerdown supported • ABORT_REASON_UNKNOWN : unknown error during suspend procedure

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_SetWakeUpSource

This function is called by a PU to add or remove a wake-up source prior to going to suspend. The list of wake sources for a PU is automatically cleared whenever the PU is woken up or when one of its CPUs aborts the suspend procedure.

Note: Declaring a node as a wakeup source will ensure that the node will not be powered off. It also will cause the PMU to configure the GIC Proxy accordingly if the FPD is powered off.

Prototype

```
XStatus XPm_SetWakeUpSource(const enum XPmNodeId target, const enum
XPmNodeId wkup_node, const u8 enable);
```

Parameters

The following table lists the `XPm_SetWakeUpSource` function arguments.

Table 11: **XPm_SetWakeUpSource Arguments**

Type	Name	Description
const enum `XPmNodeId`	target	Node ID of the target to be woken up.
const enum `XPmNodeId`	wkup_node	Node ID of the wakeup device.
const u8	enable	Enable flag: • 1 : the wakeup source is added to the list • 0 : the wakeup source is removed from the list

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_SystemShutdown

This function can be used by a privileged PU to shut down or restart the complete device.

Note: In either case the PMU will call XPm_InitSuspendCb for each of the other PUs, allowing them to gracefully shut down. If a PU is asleep it will be woken up by the PMU. The PU making the XPm_SystemShutdown should perform its own suspend procedure after calling this API. It will not receive an init suspend callback.

Prototype

```
XStatus XPm_SystemShutdown(u32 type, u32 subtype);
```

Parameters

The following table lists the `XPm_SystemShutdown` function arguments.

Table 12: **XPm_SystemShutdown Arguments**

Type	Name	Description
Commented parameter restart does not exist in function XPm_SystemShutdown.	restart	Should the system be restarted automatically? • PM_SHUTDOWN : no restart requested, system will be powered off permanently • PM_RESTART : restart is requested, system will go through a full reset

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_RequestNode

Used to request the usage of a PM-slave. Using this API call a PU requests access to a slave device and asserts its requirements on that device. Provided the PU is sufficiently privileged, the PMU will enable access to the memory mapped region containing the control registers of that device. For devices that can only be serving a single PU, any other privileged PU will now be blocked from accessing this device until the node is released.

Note: None

Prototype

```
XStatus XPm_RequestNode(const enum XPmNodeId node, const u32 capabilities,
const u32 qos, const enum XPmRequestAck ack);
```

Parameters

The following table lists the `XPm_RequestNode` function arguments.

Table 13: **XPm_RequestNode Arguments**

Type	Name	Description
const enum `XPmNodeId`	node	Node ID of the PM slave requested
const u32	capabilities	Slave-specific capabilities required, can be combined • PM_CAP_ACCESS : full access / functionality • PM_CAP_CONTEXT : preserve context • PM_CAP_WAKEUP : emit wake interrupts
const u32	qos	Quality of Service (0-100) required
const enum `XPmRequestAck`	ack	Requested acknowledge type

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_SetRequirement

This function is used by a PU to announce a change in requirements for a specific slave node which is currently in use.

Note: If this function is called after the last awake CPU within the PU calls SelfSuspend, the requirement change shall be performed after the CPU signals the end of suspend to the power management controller, (e.g. WFI interrupt).

Prototype

```
XStatus XPm_SetRequirement(const enum XPmNodeId nid, const u32
capabilities, const u32 qos, const enum XPmRequestAck ack);
```

Parameters

The following table lists the `XPm_SetRequirement` function arguments.

Table 14: **XPm_SetRequirement Arguments**

Type	Name	Description
const enum `XPmNodeId`	nid	Node ID of the PM slave.
const u32	capabilities	Slave-specific capabilities required.
const u32	qos	Quality of Service (0-100) required.
const enum `XPmRequestAck`	ack	Requested acknowledge type

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ReleaseNode

This function is used by a PU to release the usage of a PM slave. This will tell the power management controller that the node is no longer needed by that PU, potentially allowing the node to be placed into an inactive state.

Note: None

Prototype

```
XStatus XPm_ReleaseNode(const enum XPmNodeId node);
```

Parameters

The following table lists the `XPm_ReleaseNode` function arguments.

Table 15: **XPm_ReleaseNode Arguments**

Type	Name	Description
const enum `XPmNodeId`	node	Node ID of the PM slave.

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_SetMaxLatency

This function is used by a PU to announce a change in the maximum wake-up latency requirements for a specific slave node currently used by that PU.

Note: Setting maximum wake-up latency can constrain the set of possible power states a resource can be put into.

Prototype

```
XStatus XPm_SetMaxLatency(const enum XPmNodeId node, const u32 latency);
```

Parameters

The following table lists the `XPm_SetMaxLatency` function arguments.

Table 16: **XPm_SetMaxLatency Arguments**

Type	Name	Description
const enum `XPmNodeId`	node	Node ID of the PM slave.
const u32	latency	Maximum wake-up latency required.

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_InitSuspendCb

Callback function to be implemented in each PU, allowing the power management controller to request that the PU suspend itself.

Note: If the PU fails to act on this request the power management controller or the requesting PU may choose to employ the forceful power down option.

Prototype

```
void XPm_InitSuspendCb(const enum XPmSuspendReason reason, const u32
latency, const u32 state, const u32 timeout);
```

Parameters

The following table lists the `XPm_InitSuspendCb` function arguments.

Table 17: **XPm_InitSuspendCb Arguments**

Type	Name	Description
const enum `XPmSuspendReason`	reason	Suspend reason: • SUSPEND_REASON_PU_REQ : Request by another PU • SUSPEND_REASON_ALERT : Unrecoverable SysMon alert • SUSPEND_REASON_SHUTDOWN : System shutdown • SUSPEND_REASON_RESTART : System restart

Table 17: **XPm_InitSuspendCb Arguments** *(cont'd)*

Type	Name	Description
const u32	latency	Maximum wake-up latency in us(micro secs). This information can be used by the PU to decide what level of context saving may be required.
const u32	state	Targeted sleep/suspend state.
const u32	timeout	Timeout in ms, specifying how much time a PU has to initiate its suspend procedure before it's being considered unresponsive.

Returns

None

XPm_AcknowledgeCb

This function is called by the power management controller in response to any request where an acknowledge callback was requested, i.e. where the 'ack' argument passed by the PU was REQUEST_ACK_NON_BLOCKING.

***Note*:** None

Prototype

```
void XPm_AcknowledgeCb(const enum XPmNodeId node, const XStatus status,
const u32 oppoint);
```

Parameters

The following table lists the `XPm_AcknowledgeCb` function arguments.

Table 18: **XPm_AcknowledgeCb Arguments**

Type	Name	Description
const enum `XPmNodeId`	node	ID of the component or sub-system in question.
const XStatus	status	Status of the operation: • OK: the operation completed successfully • ERR: the requested operation failed
const u32	oppoint	Operating point of the node in question

Returns

None

XPm_NotifyCb

This function is called by the power management controller if an event the PU was registered for has occurred. It will populate the notifier data structure passed when calling XPm_RegisterNotifier.

Note: None

Prototype

```
void XPm_NotifyCb(const enum XPmNodeId node, const enum XPmNotifyEvent
event, const u32 oppoint);
```

Parameters

The following table lists the `XPm_NotifyCb` function arguments.

Table 19: **XPm_NotifyCb Arguments**

Type	Name	Description
const enum `XPmNodeId`	node	ID of the node the event notification is related to.
const enum `XPmNotifyEvent`	event	ID of the event
const u32	oppoint	Current operating state of the node.

Returns

None

XPm_GetApiVersion

This function is used to request the version number of the API running on the power management controller.

Note: None

Prototype

```
XStatus XPm_GetApiVersion(u32 *version);
```

Parameters

The following table lists the `XPm_GetApiVersion` function arguments.

Table 20: **XPm_GetApiVersion Arguments**

Type	Name	Description
u32 *	version	Returns the API 32-bit version number. Returns 0 if no PM firmware present.

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_GetNodeStatus

This function is used to obtain information about the current state of a component. The caller must pass a pointer to an `XPm_NodeStatus` structure, which must be pre-allocated by the caller.

- status - The current power state of the requested node.
  - For CPU nodes:
    - 0 : if CPU is off (powered down),
    - 1 : if CPU is active (powered up),
    - 2 : if CPU is in sleep (powered down),
    - 3 : if CPU is suspending (powered up)
  - For power islands and power domains:
    - 0 : if island is powered down,
    - 1 : if island is powered up
  - For PM slaves:
    - 0 : if slave is powered down,
    - 1 : if slave is powered up,
    - 2 : if slave is in retention
- requirement - Slave nodes only: Returns current requirements the requesting PU has requested of the node.
- usage - Slave nodes only: Returns current usage status of the node:
  - 0 : node is not used by any PU,
  - 1 : node is used by caller exclusively,
  - 2 : node is used by other PU(s) only,
  - 3 : node is used by caller and by other PU(s)

Note: None

Prototype

```
XStatus XPm_GetNodeStatus(const enum XPmNodeId node, XPm_NodeStatus *const
nodestatus);
```

Parameters

The following table lists the `XPm_GetNodeStatus` function arguments.

Table 21: **XPm_GetNodeStatus Arguments**

Type	Name	Description
const enum `XPmNodeId`	node	ID of the component or sub-system in question.
`XPm_NodeStatus` *const	nodestatus	Used to return the complete status of the node.

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_GetOpCharacteristic

Call this function to request the power management controller to return information about an operating characteristic of a component.

Note: Power value is not actual power consumption of device. It is default dummy power value which is fixed in PMUFW. Temperature type is not supported for ZynqMP.

Prototype

```
XStatus XPm_GetOpCharacteristic(const enum XPmNodeId node, const enum
XPmOpCharType type, u32 *const result);
```

Parameters

The following table lists the `XPm_GetOpCharacteristic` function arguments.

Table 22: **XPm_GetOpCharacteristic Arguments**

Type	Name	Description
const enum `XPmNodeId`	node	ID of the component or sub-system in question.

Table 22: **XPm_GetOpCharacteristic Arguments** *(cont'd)*

Type	Name	Description
const enum `XPmOpCharType`	type	Type of operating characteristic requested: • power (current power consumption), • latency (current latency in micro seconds to return to active state), • temperature (current temperature),
u32 *const	result	Used to return the requested operating characteristic.

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ResetAssert

This function is used to assert or release reset for a particular reset line. Alternatively a reset pulse can be requested as well.

Note: None

Prototype

```
XStatus XPm_ResetAssert(const enum XPmReset reset, const enum
XPmResetAction resetaction);
```

Parameters

The following table lists the `XPm_ResetAssert` function arguments.

Table 23: **XPm_ResetAssert Arguments**

Type	Name	Description
const enum `XPmReset`	reset	ID of the reset line
Commented parameter assert does not exist in function XPm_ResetAssert.	assert	Identifies action: • PM_RESET_ACTION_RELEASE : release reset, • PM_RESET_ACTION_ASSERT : assert reset, • PM_RESET_ACTION_PULSE : pulse reset,

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ResetGetStatus

Call this function to get the current status of the selected reset line.

***Note*:** None

Prototype

```
XStatus XPm_ResetGetStatus(const enum XPmReset reset, u32 *status);
```

Parameters

The following table lists the `XPm_ResetGetStatus` function arguments.

Table 24: **XPm_ResetGetStatus Arguments**

Type	Name	Description
const enum `XPmReset`	reset	Reset line
u32 *	status	Status of specified reset (true - asserted, false - released)

Returns

Returns 1/XST_FAILURE for 'asserted' or 0/XST_SUCCESS for 'released'.

XPm_RegisterNotifier

A PU can call this function to request that the power management controller call its notify callback whenever a qualifying event occurs. One can request to be notified for a specific or any event related to a specific node.

- nodeID : ID of the node to be notified about,
- eventID : ID of the event in question, '-1' denotes all events (- EVENT_STATE_CHANGE, EVENT_ZERO_USERS),
- wake : true: wake up on event, false: do not wake up (only notify if awake), no buffering/ queueing
- callback : Pointer to the custom callback function to be called when the notification is available. The callback executes from interrupt context, so the user must take special care when implementing the callback. Callback is optional, may be set to NULL.
- received : Variable indicating how many times the notification has been received since the notifier is registered.

***Note*:** The caller shall initialize the notifier object before invoking the XPm_RegisteredNotifier function. While notifier is registered, the notifier object shall not be modified by the caller.

Prototype

```
XStatus XPm_RegisterNotifier(XPm_Notifier *const notifier);
```

Parameters

The following table lists the `XPm_RegisterNotifier` function arguments.

Table 25: **XPm_RegisterNotifier Arguments**

Type	Name	Description
`XPm_Notifier` *const	notifier	Pointer to the notifier object to be associated with the requested notification. The notifier object contains the following data related to the notification:

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_UnregisterNotifier

A PU calls this function to unregister for the previously requested notifications.

Note: None

Prototype

```
XStatus XPm_UnregisterNotifier(XPm_Notifier *const notifier);
```

Parameters

The following table lists the `XPm_UnregisterNotifier` function arguments.

Table 26: **XPm_UnregisterNotifier Arguments**

Type	Name	Description
`XPm_Notifier` *const	notifier	Pointer to the notifier object associated with the previously requested notification

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_MmioWrite

Call this function to write a value directly into a register that isn't accessible directly, such as registers in the clock control unit. This call is bypassing the power management logic. The permitted addresses are subject to restrictions as defined in the PCW configuration.

Note: If the access isn't permitted this function returns an error code.

Prototype

```
XStatus XPm_MmioWrite(const u32 address, const u32 mask, const u32 value);
```

Parameters

The following table lists the `XPm_MmioWrite` function arguments.

Table 27: **XPm_MmioWrite Arguments**

Type	Name	Description
const u32	address	Physical 32-bit address of memory mapped register to write to.
const u32	mask	32-bit value used to limit write to specific bits in the register.
const u32	value	Value to write to the register bits specified by the mask.

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_MmioRead

Call this function to read a value from a register that isn't accessible directly. The permitted addresses are subject to restrictions as defined in the PCW configuration.

Note: If the access isn't permitted this function returns an error code.

Prototype

```
XStatus XPm_MmioRead(const u32 address, u32 *const value);
```

Parameters

The following table lists the `XPm_MmioRead` function arguments.

Table 28: **XPm_MmioRead Arguments**

Type	Name	Description
const u32	address	Physical 32-bit address of memory mapped register to read from.

Table 28: **XPm_MmioRead Arguments** *(cont'd)*

Type	Name	Description
u32 *const	value	Returns the 32-bit value read from the register

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ClockEnable

Call this function to enable (activate) a clock.

***Note*:** If the access isn't permitted this function returns an error code.

Prototype

```
XStatus XPm_ClockEnable(const enum XPmClock clk);
```

Parameters

The following table lists the `XPm_ClockEnable` function arguments.

Table 29: **XPm_ClockEnable Arguments**

Type	Name	Description
const enum `XPmClock`	clk	Identifier of the target clock to be enabled

Returns

Status of performing the operation as returned by the PMU-FW

XPm_ClockDisable

Call this function to disable (gate) a clock.

***Note*:** If the access isn't permitted this function returns an error code.

Prototype

```
XStatus XPm_ClockDisable(const enum XPmClock clk);
```

Parameters

The following table lists the `XPm_ClockDisable` function arguments.

Table 30: **XPm_ClockDisable Arguments**

Type	Name	Description
const enum XPmClock	clk	Identifier of the target clock to be disabled

Returns

Status of performing the operation as returned by the PMU-FW

XPm_ClockGetStatus

Call this function to get status of a clock gate state.

Prototype

```
XStatus XPm_ClockGetStatus(const enum XPmClock clk, u32 *const status);
```

Parameters

The following table lists the `XPm_ClockGetStatus` function arguments.

Table 31: **XPm_ClockGetStatus Arguments**

Type	Name	Description
const enum XPmClock	clk	Identifier of the target clock
u32 *const	status	Location to store clock gate state (1=enabled, 0=disabled)

Returns

Status of performing the operation as returned by the PMU-FW

XPm_ClockSetOneDivider

Call this function to set divider for a clock.

***Note*:** If the access isn't permitted this function returns an error code.

Prototype

```
XStatus XPm_ClockSetOneDivider(const enum XPmClock clk, const u32 divider,
const u32 divId);
```

Parameters

The following table lists the `XPm_ClockSetOneDivider` function arguments.

Table 32: **XPm_ClockSetOneDivider Arguments**

Type	Name	Description
const enum XPmClock	clk	Identifier of the target clock
const u32	divider	Divider value to be set
const u32	divId	ID of the divider to be set

Returns

Status of performing the operation as returned by the PMU-FW

XPm_ClockSetDivider

Call this function to set divider for a clock.

Note: If the access isn't permitted this function returns an error code.

Prototype

```
XStatus XPm_ClockSetDivider(const enum XPmClock clk, const u32 divider);
```

Parameters

The following table lists the `XPm_ClockSetDivider` function arguments.

Table 33: **XPm_ClockSetDivider Arguments**

Type	Name	Description
const enum XPmClock	clk	Identifier of the target clock
const u32	divider	Divider value to be set

Returns

XST_INVALID_PARAM or status of performing the operation as returned by the PMU-FW

XPm_ClockGetOneDivider

Local function to get one divider (DIV0 or DIV1) of a clock.

Prototype

```
XStatus XPm_ClockGetOneDivider(const enum XPmClock clk, u32 *const divider,
const u32 divId);
```

Parameters

The following table lists the `XPm_ClockGetOneDivider` function arguments.

Table 34: **XPm_ClockGetOneDivider Arguments**

Type	Name	Description
const enum `XPmClock`	clk	Identifier of the target clock
u32 *const	divider	Location to store the divider value

Returns

Status of performing the operation as returned by the PMU-FW

XPm_ClockGetDivider

Call this function to get divider of a clock.

Prototype

```
XStatus XPm_ClockGetDivider(const enum XPmClock clk, u32 *const divider);
```

Parameters

The following table lists the `XPm_ClockGetDivider` function arguments.

Table 35: **XPm_ClockGetDivider Arguments**

Type	Name	Description
const enum `XPmClock`	clk	Identifier of the target clock
u32 *const	divider	Location to store the divider value

Returns

XST_INVALID_PARAM or status of performing the operation as returned by the PMU-FW

XPm_ClockSetParent

Call this function to set parent for a clock.

***Note*:** If the access isn't permitted this function returns an error code.

Prototype

```
XStatus XPm_ClockSetParent(const enum XPmClock clk, const enum XPmClock
parent);
```

Parameters

The following table lists the `XPm_ClockSetParent` function arguments.

Table 36: **XPm_ClockSetParent Arguments**

Type	Name	Description
const enum `XPmClock`	clk	Identifier of the target clock
const enum `XPmClock`	parent	Identifier of the target parent clock

Returns

XST_INVALID_PARAM or status of performing the operation as returned by the PMU-FW.

XPm_ClockGetParent

Call this function to get parent of a clock.

Prototype

```
XStatus XPm_ClockGetParent(const enum XPmClock clk, enum XPmClock *const
parent);
```

Parameters

The following table lists the `XPm_ClockGetParent` function arguments.

Table 37: **XPm_ClockGetParent Arguments**

Type	Name	Description
const enum `XPmClock`	clk	Identifier of the target clock
enum `XPmClock` *const	parent	Location to store clock parent ID

Returns

XST_INVALID_PARAM or status of performing the operation as returned by the PMU-FW.

XPm_ClockSetRate

Call this function to set rate of a clock.

***Note*:** If the action isn't permitted this function returns an error code.

Prototype

```
XStatus XPm_ClockSetRate(const enum XPmClock clk, const u32 rate);
```

Parameters

The following table lists the `XPm_ClockSetRate` function arguments.

Table 38: **XPm_ClockSetRate Arguments**

Type	Name	Description
const enum XPmClock	clk	Identifier of the target clock
const u32	rate	Clock frequency (rate) to be set

Returns

Status of performing the operation as returned by the PMU-FW

XPm_ClockGetRate

Call this function to get rate of a clock.

Prototype

```
XStatus XPm_ClockGetRate(const enum XPmClock clk, u32 *const rate);
```

Parameters

The following table lists the `XPm_ClockGetRate` function arguments.

Table 39: **XPm_ClockGetRate Arguments**

Type	Name	Description
const enum XPmClock	clk	Identifier of the target clock
u32 *const	rate	Location where the rate should be stored

Returns

Status of performing the operation as returned by the PMU-FW

XPm_PllSetParameter

Call this function to set a PLL parameter.

Note: If the access isn't permitted this function returns an error code.

Prototype

```
XStatus XPm_PllSetParameter(const enum XPmNodeId node, const enum
XPmPllParam parameter, const u32 value);
```

Parameters

The following table lists the `XPm_PllSetParameter` function arguments.

Table 40: **XPm_PllSetParameter Arguments**

Type	Name	Description
const enum XPmNodeId	node	PLL node identifier
const enum XPmPllParam	parameter	PLL parameter identifier
const u32	value	Value of the PLL parameter

Returns

Status of performing the operation as returned by the PMU-FW

XPm_PllGetParameter

Call this function to get a PLL parameter.

Prototype

```
XStatus XPm_PllGetParameter(const enum XPmNodeId node, const enum
XPmPllParam parameter, u32 *const value);
```

Parameters

The following table lists the `XPm_PllGetParameter` function arguments.

Table 41: **XPm_PllGetParameter Arguments**

Type	Name	Description
const enum XPmNodeId	node	PLL node identifier
const enum XPmPllParam	parameter	PLL parameter identifier
u32 *const	value	Location to store value of the PLL parameter

Returns

Status of performing the operation as returned by the PMU-FW

XPm_PllSetMode

Call this function to set a PLL mode.

***Note*:** If the access isn't permitted this function returns an error code.

Prototype

```
XStatus XPm_PllSetMode(const enum XPmNodeId node, const enum XPmPllMode
mode);
```

Parameters

The following table lists the `XPm_PllSetMode` function arguments.

Table 42: **XPm_PllSetMode Arguments**

Type	Name	Description
const enum `XPmNodeId`	node	PLL node identifier
const enum XPmPllMode	mode	PLL mode to be set

Returns

Status of performing the operation as returned by the PMU-FW

XPm_PllGetMode

Call this function to get a PLL mode.

Prototype

```
XStatus XPm_PllGetMode(const enum XPmNodeId node, enum XPmPllMode *const
mode);
```

Parameters

The following table lists the `XPm_PllGetMode` function arguments.

Table 43: **XPm_PllGetMode Arguments**

Type	Name	Description
const enum `XPmNodeId`	node	PLL node identifier
enum XPmPllMode *const	mode	Location to store the PLL mode

Returns

Status of performing the operation as returned by the PMU-FW

XPm_PinCtrlAction

Locally used function to request or release a pin control.

Prototype

```
XStatus XPm_PinCtrlAction(const u32 pin, const enum XPmApiId api);
```

Parameters

The following table lists the `XPm_PinCtrlAction` function arguments.

Table 44: **XPm_PinCtrlAction Arguments**

Type	Name	Description
const u32	pin	PIN identifier (index from range 0-77) API identifier (request or release pin control)

Returns

Status of performing the operation as returned by the PMU-FW

XPm_PinCtrlRequest

Call this function to request a pin control.

Prototype

```
XStatus XPm_PinCtrlRequest(const u32 pin);
```

Parameters

The following table lists the `XPm_PinCtrlRequest` function arguments.

Table 45: **XPm_PinCtrlRequest Arguments**

Type	Name	Description
const u32	pin	PIN identifier (index from range 0-77)

Returns

Status of performing the operation as returned by the PMU-FW

XPm_PinCtrlRelease

Call this function to release a pin control.

Prototype

```
XStatus XPm_PinCtrlRelease(const u32 pin);
```

Parameters

The following table lists the `XPm_PinCtrlRelease` function arguments.

Table 46: **XPm_PinCtrlRelease Arguments**

Type	Name	Description
const u32	pin	PIN identifier (index from range 0-77)

Returns

Status of performing the operation as returned by the PMU-FW

XPm_PinCtrlSetFunction

Call this function to set a pin function.

Note: If the access isn't permitted this function returns an error code.

Prototype

```
XStatus XPm_PinCtrlSetFunction(const u32 pin, const enum XPmPinFn fn);
```

Parameters

The following table lists the `XPm_PinCtrlSetFunction` function arguments.

Table 47: **XPm_PinCtrlSetFunction Arguments**

Type	Name	Description
const u32	pin	Pin identifier
const enum XPmPinFn	fn	Pin function to be set

Returns

Status of performing the operation as returned by the PMU-FW

XPm_PinCtrlGetFunction

Call this function to get currently configured pin function.

Prototype

```
XStatus XPm_PinCtrlGetFunction(const u32 pin, enum XPmPinFn *const fn);
```

Parameters

The following table lists the `XPm_PinCtrlGetFunction` function arguments.

Table 48: **XPm_PinCtrlGetFunction Arguments**

Type	Name	Description
const u32	pin	PLL node identifier
enum XPmPinFn *const	fn	Location to store the pin function

Returns

Status of performing the operation as returned by the PMU-FW

XPm_PinCtrlSetParameter

Call this function to set a pin parameter.

Note: If the access isn't permitted this function returns an error code.

Prototype

```
XStatus XPm_PinCtrlSetParameter(const u32 pin, const enum XPmPinParam
param, const u32 value);
```

Parameters

The following table lists the `XPm_PinCtrlSetParameter` function arguments.

Table 49: **XPm_PinCtrlSetParameter Arguments**

Type	Name	Description
const u32	pin	Pin identifier
const enum XPmPinParam	param	Pin parameter identifier
const u32	value	Value of the pin parameter to set

Returns

Status of performing the operation as returned by the PMU-FW

XPm_PinCtrlGetParameter

Call this function to get currently configured value of pin parameter.

Prototype

```
XStatus XPm_PinCtrlGetParameter(const u32 pin, const enum XPmPinParam
param, u32 *const value);
```

Parameters

The following table lists the `XPm_PinCtrlGetParameter` function arguments.

Table 50: **XPm_PinCtrlGetParameter Arguments**

Type	Name	Description
const u32	pin	Pin identifier
const enum XPmPinParam	param	Pin parameter identifier
u32 *const	value	Location to store value of the pin parameter

Returns

Status of performing the operation as returned by the PMU-FW

Enumerations

Enumeration XPmApiId

APIs for Miscellaneous functions, suspending of PUs, managing PM slaves and Direct control.

Table 51: **Enumeration XPmApiId Values**

Value	Description
PM_GET_API_VERSION	
PM_SET_CONFIGURATION	
PM_GET_NODE_STATUS	
PM_GET_OP_CHARACTERISTIC	
PM_REGISTER_NOTIFIER	
PM_REQUEST_SUSPEND	
PM_SELF_SUSPEND	
PM_FORCE_POWERDOWN	
PM_ABORT_SUSPEND	
PM_REQUEST_WAKEUP	
PM_SET_WAKEUP_SOURCE	
PM_SYSTEM_SHUTDOWN	

Table 51: **Enumeration XPmApiId Values** *(cont'd)*

Value	Description
PM_REQUEST_NODE	
PM_RELEASE_NODE	
PM_SET_REQUIREMENT	
PM_SET_MAX_LATENCY	
PM_RESET_ASSERT	
PM_RESET_GET_STATUS	
PM_MMIO_WRITE	
PM_MMIO_READ	
PM_INIT_FINALIZE	
PM_FPGA_LOAD	
PM_FPGA_GET_STATUS	
PM_GET_CHIPID	
PM_SECURE_SHA	
PM_SECURE_RSA	
PM_PINCTRL_REQUEST	
PM_PINCTRL_RELEASE	
PM_PINCTRL_GET_FUNCTION	
PM_PINCTRL_SET_FUNCTION	
PM_PINCTRL_CONFIG_PARAM_GET	
PM_PINCTRL_CONFIG_PARAM_SET	
PM_IOCTL	
PM_QUERY_DATA	
PM_CLOCK_ENABLE	
PM_CLOCK_DISABLE	
PM_CLOCK_GETSTATE	
PM_CLOCK_SETDIVIDER	
PM_CLOCK_GETDIVIDER	
PM_CLOCK_SETRATE	
PM_CLOCK_GETRATE	
PM_CLOCK_SETPARENT	
PM_CLOCK_GETPARENT	
PM_SECURE_IMAGE	
PM_FPGA_READ	
PM_SECURE_AES	
PM_PLL_SET_PARAMETER	
PM_PLL_GET_PARAMETER	
PM_PLL_SET_MODE	
PM_PLL_GET_MODE	
PM_REGISTER_ACCESS	

Table 51: **Enumeration XPmApiId Values** *(cont'd)*

Value	Description
PM_EFUSE_ACCESS	
PM_API_MAX	

Enumeration XPmApiCbId

PM API Callback Id Enum

Table 52: **Enumeration XPmApiCbId Values**

Value	Description
PM_INIT_SUSPEND_CB	
PM_ACKNOWLEDGE_CB	
PM_NOTIFY_CB	
PM_NOTIFY_STL_NO_OP	

Enumeration XPmNodeId

PM Node ID Enum

Table 53: **Enumeration XPmNodeId Values**

Value	Description
NODE_UNKNOWN	
NODE_APU	
NODE_APU_0	
NODE_APU_1	
NODE_APU_2	
NODE_APU_3	
NODE_RPU	
NODE_RPU_0	
NODE_RPU_1	
NODE_PLD	
NODE_FPD	
NODE_OCM_BANK_0	
NODE_OCM_BANK_1	
NODE_OCM_BANK_2	
NODE_OCM_BANK_3	
NODE_TCM_0_A	
NODE_TCM_0_B	

Table 53: **Enumeration XPmNodeId Values** *(cont'd)*

Value	Description
NODE_TCM_1_A	
NODE_TCM_1_B	
NODE_L2	
NODE_GPU_PP_0	
NODE_GPU_PP_1	
NODE_USB_0	
NODE_USB_1	
NODE_TTC_0	
NODE_TTC_1	
NODE_TTC_2	
NODE_TTC_3	
NODE_SATA	
NODE_ETH_0	
NODE_ETH_1	
NODE_ETH_2	
NODE_ETH_3	
NODE_UART_0	
NODE_UART_1	
NODE_SPI_0	
NODE_SPI_1	
NODE_I2C_0	
NODE_I2C_1	
NODE_SD_0	
NODE_SD_1	
NODE_DP	
NODE_GDMA	
NODE_ADMA	
NODE_NAND	
NODE_QSPI	
NODE_GPIO	
NODE_CAN_0	
NODE_CAN_1	
NODE_EXTERN	
NODE_APLL	
NODE_VPLL	
NODE_DPLL	
NODE_RPLL	
NODE_IOPLL	
NODE_DDR	

Table 53: **Enumeration XPmNodeId Values** *(cont'd)*

Value	Description
NODE_IPI_APU	
NODE_IPI_RPU_0	
NODE_GPU	
NODE_PCIE	
NODE_PCAP	
NODE_RTC	
NODE_LPD	
NODE_VCU	
NODE_IPI_RPU_1	
NODE_IPI_PL_0	
NODE_IPI_PL_1	
NODE_IPI_PL_2	
NODE_IPI_PL_3	
NODE_PL	
NODE_ID_MAX	

Enumeration XPmRequestAck

PM Acknowledge Request Types

Table 54: **Enumeration XPmRequestAck Values**

Value	Description
REQUEST_ACK_NO	
REQUEST_ACK_BLOCKING	
REQUEST_ACK_NON_BLOCKING	
REQUEST_ACK_CB_CERROR	

Enumeration XPmAbortReason

PM Abort Reasons Enum

Table 55: **Enumeration XPmAbortReason Values**

Value	Description
ABORT_REASON_WKUP_EVENT	
ABORT_REASON_PU_BUSY	
ABORT_REASON_NO_PWRDN	
ABORT_REASON_UNKNOWN	

Table 55: **Enumeration XPmAbortReason Values** *(cont'd)*

Value	Description
ABORT_REASON_WKUP_EVENT	
ABORT_REASON_PU_BUSY	
ABORT_REASON_NO_PWRDN	
ABORT_REASON_UNKNOWN	

Enumeration XPmSuspendReason

PM Suspend Reasons Enum

Table 56: **Enumeration XPmSuspendReason Values**

Value	Description
SUSPEND_REASON_PU_REQ	
SUSPEND_REASON_ALERT	
SUSPEND_REASON_SYS_SHUTDOWN	
SUSPEND_REASON_PU_REQ	
SUSPEND_REASON_ALERT	
SUSPEND_REASON_SYS_SHUTDOWN	

Enumeration XPmRamState

PM RAM States Enum

Table 57: **Enumeration XPmRamState Values**

Value	Description
PM_RAM_STATE_OFF	
PM_RAM_STATE_RETENTION	
PM_RAM_STATE_ON	

Enumeration XPmOpCharType

PM Operating Characteristic types Enum

Table 58: **Enumeration XPmOpCharType Values**

Value	Description
PM_OPCHAR_TYPE_POWER	
PM_OPCHAR_TYPE_TEMP	
PM_OPCHAR_TYPE_LATENCY	

Table 58: **Enumeration XPmOpCharType Values** *(cont'd)*

Value	Description
PM_OPCHAR_TYPE_POWER	
PM_OPCHAR_TYPE_TEMP	
PM_OPCHAR_TYPE_LATENCY	

Enumeration XPmBootStatus

Boot Status Enum

Table 59: **Enumeration XPmBootStatus Values**

Value	Description
PM_INITIAL_BOOT	boot is a fresh system startup
PM_RESUME	boot is a resume
PM_BOOT_ERROR	error, boot cause cannot be identified
PM_INITIAL_BOOT	
PM_RESUME	
PM_BOOT_ERROR	

Enumeration XPmResetAction

PM Reset Action types

Table 60: **Enumeration XPmResetAction Values**

Value	Description
XILPM_RESET_ACTION_RELEASE	
XILPM_RESET_ACTION_ASSERT	
XILPM_RESET_ACTION_PULSE	

Enumeration XPmReset

PM Reset Line IDs

Table 61: **Enumeration XPmReset Values**

Value	Description
XILPM_RESET_PCIE_CFG	
XILPM_RESET_PCIE_BRIDGE	
XILPM_RESET_PCIE_CTRL	
XILPM_RESET_DP	

Table 61: **Enumeration XPmReset Values** *(cont'd)*

Value	Description
XILPM_RESET_SWDT_CRF	
XILPM_RESET_AFI_FM5	
XILPM_RESET_AFI_FM4	
XILPM_RESET_AFI_FM3	
XILPM_RESET_AFI_FM2	
XILPM_RESET_AFI_FM1	
XILPM_RESET_AFI_FM0	
XILPM_RESET_GDMA	
XILPM_RESET_GPU_PP1	
XILPM_RESET_GPU_PP0	
XILPM_RESET_GPU	
XILPM_RESET_GT	
XILPM_RESET_SATA	
XILPM_RESET_ACPU3_PWRON	
XILPM_RESET_ACPU2_PWRON	
XILPM_RESET_ACPU1_PWRON	
XILPM_RESET_ACPU0_PWRON	
XILPM_RESET_APU_L2	
XILPM_RESET_ACPU3	
XILPM_RESET_ACPU2	
XILPM_RESET_ACPU1	
XILPM_RESET_ACPU0	
XILPM_RESET_DDR	
XILPM_RESET_APM_FPD	
XILPM_RESET_SOFT	
XILPM_RESET_GEM0	
XILPM_RESET_GEM1	
XILPM_RESET_GEM2	
XILPM_RESET_GEM3	
XILPM_RESET_QSPI	
XILPM_RESET_UART0	
XILPM_RESET_UART1	
XILPM_RESET_SPI0	
XILPM_RESET_SPI1	
XILPM_RESET_SDIO0	
XILPM_RESET_SDIO1	
XILPM_RESET_CAN0	
XILPM_RESET_CAN1	
XILPM_RESET_I2C0	

Table 61: **Enumeration XPmReset Values** *(cont'd)*

Value	Description
XILPM_RESET_I2C1	
XILPM_RESET_TTC0	
XILPM_RESET_TTC1	
XILPM_RESET_TTC2	
XILPM_RESET_TTC3	
XILPM_RESET_SWDT_CRL	
XILPM_RESET_NAND	
XILPM_RESET_ADMA	
XILPM_RESET_GPIO	
XILPM_RESET_IOU_CC	
XILPM_RESET_TIMESTAMP	
XILPM_RESET_RPU_R50	
XILPM_RESET_RPU_R51	
XILPM_RESET_RPU_AMBA	
XILPM_RESET_OCM	
XILPM_RESET_RPU_PGE	
XILPM_RESET_USB0_CORERESET	
XILPM_RESET_USB1_CORERESET	
XILPM_RESET_USB0_HIBERRESET	
XILPM_RESET_USB1_HIBERRESET	
XILPM_RESET_USB0_APB	
XILPM_RESET_USB1_APB	
XILPM_RESET_IPI	
XILPM_RESET_APM_LPD	
XILPM_RESET_RTC	
XILPM_RESET_SYSMON	
XILPM_RESET_AFI_FM6	
XILPM_RESET_LPD_SWDT	
XILPM_RESET_FPD	
XILPM_RESET_RPU_DBG1	
XILPM_RESET_RPU_DBG0	
XILPM_RESET_DBG_LPD	
XILPM_RESET_DBG_FPD	
XILPM_RESET_APLL	
XILPM_RESET_DPLL	
XILPM_RESET_VPLL	
XILPM_RESET_IOPLL	
XILPM_RESET_RPLL	
XILPM_RESET_GPO3_PL_0	

Table 61: **Enumeration XPmReset Values** *(cont'd)*

Value	Description
XILPM_RESET_GPO3_PL_1	
XILPM_RESET_GPO3_PL_2	
XILPM_RESET_GPO3_PL_3	
XILPM_RESET_GPO3_PL_4	
XILPM_RESET_GPO3_PL_5	
XILPM_RESET_GPO3_PL_6	
XILPM_RESET_GPO3_PL_7	
XILPM_RESET_GPO3_PL_8	
XILPM_RESET_GPO3_PL_9	
XILPM_RESET_GPO3_PL_10	
XILPM_RESET_GPO3_PL_11	
XILPM_RESET_GPO3_PL_12	
XILPM_RESET_GPO3_PL_13	
XILPM_RESET_GPO3_PL_14	
XILPM_RESET_GPO3_PL_15	
XILPM_RESET_GPO3_PL_16	
XILPM_RESET_GPO3_PL_17	
XILPM_RESET_GPO3_PL_18	
XILPM_RESET_GPO3_PL_19	
XILPM_RESET_GPO3_PL_20	
XILPM_RESET_GPO3_PL_21	
XILPM_RESET_GPO3_PL_22	
XILPM_RESET_GPO3_PL_23	
XILPM_RESET_GPO3_PL_24	
XILPM_RESET_GPO3_PL_25	
XILPM_RESET_GPO3_PL_26	
XILPM_RESET_GPO3_PL_27	
XILPM_RESET_GPO3_PL_28	
XILPM_RESET_GPO3_PL_29	
XILPM_RESET_GPO3_PL_30	
XILPM_RESET_GPO3_PL_31	
XILPM_RESET_RPU_LS	
XILPM_RESET_PS_ONLY	
XILPM_RESET_PL	
XILPM_RESET_GPIO5_EMIO_92	
XILPM_RESET_GPIO5_EMIO_93	
XILPM_RESET_GPIO5_EMIO_94	
XILPM_RESET_GPIO5_EMIO_95	

Enumeration XPmNotifyEvent

PM Notify Events Enum

Table 62: **Enumeration XPmNotifyEvent Values**

Value	Description
EVENT_STATE_CHANGE	
EVENT_ZERO_USERS	
EVENT_STATE_CHANGE	
EVENT_ZERO_USERS	

Enumeration XPmClock

PM Clock IDs

Table 63: **Enumeration XPmClock Values**

Value	Description
PM_CLOCK_IOPLL	
PM_CLOCK_RPLL	
PM_CLOCK_APLL	
PM_CLOCK_DPLL	
PM_CLOCK_VPLL	
PM_CLOCK_IOPLL_TO_FPD	
PM_CLOCK_RPLL_TO_FPD	
PM_CLOCK_APLL_TO_LPD	
PM_CLOCK_DPLL_TO_LPD	
PM_CLOCK_VPLL_TO_LPD	
PM_CLOCK_ACPU	
PM_CLOCK_ACPU_HALF	
PM_CLOCK_DBG_FPD	
PM_CLOCK_DBG_LPD	
PM_CLOCK_DBG_TRACE	
PM_CLOCK_DBG_TSTMP	
PM_CLOCK_DP_VIDEO_REF	
PM_CLOCK_DP_AUDIO_REF	
PM_CLOCK_DP_STC_REF	
PM_CLOCK_GDMA_REF	
PM_CLOCK_DPDMA_REF	
PM_CLOCK_DDR_REF	
PM_CLOCK_SATA_REF	

Table 63: **Enumeration XPmClock Values** *(cont'd)*

Value	Description
PM_CLOCK_PCIE_REF	
PM_CLOCK_GPU_REF	
PM_CLOCK_GPU_PP0_REF	
PM_CLOCK_GPU_PP1_REF	
PM_CLOCK_TOPSW_MAIN	
PM_CLOCK_TOPSW_LSBUS	
PM_CLOCK_GTGREF0_REF	
PM_CLOCK_LPD_SWITCH	
PM_CLOCK_LPD_LSBUS	
PM_CLOCK_USB0_BUS_REF	
PM_CLOCK_USB1_BUS_REF	
PM_CLOCK_USB3_DUAL_REF	
PM_CLOCK_USB0	
PM_CLOCK_USB1	
PM_CLOCK_CPU_R5	
PM_CLOCK_CPU_R5_CORE	
PM_CLOCK_CSU_SPB	
PM_CLOCK_CSU_PLL	
PM_CLOCK_PCAP	
PM_CLOCK_IOU_SWITCH	
PM_CLOCK_GEM_TSU_REF	
PM_CLOCK_GEM_TSU	
PM_CLOCK_GEM0_TX	
PM_CLOCK_GEM1_TX	
PM_CLOCK_GEM2_TX	
PM_CLOCK_GEM3_TX	
PM_CLOCK_GEM0_RX	
PM_CLOCK_GEM1_RX	
PM_CLOCK_GEM2_RX	
PM_CLOCK_GEM3_RX	
PM_CLOCK_QSPI_REF	
PM_CLOCK_SDIO0_REF	
PM_CLOCK_SDIO1_REF	
PM_CLOCK_UART0_REF	
PM_CLOCK_UART1_REF	
PM_CLOCK_SPI0_REF	
PM_CLOCK_SPI1_REF	
PM_CLOCK_NAND_REF	
PM_CLOCK_I2C0_REF	

Table 63: **Enumeration XPmClock Values** *(cont'd)*

Value	Description
PM_CLOCK_I2C1_REF	
PM_CLOCK_CAN0_REF	
PM_CLOCK_CAN1_REF	
PM_CLOCK_CAN0	
PM_CLOCK_CAN1	
PM_CLOCK_DLL_REF	
PM_CLOCK_ADMA_REF	
PM_CLOCK_TIMESTAMP_REF	
PM_CLOCK_AMS_REF	
PM_CLOCK_PL0_REF	
PM_CLOCK_PL1_REF	
PM_CLOCK_PL2_REF	
PM_CLOCK_PL3_REF	
PM_CLOCK_WDT	
PM_CLOCK_IOPLL_INT	
PM_CLOCK_IOPLL_PRE_SRC	
PM_CLOCK_IOPLL_HALF	
PM_CLOCK_IOPLL_INT_MUX	
PM_CLOCK_IOPLL_POST_SRC	
PM_CLOCK_RPLL_INT	
PM_CLOCK_RPLL_PRE_SRC	
PM_CLOCK_RPLL_HALF	
PM_CLOCK_RPLL_INT_MUX	
PM_CLOCK_RPLL_POST_SRC	
PM_CLOCK_APLL_INT	
PM_CLOCK_APLL_PRE_SRC	
PM_CLOCK_APLL_HALF	
PM_CLOCK_APLL_INT_MUX	
PM_CLOCK_APLL_POST_SRC	
PM_CLOCK_DPLL_INT	
PM_CLOCK_DPLL_PRE_SRC	
PM_CLOCK_DPLL_HALF	
PM_CLOCK_DPLL_INT_MUX	
PM_CLOCK_DPLL_POST_SRC	
PM_CLOCK_VPLL_INT	
PM_CLOCK_VPLL_PRE_SRC	
PM_CLOCK_VPLL_HALF	
PM_CLOCK_VPLL_INT_MUX	
PM_CLOCK_VPLL_POST_SRC	

Table 63: **Enumeration XPmClock Values** *(cont'd)*

Value	Description
PM_CLOCK_CAN0_MIO	
PM_CLOCK_CAN1_MIO	
PM_CLOCK_ACPU_FULL	
PM_CLOCK_GEM0_REF	
PM_CLOCK_GEM1_REF	
PM_CLOCK_GEM2_REF	
PM_CLOCK_GEM3_REF	
PM_CLOCK_GEM0_REF_UNGATED	
PM_CLOCK_GEM1_REF_UNGATED	
PM_CLOCK_GEM2_REF_UNGATED	
PM_CLOCK_GEM3_REF_UNGATED	
PM_CLOCK_EXT_PSS_REF	
PM_CLOCK_EXT_VIDEO	
PM_CLOCK_EXT_PSS_ALT_REF	
PM_CLOCK_EXT_AUX_REF	
PM_CLOCK_EXT_GT_CRX_REF	
PM_CLOCK_EXT_SWDT0	
PM_CLOCK_EXT_SWDT1	
PM_CLOCK_EXT_GEM0_TX_EMIO	
PM_CLOCK_EXT_GEM1_TX_EMIO	
PM_CLOCK_EXT_GEM2_TX_EMIO	
PM_CLOCK_EXT_GEM3_TX_EMIO	
PM_CLOCK_EXT_GEM0_RX_EMIO	
PM_CLOCK_EXT_GEM1_RX_EMIO	
PM_CLOCK_EXT_GEM2_RX_EMIO	
PM_CLOCK_EXT_GEM3_RX_EMIO	
PM_CLOCK_EXT_MIO50_OR_MIO51	
PM_CLOCK_EXT_MIO0	
PM_CLOCK_EXT_MIO1	
PM_CLOCK_EXT_MIO2	
PM_CLOCK_EXT_MIO3	
PM_CLOCK_EXT_MIO4	
PM_CLOCK_EXT_MIO5	
PM_CLOCK_EXT_MIO6	
PM_CLOCK_EXT_MIO7	
PM_CLOCK_EXT_MIO8	
PM_CLOCK_EXT_MIO9	
PM_CLOCK_EXT_MIO10	
PM_CLOCK_EXT_MIO11	

Table 63: **Enumeration XPmClock Values** *(cont'd)*

Value	Description
PM_CLOCK_EXT_MIO12	
PM_CLOCK_EXT_MIO13	
PM_CLOCK_EXT_MIO14	
PM_CLOCK_EXT_MIO15	
PM_CLOCK_EXT_MIO16	
PM_CLOCK_EXT_MIO17	
PM_CLOCK_EXT_MIO18	
PM_CLOCK_EXT_MIO19	
PM_CLOCK_EXT_MIO20	
PM_CLOCK_EXT_MIO21	
PM_CLOCK_EXT_MIO22	
PM_CLOCK_EXT_MIO23	
PM_CLOCK_EXT_MIO24	
PM_CLOCK_EXT_MIO25	
PM_CLOCK_EXT_MIO26	
PM_CLOCK_EXT_MIO27	
PM_CLOCK_EXT_MIO28	
PM_CLOCK_EXT_MIO29	
PM_CLOCK_EXT_MIO30	
PM_CLOCK_EXT_MIO31	
PM_CLOCK_EXT_MIO32	
PM_CLOCK_EXT_MIO33	
PM_CLOCK_EXT_MIO34	
PM_CLOCK_EXT_MIO35	
PM_CLOCK_EXT_MIO36	
PM_CLOCK_EXT_MIO37	
PM_CLOCK_EXT_MIO38	
PM_CLOCK_EXT_MIO39	
PM_CLOCK_EXT_MIO40	
PM_CLOCK_EXT_MIO41	
PM_CLOCK_EXT_MIO42	
PM_CLOCK_EXT_MIO43	
PM_CLOCK_EXT_MIO44	
PM_CLOCK_EXT_MIO45	
PM_CLOCK_EXT_MIO46	
PM_CLOCK_EXT_MIO47	
PM_CLOCK_EXT_MIO48	
PM_CLOCK_EXT_MIO49	
PM_CLOCK_EXT_MIO50	

Table 63: **Enumeration XPmClock Values** *(cont'd)*

Value	Description
PM_CLOCK_EXT_MIO51	
PM_CLOCK_EXT_MIO52	
PM_CLOCK_EXT_MIO53	
PM_CLOCK_EXT_MIO54	
PM_CLOCK_EXT_MIO55	
PM_CLOCK_EXT_MIO56	
PM_CLOCK_EXT_MIO57	
PM_CLOCK_EXT_MIO58	
PM_CLOCK_EXT_MIO59	
PM_CLOCK_EXT_MIO60	
PM_CLOCK_EXT_MIO61	
PM_CLOCK_EXT_MIO62	
PM_CLOCK_EXT_MIO63	
PM_CLOCK_EXT_MIO64	
PM_CLOCK_EXT_MIO65	
PM_CLOCK_EXT_MIO66	
PM_CLOCK_EXT_MIO67	
PM_CLOCK_EXT_MIO68	
PM_CLOCK_EXT_MIO69	
PM_CLOCK_EXT_MIO70	
PM_CLOCK_EXT_MIO71	
PM_CLOCK_EXT_MIO72	
PM_CLOCK_EXT_MIO73	
PM_CLOCK_EXT_MIO74	
PM_CLOCK_EXT_MIO75	
PM_CLOCK_EXT_MIO76	
PM_CLOCK_EXT_MIO77	

Enumeration XPmPllParam

Table 64: **Enumeration XPmPllParam Values**

Value	Description
PM_PLL_PARAM_ID_DIV2	
PM_PLL_PARAM_ID_FBDIV	
PM_PLL_PARAM_ID_DATA	
PM_PLL_PARAM_ID_PRE_SRC	
PM_PLL_PARAM_ID_POST_SRC	
PM_PLL_PARAM_ID_LOCK_DLY	

Table 64: **Enumeration XPmPllParam Values** *(cont'd)*

Value	Description
PM_PLL_PARAM_ID_LOCK_CNT	
PM_PLL_PARAM_ID_LFHF	
PM_PLL_PARAM_ID_CP	
PM_PLL_PARAM_ID_RES	

Enumeration XPmPllMode

Table 65: **Enumeration XPmPllMode Values**

Value	Description
PM_PLL_MODE_INTEGER	
PM_PLL_MODE_FRACTIONAL	
PM_PLL_MODE_RESET	
PM_PLL_MODE_RESET	
PM_PLL_MODE_INTEGER	
PM_PLL_MODE_FRACTIONAL	

Enumeration XPmPinFn

Table 66: **Enumeration XPmPinFn Values**

Value	Description
PINCTRL_FUNC_CAN0	
PINCTRL_FUNC_CAN1	
PINCTRL_FUNC_ETHERNET0	
PINCTRL_FUNC_ETHERNET1	
PINCTRL_FUNC_ETHERNET2	
PINCTRL_FUNC_ETHERNET3	
PINCTRL_FUNC_GEMTSU0	
PINCTRL_FUNC_GPIO0	
PINCTRL_FUNC_I2C0	
PINCTRL_FUNC_I2C1	
PINCTRL_FUNC_MDIO0	
PINCTRL_FUNC_MDIO1	
PINCTRL_FUNC_MDIO2	
PINCTRL_FUNC_MDIO3	
PINCTRL_FUNC_QSPI0	
PINCTRL_FUNC_QSPI_FBCLK	
PINCTRL_FUNC_QSPI_SS	

Table 66: **Enumeration XPmPinFn Values** *(cont'd)*

Value	Description
PINCTRL_FUNC_SPI0	
PINCTRL_FUNC_SPI1	
PINCTRL_FUNC_SPI0_SS	
PINCTRL_FUNC_SPI1_SS	
PINCTRL_FUNC_SDIO0	
PINCTRL_FUNC_SDIO0_PC	
PINCTRL_FUNC_SDIO0_CD	
PINCTRL_FUNC_SDIO0_WP	
PINCTRL_FUNC_SDIO1	
PINCTRL_FUNC_SDIO1_PC	
PINCTRL_FUNC_SDIO1_CD	
PINCTRL_FUNC_SDIO1_WP	
PINCTRL_FUNC_NAND0	
PINCTRL_FUNC_NAND0_CE	
PINCTRL_FUNC_NAND0_RB	
PINCTRL_FUNC_NAND0_DQS	
PINCTRL_FUNC_TTC0_CLK	
PINCTRL_FUNC_TTC0_WAV	
PINCTRL_FUNC_TTC1_CLK	
PINCTRL_FUNC_TTC1_WAV	
PINCTRL_FUNC_TTC2_CLK	
PINCTRL_FUNC_TTC2_WAV	
PINCTRL_FUNC_TTC3_CLK	
PINCTRL_FUNC_TTC3_WAV	
PINCTRL_FUNC_UART0	
PINCTRL_FUNC_UART1	
PINCTRL_FUNC_USB0	
PINCTRL_FUNC_USB1	
PINCTRL_FUNC_SWDT0_CLK	
PINCTRL_FUNC_SWDT0_RST	
PINCTRL_FUNC_SWDT1_CLK	
PINCTRL_FUNC_SWDT1_RST	
PINCTRL_FUNC_PMU0	
PINCTRL_FUNC_PCIE0	
PINCTRL_FUNC_CSU0	
PINCTRL_FUNC_DPAUX0	
PINCTRL_FUNC_PJTAG0	
PINCTRL_FUNC_TRACE0	
PINCTRL_FUNC_TRACE0_CLK	

Table 66: **Enumeration XPmPinFn Values** *(cont'd)*

Value	Description
PINCTRL_FUNC_TESTSCAN0	

Enumeration XPmPinParam

Table 67: **Enumeration XPmPinParam Values**

Value	Description
PINCTRL_CONFIG_SLEW_RATE	
PINCTRL_CONFIG_BIAS_STATUS	
PINCTRL_CONFIG_PULL_CTRL	
PINCTRL_CONFIG_SCHMITT_CMOS	
PINCTRL_CONFIG_DRIVE_STRENGTH	
PINCTRL_CONFIG_VOLTAGE_STATUS	

Definitions

Define PACK_PAYLOAD

Definition

```
#define PACK_PAYLOADpl[0] = (u32)(arg0);                         \
    pl[1] = (u32)(arg1);                         \
    pl[2] = (u32)(arg2);                         \
    pl[3] = (u32)(arg3);                         \
    pl[4] = (u32)(arg4);                         \
    pl[5] = (u32)(arg5);                         \
    pl[6] = (u32)(rsvd);                         \
    pm_dbg("%s(%x, %x, %x, %x, %x, %x)\n", (__func__), (arg1), (arg2),
(arg3), (arg4), (arg5), (rsvd));
```

Description

Define PACK_PAYLOAD0

Definition

```
#define PACK_PAYLOAD0PACK_PAYLOAD(pl, (api_id), 0U, 0U, 0U, 0U, 0U, 0U)
```

Description

Define PACK_PAYLOAD1

Definition

```
#define PACK_PAYLOAD1PACK_PAYLOAD(pl, (api_id), (arg1), 0U, 0U, 0U, 0U, 0U)
```

Description

Define PACK_PAYLOAD2

Definition

```
#define PACK_PAYLOAD2PACK_PAYLOAD(pl, (api_id), (arg1), (arg2), 0U, 0U, 0U,
0U)
```

Description

Define PACK_PAYLOAD3

Definition

```
#define PACK_PAYLOAD3PACK_PAYLOAD(pl, (api_id), (arg1), (arg2), (arg3), 0U,
0U, 0U)
```

Description

Define PACK_PAYLOAD4

Definition

```
#define PACK_PAYLOAD4PACK_PAYLOAD(pl, (api_id), (arg1), (arg2), (arg3),
(arg4), 0U, 0U)
```

Description

Define PACK_PAYLOAD5

Definition

```
#define PACK_PAYLOAD5PACK_PAYLOAD(pl, (api_id), (arg1), (arg2), (arg3),
(arg4), (arg5), 0U)
```

Description

Define PM_VERSION_MAJOR

Definition

```
#define PM_VERSION_MAJOR1
```

Description

Define PM_VERSION_MINOR

Definition

```
#define PM_VERSION_MINOR1
```

Description

Define PM_VERSION

Definition

```
#define PM_VERSION((PM_VERSION_MAJOR << 16) | PM_VERSION_MINOR)
```

Description

Define PM_CAP_ACCESS

Definition

```
#define PM_CAP_ACCESS0x1U
```

Description

Define PM_CAP_CONTEXT

Definition

```
#define PM_CAP_CONTEXT0x2U
```

Description

Define PM_CAP_WAKEUP

Definition

```
#define PM_CAP_WAKEUP0x4U
```

Description

Define NODE_STATE_OFF

Definition

```
#define NODE_STATE_OFF0
```

Description

Define NODE_STATE_ON

Definition

```
#define NODE_STATE_ON1
```

Description

Define PROC_STATE_FORCEDOFF

Definition

```
#define PROC_STATE_FORCEDOFF0
```

Description

Define PROC_STATE_ACTIVE

Definition

```
#define PROC_STATE_ACTIVE1
```

Description

Define PROC_STATE_SLEEP

Definition

```
#define PROC_STATE_SLEEP2
```

Description

Define PROC_STATE_SUSPENDING

Definition

```
#define PROC_STATE_SUSPENDING3
```

Description

Define MAX_LATENCY

Definition

```
#define MAX_LATENCY(~0U)
```

Description

Define MAX_QOS

Definition

```
#define MAX_QOS100U
```

Description

Define PMF_SHUTDOWN_TYPE_SHUTDOWN

Definition

```
#define PMF_SHUTDOWN_TYPE_SHUTDOWN0U
```

Description

Define PMF_SHUTDOWN_TYPE_RESET

Definition

```
#define PMF_SHUTDOWN_TYPE_RESET1U
```

Description

Define PMF_SHUTDOWN_SUBTYPE_SUBSYSTEM

Definition

```
#define PMF_SHUTDOWN_SUBTYPE_SUBSYSTEM0U
```

Description

Define PMF_SHUTDOWN_SUBTYPE_PS_ONLY

Definition

```
#define PMF_SHUTDOWN_SUBTYPE_PS_ONLY1U
```

Description

Define PMF_SHUTDOWN_SUBTYPE_SYSTEM

Definition

```
#define PMF_SHUTDOWN_SUBTYPE_SYSTEM2U
```

Description

Define PM_API_MIN

Definition

```
#define PM_API_MINPM_GET_API_VERSION
```

Description

Define PM_CLOCK_DIV0_ID

Definition

```
#define PM_CLOCK_DIV0_ID0U
```

Description

Define PM_CLOCK_DIV1_ID

Definition

```
#define PM_CLOCK_DIV1_ID1U
```

Description

Chapter 2

XilPM Versal ACAP APIs

Xilinx Power Management (XilPM) provides Embedded Energy Management Interface (EEMI) APIs for power management on Versal ACAP devices. For more details about EEMI, see the Embedded Energy Management Interface (EEMI) API User Guide (UG1200).

The platform and power management functionality used by the APU/RPU applications is provided by the files in the 'XilPM<version>/versal/client' folder, where '<version>' is the version of the XilPM library.

Table 68: **Quick Function Reference**

Type	Name	Arguments
XStatus	XPm_IpiSend	struct XPm_Proc *const Proc u32 * Payload
XStatus	Xpm_IpiReadBuff32	struct XPm_Proc *const Proc u32 * Val1 u32 * Val2 u32 * Val3
XStatus	XPm_InitXilpm	XIpiPsu * IpiInst
enum XPmBootStatus	XPm_GetBootStatus	void
XStatus	XPm_GetChipID	u32 * IDCode u32 * Version
XStatus	XPm_GetApiVersion	version
XStatus	XPm_RequestNode	const u32 DeviceId const u32 Capabilities const u32 QoS const u32 Ack
XStatus	XPm_ReleaseNode	const u32 DeviceId

Table 68: **Quick Function Reference** *(cont'd)*

Type	Name	Arguments
XStatus	XPm_SetRequirement	const u32 DeviceId const u32 Capabilities const u32 QoS const u32 Ack
XStatus	XPm_GetNodeStatus	const u32 DeviceId XPm_NodeStatus *const NodeStatus
XStatus	XPm_ResetAssert	const u32 ResetId const u32 Action
XStatus	XPm_ResetGetStatus	const u32 ResetId u32 *const State
XStatus	XPm_PinCtrlRequest	const u32 PinId
XStatus	XPm_PinCtrlRelease	const u32 PinId
XStatus	XPm_PinCtrlSetFunction	const u32 PinId const u32 FunctionId
XStatus	XPm_PinCtrlGetFunction	const u32 PinId u32 *const FunctionId
XStatus	XPm_PinCtrlSetParameter	const u32 PinId const u32 ParamId const u32 ParamVal
XStatus	XPm_PinCtrlGetParameter	const u32 PinId const u32 ParamId u32 *const ParamVal
XStatus	XPm_DevIoctl	const u32 DeviceId const pm_ioctl_id IoctlId const u32 Arg1 const u32 Arg2 u32 *const Response
XStatus	XPm_ClockEnable	const u32 ClockId
XStatus	XPm_ClockDisable	const u32 ClockId

Table 68: **Quick Function Reference** *(cont'd)*

Type	Name	Arguments
XStatus	XPm_ClockGetStatus	const u32 ClockId u32 *const State
XStatus	XPm_ClockSetDivider	const u32 ClockId const u32 Divider
XStatus	XPm_ClockGetDivider	const u32 ClockId u32 *const Divider
XStatus	XPm_ClockSetParent	const u32 ClockId const u32 ParentIdx
XStatus	XPm_ClockGetParent	const u32 ClockId u32 *const ParentIdx
int	XPm_ClockGetRate	const u32 ClockId u32 *const Rate
int	XPm_ClockSetRate	const u32 ClockId const u32 Rate
XStatus	XPm_PllSetParameter	const u32 ClockId const enum XPm_PllConfigParams ParamId const u32 Value
XStatus	XPm_PllGetParameter	const u32 ClockId const enum XPm_PllConfigParams ParamId u32 *const Value
XStatus	XPm_PllSetMode	const u32 ClockId const u32 Value
XStatus	XPm_PllGetMode	const u32 ClockId u32 *const Value
XStatus	XPm_SelfSuspend	const u32 DeviceId const u32 Latency const u8 State const u64 Address

Table 68: **Quick Function Reference** *(cont'd)*

Type	Name	Arguments
XStatus	XPm_RequestWakeUp	const u32 TargetDevId const u8 SetAddress const u64 Address const u32 Ack
void	XPm_SuspendFinalize	void
XStatus	XPm_RequestSuspend	const u32 TargetSubsystemId const u32 Ack const u32 Latency const u32 State
XStatus	XPm_AbortSuspend	reason
XStatus	XPm_ForcePowerDown	const u32 TargetDevId const u32 Ack
XStatus	XPm_SystemShutdown	const u32 Type const u32 SubType
XStatus	XPm_SetWakeUpSource	const u32 TargetDeviceId const u32 DeviceId const u32 Enable
XStatus	XPm_Query	Qid const u32 Arg1 const u32 Arg2 const u32 Arg3 u32 *const Data
int	XPm_SetMaxLatency	const u32 DeviceId const u32 Latency
XStatus	XPm_GetOpCharacteristic	const u32 DeviceId const enum `XPmOpCharType` Type u32 *const Result
int	XPm_InitFinalize	void
int	XPm_RegisterNotifier	`XPm_Notifier` *const Notifier

Table 68: **Quick Function Reference** *(cont'd)*

Type	Name	Arguments
int	XPm_UnregisterNotifier	XPm_Notifier *const Notifier
void	XPm_InitSuspendCb	const enum XPmSuspendReason Reason const u32 Latency const u32 State const u32 Timeout
void	XPm_AcknowledgeCb	const u32 Node const XStatus Status const u32 Oppoint
void	XPm_NotifyCb	const u32 Node const enum XPmNotifyEvent Event const u32 Oppoint
int	XPm_SetConfiguration	void
int	XPm_MmioWrite	void
int	XPm_MmioRead	void
XStatus	XPm_FeatureCheck	const u32 FeatureId u32 * Version

Functions

XPm_IpiSend

Sends IPI request to the target module.

Prototype

```
XStatus XPm_IpiSend(struct XPm_Proc *const Proc, u32 *Payload);
```

Parameters

The following table lists the `XPm_IpiSend` function arguments.

Table 69: **XPm_IpiSend Arguments**

Type	Name	Description
struct XPm_Proc *const	Proc	Pointer to the processor who is initiating request
u32 *	Payload	API id and call arguments to be written in IPI buffer

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

Xpm_IpiReadBuff32

Reads IPI Response after target module has handled interrupt.

Prototype

```
XStatus Xpm_IpiReadBuff32(struct XPm_Proc *const Proc, u32 *Val1, u32
*Val2, u32 *Val3);
```

Parameters

The following table lists the `Xpm_IpiReadBuff32` function arguments.

Table 70: **Xpm_IpiReadBuff32 Arguments**

Type	Name	Description
struct XPm_Proc *const	Proc	Pointer to the processor who is waiting and reading Response
u32 *	Val1	Used to return value from 2nd IPI buffer element (optional)
u32 *	Val2	Used to return value from 3rd IPI buffer element (optional)
u32 *	Val3	Used to return value from 4th IPI buffer element (optional)

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_InitXilpm

Initialize xilpm library.

Prototype

```
XStatus XPm_InitXilpm(XIpiPsu *IpiInst);
```

Parameters

The following table lists the `XPm_InitXilpm` function arguments.

Table 71: **XPm_InitXilpm Arguments**

Type	Name	Description
XIpiPsu *	IpiInst	Pointer to IPI driver instance

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_GetBootStatus

This Function returns information about the boot reason. If the boot is not a system startup but a resume, power down request bitfield for this processor will be cleared.

Prototype

```
enum
               XPmBootStatus
             XPm_GetBootStatus(void);
```

Returns

Returns processor boot status

- PM_RESUME : If the boot reason is because of system resume.
- PM_INITIAL_BOOT : If this boot is the initial system startup.

XPm_GetChipID

This function is used to request the version and ID code of a chip.

Prototype

```
XStatus XPm_GetChipID(u32 *IDCode, u32 *Version);
```

Parameters

The following table lists the `XPm_GetChipID` function arguments.

Table 72: **XPm_GetChipID Arguments**

Type	Name	Description
u32 *	IDCode	Returns the chip ID code.
u32 *	Version	Returns the chip version.

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_GetApiVersion

This function is used to request the version number of the API running on the platform management controller.

Prototype

```
XStatus XPm_GetApiVersion(u32 *Version);
```

Parameters

The following table lists the `XPm_GetApiVersion` function arguments.

Table 73: **XPm_GetApiVersion Arguments**

Type	Name	Description
Commented parameter version does not exist in function XPm_GetApiVersion.	version	Returns the API 32-bit version number.

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_RequestNode

This function is used to request the device.

Prototype

```
XStatus XPm_RequestNode(const u32 DeviceId, const u32 Capabilities, const
u32 QoS, const u32 Ack);
```

Parameters

The following table lists the `XPm_RequestNode` function arguments.

Table 74: **XPm_RequestNode Arguments**

Type	Name	Description
const u32	DeviceId	Device which needs to be requested
const u32	Capabilities	Device Capabilities, can be combined • PM_CAP_ACCESS : full access / functionality • PM_CAP_CONTEXT : preserve context • PM_CAP_WAKEUP : emit wake interrupts
const u32	QoS	Quality of Service (0-100) required
const u32	Ack	Requested acknowledge type

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ReleaseNode

This function is used to release the requested device.

Prototype

```
XStatus XPm_ReleaseNode(const u32 DeviceId);
```

Parameters

The following table lists the `XPm_ReleaseNode` function arguments.

Table 75: **XPm_ReleaseNode Arguments**

Type	Name	Description
const u32	DeviceId	Device which needs to be released

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_SetRequirement

This function is used to set the requirement for specified device.

Prototype

```
XStatus XPm_SetRequirement(const u32 DeviceId, const u32 Capabilities,
const u32 QoS, const u32 Ack);
```

Parameters

The following table lists the `XPm_SetRequirement` function arguments.

Table 76: **XPm_SetRequirement Arguments**

Type	Name	Description
const u32	DeviceId	Device for which requirement needs to be set
const u32	Capabilities	Device Capabilities, can be combined • PM_CAP_ACCESS : full access / functionality • PM_CAP_CONTEXT : preserve context • PM_CAP_WAKEUP : emit wake interrupts
const u32	QoS	Quality of Service (0-100) required
const u32	Ack	Requested acknowledge type

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_GetNodeStatus

This function is used to get the device status.

Prototype

```
XStatus XPm_GetNodeStatus(const u32 DeviceId, XPm_NodeStatus *const
NodeStatus);
```

Parameters

The following table lists the `XPm_GetNodeStatus` function arguments.

Table 77: **XPm_GetNodeStatus Arguments**

Type	Name	Description
const u32	DeviceId	Device for which status is requested

Table 77: **XPm_GetNodeStatus Arguments** *(cont'd)*

Type	Name	Description
XPm_NodeStatus *const	NodeStatus	Structure pointer to store device status • Status - The current power state of the device ◦ For CPU nodes: - 0 : if CPU is powered down, - 1 : if CPU is active (powered up), - 8 : if CPU is suspending (powered up) ◦ For power islands and power domains: - 0 : if island is powered down, - 2 : if island is powered up ◦ For slaves: - 0 : if slave is powered down, - 1 : if slave is powered up, - 9 : if slave is in retention • Requirement - Requirements placed on the device by the caller • Usage ◦ 0 : node is not used by any PU, ◦ 1 : node is used by caller exclusively, ◦ 2 : node is used by other PU(s) only, ◦ 3 : node is used by caller and by other PU(s)

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ResetAssert

This function is used to assert or release reset for a particular reset line. Alternatively a reset pulse can be requested as well.

Prototype

```
XStatus XPm_ResetAssert(const u32 ResetId, const u32 Action);
```

Parameters

The following table lists the `XPm_ResetAssert` function arguments.

Table 78: **XPm_ResetAssert Arguments**

Type	Name	Description
const u32	ResetId	Reset ID
const u32	Action	Reset action to be taken • PM_RESET_ACTION_RELEASE for Release Reset • PM_RESET_ACTION_ASSERT for Assert Reset • PM_RESET_ACTION_PULSE for Pulse Reset

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ResetGetStatus

This function is used to get the status of reset.

Prototype

```
XStatus XPm_ResetGetStatus(const u32 ResetId, u32 *const State);
```

Parameters

The following table lists the `XPm_ResetGetStatus` function arguments.

Table 79: **XPm_ResetGetStatus Arguments**

Type	Name	Description
const u32	ResetId	Reset ID
u32 *const	State	Pointer to store the status of specified reset • 0 for reset released • 1 for reset asserted

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_PinCtrlRequest

This function is used to request the pin.

Prototype

```
XStatus XPm_PinCtrlRequest(const u32 PinId);
```

Parameters

The following table lists the `XPm_PinCtrlRequest` function arguments.

Table 80: **XPm_PinCtrlRequest Arguments**

Type	Name	Description
const u32	PinId	Pin ID

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_PinCtrlRelease

This function is used to release the pin.

Prototype

```
XStatus XPm_PinCtrlRelease(const u32 PinId);
```

Parameters

The following table lists the `XPm_PinCtrlRelease` function arguments.

Table 81: **XPm_PinCtrlRelease Arguments**

Type	Name	Description
const u32	PinId	Pin ID

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_PinCtrlSetFunction

This function is used to set the function on specified pin.

Prototype

```
XStatus XPm_PinCtrlSetFunction(const u32 PinId, const u32 FunctionId);
```

Parameters

The following table lists the `XPm_PinCtrlSetFunction` function arguments.

Table 82: **XPm_PinCtrlSetFunction Arguments**

Type	Name	Description
const u32	PinId	Pin ID
const u32	FunctionId	Function ID which needs to be set

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_PinCtrlGetFunction

This function is used to get the function on specified pin.

Prototype

```
XStatus XPm_PinCtrlGetFunction(const u32 PinId, u32 *const FunctionId);
```

Parameters

The following table lists the `XPm_PinCtrlGetFunction` function arguments.

Table 83: **XPm_PinCtrlGetFunction Arguments**

Type	Name	Description
const u32	PinId	Pin ID
u32 *const	FunctionId	Pointer to Function ID

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_PinCtrlSetParameter

This function is used to set the pin parameter of specified pin.

The following table lists the parameter ID and their respective values:

ParamId	ParamVal
PINCTRL_CONFIG_SLEW_RATE	PINCTRL_SLEW_RATE_SLOW, PINCTRL_SLEW_RATE_FAST
PINCTRL_CONFIG_BIAS_STATUS	PINCTRL_BIAS_DISABLE, PINCTRL_BIAS_ENABLE

ParamId	ParamVal
PINCTRL_CONFIG_PULL_CTRL	PINCTRL_BIAS_PULL_DOWN, PINCTRL_BIAS_PULL_UP
PINCTRL_CONFIG_SCHMITT_CMOS	PINCTRL_INPUT_TYPE_CMOS, PINCTRL_INPUT_TYPE_SCHMITT
PINCTRL_CONFIG_DRIVE_STRENGTH	PINCTRL_DRIVE_STRENGTH_TRISTATE, PINCTRL_DRIVE_STRENGTH_4MA, PINCTRL_DRIVE_STRENGTH_8MA, PINCTRL_DRIVE_STRENGTH_12MA
PINCTRL_CONFIG_TRI_STATE	PINCTRL_TRI_STATE_DISABLE, PINCTRL_TRI_STATE_ENABLE

Prototype

```
XStatus XPm_PinCtrlSetParameter(const u32 PinId, const u32 ParamId, const
u32 ParamVal);
```

Parameters

The following table lists the `XPm_PinCtrlSetParameter` function arguments.

Table 84: **XPm_PinCtrlSetParameter Arguments**

Type	Name	Description
const u32	PinId	Pin ID
const u32	ParamId	Parameter ID
const u32	ParamVal	Value of the parameter

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_PinCtrlGetParameter

This function is used to get the pin parameter of specified pin.

The following table lists the parameter ID and their respective values:

ParamId	ParamVal
PINCTRL_CONFIG_SLEW_RATE	PINCTRL_SLEW_RATE_SLOW, PINCTRL_SLEW_RATE_FAST
PINCTRL_CONFIG_BIAS_STATUS	PINCTRL_BIAS_DISABLE, PINCTRL_BIAS_ENABLE
PINCTRL_CONFIG_PULL_CTRL	PINCTRL_BIAS_PULL_DOWN, PINCTRL_BIAS_PULL_UP
PINCTRL_CONFIG_SCHMITT_CMOS	PINCTRL_INPUT_TYPE_CMOS, PINCTRL_INPUT_TYPE_SCHMITT
PINCTRL_CONFIG_DRIVE_STRENGTH	PINCTRL_DRIVE_STRENGTH_TRISTATE, PINCTRL_DRIVE_STRENGTH_4MA, PINCTRL_DRIVE_STRENGTH_8MA, PINCTRL_DRIVE_STRENGTH_12MA
PINCTRL_CONFIG_VOLTAGE_STATUS	1 for 1.8v mode, 0 for 3.3v mode

ParamId	ParamVal
PINCTRL_CONFIG_TRI_STATE	PINCTRL_TRI_STATE_DISABLE, PINCTRL_TRI_STATE_ENABLE

Prototype

```
XStatus XPm_PinCtrlGetParameter(const u32 PinId, const u32 ParamId, u32
*const ParamVal);
```

Parameters

The following table lists the `XPm_PinCtrlGetParameter` function arguments.

Table 85: **XPm_PinCtrlGetParameter Arguments**

Type	Name	Description
const u32	PinId	Pin ID
const u32	ParamId	Parameter ID
u32 *const	ParamVal	Pointer to the value of the parameter

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_DevIoctl

This function performs driver-like IOCTL functions on shared system devices.

Prototype

```
XStatus XPm_DevIoctl(const u32 DeviceId, const pm_ioctl_id IoctlId, const
u32 Arg1, const u32 Arg2, u32 *const Response);
```

Parameters

The following table lists the `XPm_DevIoctl` function arguments.

Table 86: **XPm_DevIoctl Arguments**

Type	Name	Description
const u32	DeviceId	ID of the device
const pm_ioctl_id	IoctlId	IOCTL function ID
const u32	Arg1	Argument 1
const u32	Arg2	Argument 2
u32 *const	Response	Ioctl response

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ClockEnable

This function is used to enable the specified clock.

Prototype

```
XStatus XPm_ClockEnable(const u32 ClockId);
```

Parameters

The following table lists the `XPm_ClockEnable` function arguments.

Table 87: **XPm_ClockEnable Arguments**

Type	Name	Description
const u32	ClockId	Clock ID

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ClockDisable

This function is used to disable the specified clock.

Prototype

```
XStatus XPm_ClockDisable(const u32 ClockId);
```

Parameters

The following table lists the `XPm_ClockDisable` function arguments.

Table 88: **XPm_ClockDisable Arguments**

Type	Name	Description
const u32	ClockId	Clock ID

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ClockGetStatus

This function is used to get the state of specified clock.

Prototype

```
XStatus XPm_ClockGetStatus(const u32 ClockId, u32 *const State);
```

Parameters

The following table lists the `XPm_ClockGetStatus` function arguments.

Table 89: **XPm_ClockGetStatus Arguments**

Type	Name	Description
const u32	ClockId	Clock ID
u32 *const	State	Pointer to store the clock state • 1 for enable and 0 for disable

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ClockSetDivider

This function is used to set the divider value for specified clock.

Prototype

```
XStatus XPm_ClockSetDivider(const u32 ClockId, const u32 Divider);
```

Parameters

The following table lists the `XPm_ClockSetDivider` function arguments.

Table 90: **XPm_ClockSetDivider Arguments**

Type	Name	Description
const u32	ClockId	Clock ID
const u32	Divider	Value of the divider

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ClockGetDivider

This function is used to get divider value for specified clock.

Prototype

```
XStatus XPm_ClockGetDivider(const u32 ClockId, u32 *const Divider);
```

Parameters

The following table lists the `XPm_ClockGetDivider` function arguments.

Table 91: **XPm_ClockGetDivider Arguments**

Type	Name	Description
const u32	ClockId	Clock ID
u32 *const	Divider	Pointer to store divider value

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ClockSetParent

This function is used to set the parent for specified clock.

Prototype

```
XStatus XPm_ClockSetParent(const u32 ClockId, const u32 ParentIdx);
```

Parameters

The following table lists the `XPm_ClockSetParent` function arguments.

Table 92: **XPm_ClockSetParent Arguments**

Type	Name	Description
const u32	ClockId	Clock ID
const u32	ParentIdx	Parent clock index

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ClockGetParent

This function is used to get the parent of specified clock.

Prototype

```
XStatus XPm_ClockGetParent(const u32 ClockId, u32 *const ParentIdx);
```

Parameters

The following table lists the `XPm_ClockGetParent` function arguments.

Table 93: **XPm_ClockGetParent Arguments**

Type	Name	Description
const u32	ClockId	Clock ID
u32 *const	ParentIdx	Pointer to store the parent clock index

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ClockGetRate

This function is used to get rate of specified clock.

Prototype

```
int XPm_ClockGetRate(const u32 ClockId, u32 *const Rate);
```

Parameters

The following table lists the `XPm_ClockGetRate` function arguments.

Table 94: **XPm_ClockGetRate Arguments**

Type	Name	Description
const u32	ClockId	Clock ID
u32 *const	Rate	Pointer to store the rate clock

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ClockSetRate

This function is used to set the rate of specified clock.

Prototype

```
int XPm_ClockSetRate(const u32 ClockId, const u32 Rate);
```

Parameters

The following table lists the `XPm_ClockSetRate` function arguments.

Table 95: **XPm_ClockSetRate Arguments**

Type	Name	Description
const u32	ClockId	Clock ID
const u32	Rate	Clock rate

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_PllSetParameter

This function is used to set the parameters for specified PLL clock.

Prototype

```
XStatus XPm_PllSetParameter(const u32 ClockId, const enum
XPm_PllConfigParams ParamId, const u32 Value);
```

Parameters

The following table lists the `XPm_PllSetParameter` function arguments.

Table 96: **XPm_PllSetParameter Arguments**

Type	Name	Description
const u32	ClockId	Clock ID

Table 96: **XPm_PllSetParameter Arguments** *(cont'd)*

Type	Name	Description
const enum XPm_PllConfigParams	ParamId	Parameter ID • PM_PLL_PARAM_ID_DIV2 • PM_PLL_PARAM_ID_FBDIV • PM_PLL_PARAM_ID_DATA • PM_PLL_PARAM_ID_PRE_SRC • PM_PLL_PARAM_ID_POST_SRC • PM_PLL_PARAM_ID_LOCK_DLY • PM_PLL_PARAM_ID_LOCK_CNT • PM_PLL_PARAM_ID_LFHF • PM_PLL_PARAM_ID_CP • PM_PLL_PARAM_ID_RES
const u32	Value	Value of parameter (See register description for possible values)

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_PllGetParameter

This function is used to get the parameter of specified PLL clock.

Prototype

```
XStatus XPm_PllGetParameter(const u32 ClockId, const enum
XPm_PllConfigParams ParamId, u32 *const Value);
```

Parameters

The following table lists the `XPm_PllGetParameter` function arguments.

Table 97: **XPm_PllGetParameter Arguments**

Type	Name	Description
const u32	ClockId	Clock ID

Table 97: **XPm_PllGetParameter Arguments** *(cont'd)*

Type	Name	Description
const enum XPm_PllConfigParams	ParamId	Parameter ID • PM_PLL_PARAM_ID_DIV2 • PM_PLL_PARAM_ID_FBDIV • PM_PLL_PARAM_ID_DATA • PM_PLL_PARAM_ID_PRE_SRC • PM_PLL_PARAM_ID_POST_SRC • PM_PLL_PARAM_ID_LOCK_DLY • PM_PLL_PARAM_ID_LOCK_CNT • PM_PLL_PARAM_ID_LFHF • PM_PLL_PARAM_ID_CP • PM_PLL_PARAM_ID_RES
u32 *const	Value	Pointer to store parameter value (See register description for possible values)

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_PllSetMode

This function is used to set the mode of specified PLL clock.

Prototype

```
XStatus XPm_PllSetMode(const u32 ClockId, const u32 Value);
```

Parameters

The following table lists the `XPm_PllSetMode` function arguments.

Table 98: **XPm_PllSetMode Arguments**

Type	Name	Description
const u32	ClockId	Clock ID
const u32	Value	Mode which need to be set • 0 for Reset mode • 1 for Integer mode • 2 for Fractional mode

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_PllGetMode

This function is used to get the mode of specified PLL clock.

Prototype

```
XStatus XPm_PllGetMode(const u32 ClockId, u32 *const Value);
```

Parameters

The following table lists the `XPm_PllGetMode` function arguments.

Table 99: **XPm_PllGetMode Arguments**

Type	Name	Description
const u32	ClockId	Clock ID
u32 *const	Value	Pointer to store the value of mode • 0 for Reset mode • 1 for Integer mode • 2 for Fractional mode

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_SelfSuspend

This function is used by a CPU to declare that it is about to suspend itself.

Prototype

```
XStatus XPm_SelfSuspend(const u32 DeviceId, const u32 Latency, const u8
State, const u64 Address);
```

Parameters

The following table lists the `XPm_SelfSuspend` function arguments.

Table 100: **XPm_SelfSuspend Arguments**

Type	Name	Description
const u32	DeviceId	Device ID of the CPU
const u32	Latency	Maximum wake-up latency requirement in us(microsecs)
const u8	State	Instead of specifying a maximum latency, a CPU can also explicitly request a certain power state.
const u64	Address	Address from which to resume when woken up.

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_RequestWakeUp

This function can be used to request power up of a CPU node within the same PU, or to power up another PU.

Prototype

```
XStatus XPm_RequestWakeUp(const u32 TargetDevId, const u8 SetAddress, const
u64 Address, const u32 Ack);
```

Parameters

The following table lists the `XPm_RequestWakeUp` function arguments.

Table 101: **XPm_RequestWakeUp Arguments**

Type	Name	Description
const u32	TargetDevId	Device ID of the CPU or PU to be powered/woken up.
const u8	SetAddress	Specifies whether the start address argument is being passed. • 0 : do not set start address • 1 : set start address
const u64	Address	Address from which to resume when woken up. Will only be used if set_address is 1.
const u32	Ack	Requested acknowledge type

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_SuspendFinalize

This Function waits for firmware to finish all previous API requests sent by the PU and performs client specific actions to finish suspend procedure (e.g. execution of wfi instruction on A53 and R5 processors).

***Note*:** This function should not return if the suspend procedure is successful.

Prototype

```
void XPm_SuspendFinalize(void);
```

Returns

XPm_RequestSuspend

This function is used by a CPU to request suspend to another CPU.

Prototype

```
XStatus XPm_RequestSuspend(const u32 TargetSubsystemId, const u32 Ack,
const u32 Latency, const u32 State);
```

Parameters

The following table lists the `XPm_RequestSuspend` function arguments.

Table 102: **XPm_RequestSuspend Arguments**

Type	Name	Description
const u32	TargetSubsystemId	Subsystem ID of the target
const u32	Ack	Requested acknowledge type
const u32	Latency	Maximum wake-up latency requirement in us(microsecs)
const u32	State	Power State

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_AbortSuspend

This function is called by a CPU after a SelfSuspend call to notify the platform management controller that CPU has aborted suspend or in response to an init suspend request when the PU refuses to suspend.

Prototype

```
XStatus XPm_AbortSuspend(const enum XPmAbortReason Reason);
```

Parameters

The following table lists the `XPm_AbortSuspend` function arguments.

Table 103: **XPm_AbortSuspend Arguments**

Type	Name	Description
Commented parameter reason does not exist in function XPm_AbortSuspend.	reason	Reason code why the suspend can not be performed or completed • ABORT_REASON_WKUP_EVENT : local wakeup-event received • ABORT_REASON_PU_BUSY : PU is busy • ABORT_REASON_NO_PWRDN : no external powerdown supported • ABORT_REASON_UNKNOWN : unknown error during suspend procedure

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_ForcePowerDown

This function is used by PU to request a forced poweroff of another PU or its power island or power domain. This can be used for killing an unresponsive PU, in which case all resources of that PU will be automatically released.

***Note*:** Force power down may not be requested by a PU for itself.

Prototype

```
XStatus XPm_ForcePowerDown(const u32 TargetDevId, const u32 Ack);
```

Parameters

The following table lists the `XPm_ForcePowerDown` function arguments.

Table 104: **XPm_ForcePowerDown Arguments**

Type	Name	Description
const u32	TargetDevId	Device ID of the PU node to be forced powered down.
const u32	Ack	Requested acknowledge type

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_SystemShutdown

This function can be used by a privileged PU to shut down or restart the complete device.

Prototype

```
XStatus XPm_SystemShutdown(const u32 Type, const u32 SubType);
```

Parameters

The following table lists the `XPm_SystemShutdown` function arguments.

Table 105: **XPm_SystemShutdown Arguments**

Type	Name	Description
const u32	Type	Shutdown type (shutdown/restart)
const u32	SubType	Shutdown subtype (subsystem-only/PU-only/system)

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_SetWakeUpSource

This function is used by a CPU to set wakeup source.

Prototype

```
XStatus XPm_SetWakeUpSource(const u32 TargetDeviceId, const u32 DeviceId,
const u32 Enable);
```

Parameters

The following table lists the `XPm_SetWakeUpSource` function arguments.

Table 106: **XPm_SetWakeUpSource Arguments**

Type	Name	Description
const u32	TargetDeviceId	Device ID of the target
const u32	DeviceId	Device ID used as wakeup source
const u32	Enable	1 - Enable, 0 - Disable

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_Query

This function queries information about the platform resources.

Prototype

```
XStatus XPm_Query(const u32 QueryId, const u32 Arg1, const u32 Arg2, const
u32 Arg3, u32 *const Data);
```

Parameters

The following table lists the `XPm_Query` function arguments.

Table 107: **XPm_Query Arguments**

Type	Name	Description
Commented parameter Qid does not exist in function XPm_Query.	Qid	The type of data to query
const u32	Arg1	Query argument 1
const u32	Arg2	Query argument 2
const u32	Arg3	Query argument 3
u32 *const	Data	Pointer to the output data

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_SetMaxLatency

This function is used by a CPU to announce a change in the maximum wake-up latency requirements for a specific device currently used by that CPU.

Note: Setting maximum wake-up latency can constrain the set of possible power states a resource can be put into.

Prototype

```
int XPm_SetMaxLatency(const u32 DeviceId, const u32 Latency);
```

Parameters

The following table lists the `XPm_SetMaxLatency` function arguments.

Table 108: **XPm_SetMaxLatency Arguments**

Type	Name	Description
const u32	DeviceId	Device ID.
const u32	Latency	Maximum wake-up latency required.

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_GetOpCharacteristic

Call this function to request the power management controller to return information about an operating characteristic of a component.

Note: Currently power type is not supported for Versal.

Prototype

```
XStatus XPm_GetOpCharacteristic(const u32 DeviceId, const enum
XPmOpCharType Type, u32 *const Result);
```

Parameters

The following table lists the `XPm_GetOpCharacteristic` function arguments.

Table 109: **XPm_GetOpCharacteristic Arguments**

Type	Name	Description
const u32	DeviceId	Device ID.
const enum `XPmOpCharType`	Type	Type of operating characteristic requested: • power (current power consumption), • latency (current latency in micro seconds to return to active state), • temperature (current temperature in Celsius (Q8.7 format)),
u32 *const	Result	Used to return the requested operating characteristic.

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_InitFinalize

This function is called to notify the power management controller about the completed power management initialization.

Prototype

```
int XPm_InitFinalize(void);
```

Returns

XST_SUCCESS if successful, otherwise an error code

XPm_RegisterNotifier

A PU can call this function to request that the power management controller call its notify callback whenever a qualifying event occurs. One can request to be notified for a specific or any event related to a specific node.

***Note*:** The caller shall initialize the notifier object before invoking the XPm_RegisteredNotifier function. While notifier is registered, the notifier object shall not be modified by the caller.

Prototype

```
int XPm_RegisterNotifier(XPm_Notifier *const Notifier);
```

Parameters

The following table lists the `XPm_RegisterNotifier` function arguments.

Table 110: **XPm_RegisterNotifier Arguments**

Type	Name	Description
`XPm_Notifier` *const	Notifier	Pointer to the notifier object to be associated with the requested notification.

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_UnregisterNotifier

A PU calls this function to unregister for the previously requested notifications.

Prototype

```
int XPm_UnregisterNotifier(XPm_Notifier *const Notifier);
```

Parameters

The following table lists the `XPm_UnregisterNotifier` function arguments.

Table 111: **XPm_UnregisterNotifier Arguments**

Type	Name	Description
`XPm_Notifier` *const	Notifier	Pointer to the notifier object associated with the previously requested notification

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

XPm_InitSuspendCb

Callback function to be implemented in each PU, allowing the power management controller to request that the PU suspend itself.

Note: If the PU fails to act on this request the power management controller or the requesting PU may choose to employ the forceful power down option.

Prototype

```
void XPm_InitSuspendCb(const enum XPmSuspendReason Reason, const u32
Latency, const u32 State, const u32 Timeout);
```

Parameters

The following table lists the `XPm_InitSuspendCb` function arguments.

Table 112: **XPm_InitSuspendCb Arguments**

Type	Name	Description
const enum `XPmSuspendReason`	Reason	Suspend reason: • SUSPEND_REASON_PU_REQ : Request by another PU • SUSPEND_REASON_ALERT : Unrecoverable SysMon alert • SUSPEND_REASON_SHUTDOWN : System shutdown • SUSPEND_REASON_RESTART : System restart
const u32	Latency	Maximum wake-up latency in us(micro secs). This information can be used by the PU to decide what level of context saving may be required.

Table 112: **XPm_InitSuspendCb Arguments** *(cont'd)*

Type	Name	Description
const u32	State	Targeted sleep/suspend state.
const u32	Timeout	Timeout in ms, specifying how much time a PU has to initiate its suspend procedure before it's being considered unresponsive.

Returns

None

XPm_AcknowledgeCb

This function is called by the power management controller in response to any request where an acknowledge callback was requested, i.e. where the 'ack' argument passed by the PU was REQUEST_ACK_NON_BLOCKING.

Prototype

```
void XPm_AcknowledgeCb(const u32 Node, const XStatus Status, const u32
Oppoint);
```

Parameters

The following table lists the `XPm_AcknowledgeCb` function arguments.

Table 113: **XPm_AcknowledgeCb Arguments**

Type	Name	Description
const u32	Node	ID of the component or sub-system in question.
const XStatus	Status	Status of the operation: • OK: the operation completed successfully • ERR: the requested operation failed
const u32	Oppoint	Operating point of the node in question

Returns

None

XPm_NotifyCb

This function is called by the power management controller if an event the PU was registered for has occurred. It will populate the notifier data structure passed when calling XPm_RegisterNotifier.

Prototype

```
void XPm_NotifyCb(const u32 Node, const enum XPmNotifyEvent Event, const
u32 Oppoint);
```

Parameters

The following table lists the `XPm_NotifyCb` function arguments.

Table 114: **XPm_NotifyCb Arguments**

Type	Name	Description
const u32	Node	ID of the device the event notification is related to.
const enum `XPmNotifyEvent`	Event	ID of the event
const u32	Oppoint	Current operating state of the device.

Returns

None

XPm_SetConfiguration

Prototype

```
int XPm_SetConfiguration(const u32 Address);
```

XPm_MmioWrite

Prototype

```
int XPm_MmioWrite(const u32 Address, const u32 Mask, const u32 Value);
```

XPm_MmioRead

Prototype

```
int XPm_MmioRead(const u32 Address, u32 *const Value);
```

XPm_FeatureCheck

This function queries information about the feature version.

Prototype

```
XStatus XPm_FeatureCheck(const u32 FeatureId, u32 *Version);
```

Parameters

The following table lists the `XPm_FeatureCheck` function arguments.

Table 115: **XPm_FeatureCheck Arguments**

Type	Name	Description
const u32	FeatureId	The feature ID (API-ID)
u32 *	Version	Pointer to the output data where version of feature store. For the supported feature get non zero value in version, But if version is 0U that means feature not supported.

Returns

XST_SUCCESS if successful else XST_FAILURE or an error code or a reason code

Enumerations

Enumeration XPmAbortReason

PM abort reasons enumeration.

Table 116: **Enumeration XPmAbortReason Values**

Value	Description
ABORT_REASON_WKUP_EVENT	
ABORT_REASON_PU_BUSY	
ABORT_REASON_NO_PWRDN	
ABORT_REASON_UNKNOWN	

Enumeration XPmBootStatus

Boot status enumeration.

Table 117: **Enumeration XPmBootStatus Values**

Value	Description
PM_INITIAL_BOOT	boot is a fresh system startup
PM_RESUME	boot is a resume

Table 117: **Enumeration XPmBootStatus Values** *(cont'd)*

Value	Description
PM_BOOT_ERROR	error, boot cause cannot be identified

Enumeration XPmCapability

Device capability requirements enumeration.

Table 118: **Enumeration XPmCapability Values**

Value	Description
PM_CAP_ACCESS	Full access
PM_CAP_CONTEXT	Configuration and contents retained
PM_CAP_WAKEUP	Enabled as a wake-up source
PM_CAP_UNUSABLE	Not usable

Enumeration XPmDeviceUsage

Table 119: **Enumeration XPmDeviceUsage Values**

Value	Description
PM_USAGE_CURRENT_SUBSYSTEM	
PM_USAGE_OTHER_SUBSYSTEM	

Enumeration XPmResetActions

Table 120: **Enumeration XPmResetActions Values**

Value	Description
PM_RESET_ACTION_RELEASE	
PM_RESET_ACTION_ASSERT	
PM_RESET_ACTION_PULSE	

Enumeration XPmSuspendReason

Table 121: **Enumeration XPmSuspendReason Values**

Value	Description
SUSPEND_REASON_PU_REQ	
SUSPEND_REASON_ALERT	
SUSPEND_REASON_SYS_SHUTDOWN	

Enumeration XPmApiCbId_t

Table 122: **Enumeration XPmApiCbId_t Values**

Value	Description
PM_INIT_SUSPEND_CB	
PM_ACKNOWLEDGE_CB	
PM_NOTIFY_CB	

Enumeration pm_query_id

Table 123: **Enumeration pm_query_id Values**

Value	Description
XPM_QID_INVALID	
XPM_QID_CLOCK_GET_NAME	
XPM_QID_CLOCK_GET_TOPOLOGY	
XPM_QID_CLOCK_GET_FIXEDFACTOR_PARAMS	
XPM_QID_CLOCK_GET_MUXSOURCES	
XPM_QID_CLOCK_GET_ATTRIBUTES	
XPM_QID_PINCTRL_GET_NUM_PINS	
XPM_QID_PINCTRL_GET_NUM_FUNCTIONS	
XPM_QID_PINCTRL_GET_NUM_FUNCTION_GROUPS	
XPM_QID_PINCTRL_GET_FUNCTION_NAME	
XPM_QID_PINCTRL_GET_FUNCTION_GROUPS	
XPM_QID_PINCTRL_GET_PIN_GROUPS	
XPM_QID_CLOCK_GET_NUM_CLOCKS	
XPM_QID_CLOCK_GET_MAX_DIVISOR	

Enumeration PmPinFunIds

Table 124: **Enumeration PmPinFunIds Values**

Value	Description
PIN_FUNC_SPI0	
PIN_FUNC_SPI0_SS	
PIN_FUNC_SPI1	
PIN_FUNC_SPI1_SS	
PIN_FUNC_CAN0	
PIN_FUNC_CAN1	
PIN_FUNC_I2C0	

Table 124: **Enumeration PmPinFunIds Values** *(cont'd)*

Value	Description
PIN_FUNC_I2C1	
PIN_FUNC_I2C_PMC	
PIN_FUNC_TTC0_CLK	
PIN_FUNC_TTC0_WAV	
PIN_FUNC_TTC1_CLK	
PIN_FUNC_TTC1_WAV	
PIN_FUNC_TTC2_CLK	
PIN_FUNC_TTC2_WAV	
PIN_FUNC_TTC3_CLK	
PIN_FUNC_TTC3_WAV	
PIN_FUNC_WWDT0	
PIN_FUNC_WWDT1	
PIN_FUNC_SYSMON_I2C0	
PIN_FUNC_SYSMON_I2C0_ALERT	
PIN_FUNC_UART0	
PIN_FUNC_UART0_CTRL	
PIN_FUNC_UART1	
PIN_FUNC_UART1_CTRL	
PIN_FUNC_GPIO0	
PIN_FUNC_GPIO1	
PIN_FUNC_GPIO2	
PIN_FUNC_EMIO0	
PIN_FUNC_GEM0	
PIN_FUNC_GEM1	
PIN_FUNC_TRACE0	
PIN_FUNC_TRACE0_CLK	
PIN_FUNC_MDIO0	
PIN_FUNC_MDIO1	
PIN_FUNC_GEM_TSU0	
PIN_FUNC_PCIE0	
PIN_FUNC_SMAP0	
PIN_FUNC_USB0	
PIN_FUNC_SD0	
PIN_FUNC_SD0_PC	
PIN_FUNC_SD0_CD	
PIN_FUNC_SD0_WP	
PIN_FUNC_SD1	
PIN_FUNC_SD1_PC	
PIN_FUNC_SD1_CD	

Table 124: **Enumeration PmPinFunIds Values** *(cont'd)*

Value	Description
PIN_FUNC_SD1_WP	
PIN_FUNC_OSPI0	
PIN_FUNC_OSPI0_SS	
PIN_FUNC_QSPI0	
PIN_FUNC_QSPI0_FBCLK	
PIN_FUNC_QSPI0_SS	
PIN_FUNC_TEST_CLK	
PIN_FUNC_TEST_SCAN	
PIN_FUNC_TAMPER_TRIGGER	
MAX_FUNCTION	

Enumeration pm_pinctrl_config_param

Table 125: **Enumeration pm_pinctrl_config_param Values**

Value	Description
PINCTRL_CONFIG_SLEW_RATE	
PINCTRL_CONFIG_BIAS_STATUS	
PINCTRL_CONFIG_PULL_CTRL	
PINCTRL_CONFIG_SCHMITT_CMOS	
PINCTRL_CONFIG_DRIVE_STRENGTH	
PINCTRL_CONFIG_VOLTAGE_STATUS	
PINCTRL_CONFIG_TRI_STATE	
PINCTRL_CONFIG_MAX	

Enumeration pm_pinctrl_slew_rate

Table 126: **Enumeration pm_pinctrl_slew_rate Values**

Value	Description
PINCTRL_SLEW_RATE_FAST	
PINCTRL_SLEW_RATE_SLOW	

Enumeration pm_pinctrl_bias_status

Table 127: **Enumeration pm_pinctrl_bias_status Values**

Value	Description
PINCTRL_BIAS_DISABLE	

Table 127: **Enumeration pm_pinctrl_bias_status Values** *(cont'd)*

Value	Description
PINCTRL_BIAS_ENABLE	

Enumeration pm_pinctrl_pull_ctrl

Table 128: **Enumeration pm_pinctrl_pull_ctrl Values**

Value	Description
PINCTRL_BIAS_PULL_DOWN	
PINCTRL_BIAS_PULL_UP	

Enumeration pm_pinctrl_schmitt_cmos

Table 129: **Enumeration pm_pinctrl_schmitt_cmos Values**

Value	Description
PINCTRL_INPUT_TYPE_CMOS	
PINCTRL_INPUT_TYPE_SCHMITT	

Enumeration pm_pinctrl_drive_strength

Table 130: **Enumeration pm_pinctrl_drive_strength Values**

Value	Description
PINCTRL_DRIVE_STRENGTH_TRISTATE	
PINCTRL_DRIVE_STRENGTH_4MA	
PINCTRL_DRIVE_STRENGTH_8MA	
PINCTRL_DRIVE_STRENGTH_12MA	
PINCTRL_DRIVE_STRENGTH_MAX	

Enumeration pm_pinctrl_tri_state

Table 131: **Enumeration pm_pinctrl_tri_state Values**

Value	Description
PINCTRL_TRI_STATE_DISABLE	
PINCTRL_TRI_STATE_ENABLE	

Enumeration pm_ioctl_id

Table 132: **Enumeration pm_ioctl_id Values**

Value	Description
IOCTL_GET_RPU_OPER_MODE	
IOCTL_SET_RPU_OPER_MODE	
IOCTL_RPU_BOOT_ADDR_CONFIG	
IOCTL_TCM_COMB_CONFIG	
IOCTL_SET_TAPDELAY_BYPASS	
IOCTL_SET_SGMII_MODE	
IOCTL_SD_DLL_RESET	
IOCTL_SET_SD_TAPDELAY	
IOCTL_SET_PLL_FRAC_MODE	
IOCTL_GET_PLL_FRAC_MODE	
IOCTL_SET_PLL_FRAC_DATA	
IOCTL_GET_PLL_FRAC_DATA	
IOCTL_WRITE_GGS	
IOCTL_READ_GGS	
IOCTL_WRITE_PGGS	
IOCTL_READ_PGGS	
IOCTL_ULPI_RESET	
IOCTL_SET_BOOT_HEALTH_STATUS	
IOCTL_AFI	
IOCTL_PROBE_COUNTER_READ	
IOCTL_PROBE_COUNTER_WRITE	
IOCTL_OSPI_MUX_SELECT	
IOCTL_USB_SET_STATE	

Enumeration XPm_PllConfigParams

Table 133: **Enumeration XPm_PllConfigParams Values**

Value	Description
PM_PLL_PARAM_ID_DIV2	
PM_PLL_PARAM_ID_FBDIV	
PM_PLL_PARAM_ID_DATA	
PM_PLL_PARAM_ID_PRE_SRC	
PM_PLL_PARAM_ID_POST_SRC	
PM_PLL_PARAM_ID_LOCK_DLY	
PM_PLL_PARAM_ID_LOCK_CNT	
PM_PLL_PARAM_ID_LFHF	

Table 133: **Enumeration XPm_PllConfigParams Values** *(cont'd)*

Value	Description
PM_PLL_PARAM_ID_CP	
PM_PLL_PARAM_ID_RES	
PM_PLL_PARAM_MAX	

Enumeration XPmPllMode

Table 134: **Enumeration XPmPllMode Values**

Value	Description
PM_PLL_MODE_INTEGER	
PM_PLL_MODE_FRACTIONAL	
PM_PLL_MODE_RESET	

Enumeration XPmInitFunctions

PM init node functions

Table 135: **Enumeration XPmInitFunctions Values**

Value	Description
FUNC_INIT_START	
FUNC_INIT_FINISH	
FUNC_SCAN_CLEAR	
FUNC_BISR	
FUNC_LBIST	
FUNC_MEM_INIT	
FUNC_MBIST_CLEAR	
FUNC_HOUSECLEAN_PL	
FUNC_HOUSECLEAN_COMPLETE	
FUNC_XPPU_CTRL	
FUNC_XMPU_CTRL	

Enumeration XPmOpCharType

PM operating characteristic types enumeration.

Table 136: **Enumeration XPmOpCharType Values**

Value	Description
PM_OPCHAR_TYPE_POWER	
PM_OPCHAR_TYPE_TEMP	
PM_OPCHAR_TYPE_LATENCY	

Enumeration XPmNotifyEvent

PM notify events enumeration.

Table 137: **Enumeration XPmNotifyEvent Values**

Value	Description
EVENT_STATE_CHANGE	
EVENT_ZERO_USERS	

Definitions

Define PM_VERSION_MAJOR

Definition

```
#define PM_VERSION_MAJOR1UL
```

Description

Define PM_VERSION_MINOR

Definition

```
#define PM_VERSION_MINOR0UL
```

Description

Define PM_VERSION

Definition

```
#define PM_VERSION((PM_VERSION_MAJOR << 16) | PM_VERSION_MINOR)
```

Description

Define XPM_MAX_CAPABILITY

Definition

```
#define XPM_MAX_CAPABILITY((u32)
               PM_CAP_ACCESS
             | (u32)
               PM_CAP_CONTEXT
             | (u32)
               PM_CAP_WAKEUP
            )
```

Description

Define XPM_MAX_LATENCY

Definition

```
#define XPM_MAX_LATENCY(0xFFFFU)
```

Description

Define XPM_MAX_QOS

Definition

```
#define XPM_MAX_QOS(100)
```

Description

Define XPM_MIN_CAPABILITY

Definition

```
#define XPM_MIN_CAPABILITY(0)
```

Description

Define XPM_MIN_LATENCY

Definition

```
#define XPM_MIN_LATENCY(0)
```

Description

Define XPM_MIN_QOS

Definition

```
#define XPM_MIN_QOS(0)
```

Description

Define XPM_DEF_CAPABILITY

Definition

```
#define XPM_DEF_CAPABILITYXPM_MAX_CAPABILITY
```

Description

Define XPM_DEF_LATENCY

Definition

```
#define XPM_DEF_LATENCYXPM_MAX_LATENCY
```

Description

Define XPM_DEF_QOS

Definition

```
#define XPM_DEF_QOSXPM_MAX_QOS
```

Description

Define NODE_STATE_OFF

Definition

```
#define NODE_STATE_OFF(0U)
```

Description

Define NODE_STATE_ON

Definition

```
#define NODE_STATE_ON(1U)
```

Description

Define PROC_STATE_SLEEP

Definition

```
#define PROC_STATE_SLEEPNODE_STATE_OFF
```

Description

Define PROC_STATE_ACTIVE

Definition

```
#define PROC_STATE_ACTIVENODE_STATE_ON
```

Description

Define PROC_STATE_FORCEDOFF

Definition

```
#define PROC_STATE_FORCEDOFF(7U)
```

Description

Define PROC_STATE_SUSPENDING

Definition

```
#define PROC_STATE_SUSPENDING(8U)
```

Description

Define PM_SHUTDOWN_TYPE_SHUTDOWN

Definition

```
#define PM_SHUTDOWN_TYPE_SHUTDOWN(0U)
```

Description

Define PM_SHUTDOWN_TYPE_RESET

Definition

```
#define PM_SHUTDOWN_TYPE_RESET(1U)
```

Description

Define PM_SHUTDOWN_SUBTYPE_RST_SUBSYSTEM

Definition

```
#define PM_SHUTDOWN_SUBTYPE_RST_SUBSYSTEM(0U)
```

Description

Define PM_SHUTDOWN_SUBTYPE_RST_PS_ONLY

Definition

```
#define PM_SHUTDOWN_SUBTYPE_RST_PS_ONLY(1U)
```

Description

Define PM_SHUTDOWN_SUBTYPE_RST_SYSTEM

Definition

```
#define PM_SHUTDOWN_SUBTYPE_RST_SYSTEM(2U)
```

Description

Define PM_SUSPEND_STATE_CPU_IDLE

Definition

```
#define PM_SUSPEND_STATE_CPU_IDLE0x0U
```

Description

Define PM_SUSPEND_STATE_SUSPEND_TO_RAM

Definition

```
#define PM_SUSPEND_STATE_SUSPEND_TO_RAM0xFU
```

Description

Define XPM_RPU_MODE_LOCKSTEP

Definition

```
#define XPM_RPU_MODE_LOCKSTEP0U
```

Description

Define XPM_RPU_MODE_SPLIT

Definition

```
#define XPM_RPU_MODE_SPLIT1U
```

Description

Define XPM_RPU_BOOTMEM_LOVEC

Definition

```
#define XPM_RPU_BOOTMEM_LOVEC(0U)
```

Description

Define XPM_RPU_BOOTMEM_HIVEC

Definition

```
#define XPM_RPU_BOOTMEM_HIVEC(1U)
```

Description

Define XPM_RPU_TCM_SPLIT

Definition

```
#define XPM_RPU_TCM_SPLIT0U
```

Description

Define XPM_RPU_TCM_COMB

Definition

```
#define XPM_RPU_TCM_COMB1U
```

Description

Define XPM_BOOT_HEALTH_STATUS_MASK

Definition

```
#define XPM_BOOT_HEALTH_STATUS_MASK(0x1U)
```

Description

Define XPM_TAPDELAY_QSPI

Definition

```
#define XPM_TAPDELAY_QSPI(2U)
```

Description

Define XPM_TAPDELAY_BYPASS_DISABLE

Definition

```
#define XPM_TAPDELAY_BYPASS_DISABLE(0U)
```

Description

Define XPM_TAPDELAY_BYPASS_ENABLE

Definition

```
#define XPM_TAPDELAY_BYPASS_ENABLE(1U)
```

Description

Define XPM_OSPI_MUX_SEL_DMA

Definition

```
#define XPM_OSPI_MUX_SEL_DMA(0U)
```

Description

Define XPM_OSPI_MUX_SEL_LINEAR

Definition

```
#define XPM_OSPI_MUX_SEL_LINEAR(1U)
```

Description

Define XPM_OSPI_MUX_GET_MODE

Definition

```
#define XPM_OSPI_MUX_GET_MODE(2U)
```

Description

Define XPM_TAPDELAY_INPUT

Definition

```
#define XPM_TAPDELAY_INPUT(0U)
```

Description

Define XPM_TAPDELAY_OUTPUT

Definition

```
#define XPM_TAPDELAY_OUTPUT(1U)
```

Description

Define XPM_DLL_RESET_ASSERT

Definition

```
#define XPM_DLL_RESET_ASSERT(0U)
```

Description

Define XPM_DLL_RESET_RELEASE

Definition

```
#define XPM_DLL_RESET_RELEASE(1U)
```

Description

Define XPM_DLL_RESET_PULSE

Definition

```
#define XPM_DLL_RESET_PULSE(2U)
```

Description

Define XPM_RESET_REASON_EXT_POR

Definition

```
#define XPM_RESET_REASON_EXT_POR(0U)
```

Description

Define XPM_RESET_REASON_SW_POR

Definition

```
#define XPM_RESET_REASON_SW_POR(1U)
```

Description

Define XPM_RESET_REASON_SLR_POR

Definition

```
#define XPM_RESET_REASON_SLR_POR(2U)
```

Description

Define XPM_RESET_REASON_ERR_POR

Definition

```
#define XPM_RESET_REASON_ERR_POR(3U)
```

Description

Define XPM_RESET_REASON_DAP_SRST

Definition

```
#define XPM_RESET_REASON_DAP_SRST(7U)
```

Description

Define XPM_RESET_REASON_ERR_SRST

Definition

```
#define XPM_RESET_REASON_ERR_SRST(8U)
```

Description

Define XPM_RESET_REASON_SW_SRST

Definition

```
#define XPM_RESET_REASON_SW_SRST(9U)
```

Description

Define XPM_RESET_REASON_SLR_SRST

Definition

```
#define XPM_RESET_REASON_SLR_SRST(10U)
```

Description

Define XPM_RESET_REASON_INVALID

Definition

```
#define XPM_RESET_REASON_INVALID(0xFFU)
```

Description

Define XPM_PROBE_COUNTER_TYPE_LAR_LSR

Definition

```
#define XPM_PROBE_COUNTER_TYPE_LAR_LSR(0U)
```

Description

Define XPM_PROBE_COUNTER_TYPE_MAIN_CTL

Definition

```
#define XPM_PROBE_COUNTER_TYPE_MAIN_CTL(1U)
```

Description

Define XPM_PROBE_COUNTER_TYPE_CFG_CTL

Definition

```
#define XPM_PROBE_COUNTER_TYPE_CFG_CTL(2U)
```

Description

Define XPM_PROBE_COUNTER_TYPE_STATE_PERIOD

Definition

```
#define XPM_PROBE_COUNTER_TYPE_STATE_PERIOD(3U)
```

Description

Define XPM_PROBE_COUNTER_TYPE_PORT_SEL

Definition

```
#define XPM_PROBE_COUNTER_TYPE_PORT_SEL(4U)
```

Description

Define XPM_PROBE_COUNTER_TYPE_SRC

Definition

```
#define XPM_PROBE_COUNTER_TYPE_SRC(5U)
```

Description

Define XPM_PROBE_COUNTER_TYPE_VAL

Definition

```
#define XPM_PROBE_COUNTER_TYPE_VAL(6U)
```

Description

Define XST_API_BASE_VERSION

Definition

```
#define XST_API_BASE_VERSION(1U)
```

Description

Define XST_API_QUERY_DATA_VERSION

Definition

```
#define XST_API_QUERY_DATA_VERSION(2U)
```

Description

Define PM_GET_API_VERSION

Definition

```
#define PM_GET_API_VERSION1U
```

Description

Define PM_SET_CONFIGURATION

Definition

```
#define PM_SET_CONFIGURATION2U
```

Description

Define PM_GET_NODE_STATUS

Definition

```
#define PM_GET_NODE_STATUS3U
```

Description

Define PM_GET_OP_CHARACTERISTIC

Definition

```
#define PM_GET_OP_CHARACTERISTIC4U
```

Description

Define PM_REGISTER_NOTIFIER

Definition

```
#define PM_REGISTER_NOTIFIER5U
```

Description

Define PM_REQUEST_SUSPEND

Definition

```
#define PM_REQUEST_SUSPEND6U
```

Description

Define PM_SELF_SUSPEND

Definition

```
#define PM_SELF_SUSPEND7U
```

Description

Define PM_FORCE_POWERDOWN

Definition

```
#define PM_FORCE_POWERDOWN8U
```

Description

Define PM_ABORT_SUSPEND

Definition

```
#define PM_ABORT_SUSPEND9U
```

Description

Define PM_REQUEST_WAKEUP

Definition

```
#define PM_REQUEST_WAKEUP10U
```

Description

Define PM_SET_WAKEUP_SOURCE

Definition

```
#define PM_SET_WAKEUP_SOURCE11U
```

Description

Define PM_SYSTEM_SHUTDOWN

Definition

```
#define PM_SYSTEM_SHUTDOWN12U
```

Description

Define PM_REQUEST_NODE

Definition

```
#define PM_REQUEST_NODE13U
```

Description

Define PM_RELEASE_NODE

Definition

```
#define PM_RELEASE_NODE14U
```

Description

Define PM_SET_REQUIREMENT

Definition

```
#define PM_SET_REQUIREMENT15U
```

Description

Define PM_SET_MAX_LATENCY

Definition

```
#define PM_SET_MAX_LATENCY16U
```

Description

Define PM_RESET_ASSERT

Definition

```
#define PM_RESET_ASSERT17U
```

Description

Define PM_RESET_GET_STATUS

Definition

```
#define PM_RESET_GET_STATUS18U
```

Description

Define PM_MMIO_WRITE

Definition

```
#define PM_MMIO_WRITE19U
```

Description

Define PM_MMIO_READ

Definition

```
#define PM_MMIO_READ20U
```

Description

Define PM_INIT_FINALIZE

Definition

```
#define PM_INIT_FINALIZE21U
```

Description

Define PM_FPGA_LOAD

Definition

```
#define PM_FPGA_LOAD22U
```

Description

Define PM_FPGA_GET_STATUS

Definition

```
#define PM_FPGA_GET_STATUS23U
```

Description

Define PM_GET_CHIPID

Definition

```
#define PM_GET_CHIPID24U
```

Description

Define PM_SECURE_RSA_AES

Definition

```
#define PM_SECURE_RSA_AES25U
```

Description

Define PM_SECURE_SHA

Definition

```
#define PM_SECURE_SHA26U
```

Description

Define PM_SECURE_RSA

Definition

```
#define PM_SECURE_RSA27U
```

Description

Define PM_PINCTRL_REQUEST

Definition

```
#define PM_PINCTRL_REQUEST28U
```

Description

Define PM_PINCTRL_RELEASE

Definition

```
#define PM_PINCTRL_RELEASE29U
```

Description

Define PM_PINCTRL_GET_FUNCTION

Definition

```
#define PM_PINCTRL_GET_FUNCTION30U
```

Description

Define PM_PINCTRL_SET_FUNCTION

Definition

```
#define PM_PINCTRL_SET_FUNCTION31U
```

Description

Define PM_PINCTRL_CONFIG_PARAM_GET

Definition

```
#define PM_PINCTRL_CONFIG_PARAM_GET32U
```

Description

Define PM_PINCTRL_CONFIG_PARAM_SET

Definition

```
#define PM_PINCTRL_CONFIG_PARAM_SET33U
```

Description

Define PM_IOCTL

Definition

```
#define PM_IOCTL34U
```

Description

Define PM_QUERY_DATA

Definition

```
#define PM_QUERY_DATA35U
```

Description

Define PM_CLOCK_ENABLE

Definition

```
#define PM_CLOCK_ENABLE36U
```

Description

Define PM_CLOCK_DISABLE

Definition

```
#define PM_CLOCK_DISABLE37U
```

Description

Define PM_CLOCK_GETSTATE

Definition

```
#define PM_CLOCK_GETSTATE38U
```

Description

Define PM_CLOCK_SETDIVIDER

Definition

```
#define PM_CLOCK_SETDIVIDER39U
```

Description

Define PM_CLOCK_GETDIVIDER

Definition

```
#define PM_CLOCK_GETDIVIDER40U
```

Description

Define PM_CLOCK_SETRATE

Definition

```
#define PM_CLOCK_SETRATE41U
```

Description

Define PM_CLOCK_GETRATE

Definition

```
#define PM_CLOCK_GETRATE42U
```

Description

Define PM_CLOCK_SETPARENT

Definition

```
#define PM_CLOCK_SETPARENT43U
```

Description

Define PM_CLOCK_GETPARENT

Definition

```
#define PM_CLOCK_GETPARENT44U
```

Description

Define PM_SECURE_IMAGE

Definition

```
#define PM_SECURE_IMAGE45U
```

Description

Define PM_FPGA_READ

Definition

```
#define PM_FPGA_READ46U
```

Description

Define PM_PLL_SET_PARAMETER

Definition

```
#define PM_PLL_SET_PARAMETER48U
```

Description

Define PM_PLL_GET_PARAMETER

Definition

```
#define PM_PLL_GET_PARAMETER49U
```

Description

Define PM_PLL_SET_MODE

Definition

```
#define PM_PLL_SET_MODE50U
```

Description

Define PM_PLL_GET_MODE

Definition

```
#define PM_PLL_GET_MODE51U
```

Description

Define PM_REGISTER_ACCESS

Definition

```
#define PM_REGISTER_ACCESS52U
```

Description

Define PM_EFUSE_ACCESS

Definition

```
#define PM_EFUSE_ACCESS53U
```

Description

Define PM_ADD_SUBSYSTEM

Definition

```
#define PM_ADD_SUBSYSTEM54U
```

Description

Define PM_DESTROY_SUBSYSTEM

Definition

```
#define PM_DESTROY_SUBSYSTEM55U
```

Description

Define PM_DESCRIBE_NODES

Definition

```
#define PM_DESCRIBE_NODES56U
```

Description

Define PM_ADD_NODE

Definition

```
#define PM_ADD_NODE57U
```

Description

Define PM_ADD_NODE_PARENT

Definition

```
#define PM_ADD_NODE_PARENT58U
```

Description

Define PM_ADD_NODE_NAME

Definition

```
#define PM_ADD_NODE_NAME59U
```

Description

Define PM_ADD_REQUIREMENT

Definition

```
#define PM_ADD_REQUIREMENT60U
```

Description

Define PM_SET_CURRENT_SUBSYSTEM

Definition

```
#define PM_SET_CURRENT_SUBSYSTEM61U
```

Description

Define PM_INIT_NODE

Definition

```
#define PM_INIT_NODE62U
```

Description

Define PM_FEATURE_CHECK

Definition

```
#define PM_FEATURE_CHECK63U
```

Description

Define PM_ISO_CONTROL

Definition

```
#define PM_ISO_CONTROL64U
```

Description

Define PM_ACTIVATE_SUBSYSTEM

Definition

```
#define PM_ACTIVATE_SUBSYSTEM65U
```

Description

Define PM_API_MIN

Definition

```
#define PM_API_MINPM_GET_API_VERSION
```

Description

Define PM_API_MAX

Definition

```
#define PM_API_MAXPM_ISO_CONTROL
```

Description

Define PACK_PAYLOAD

Definition

```
#define PACK_PAYLOADPayload[0] = (u32)Arg0;                                \
    Payload[1] = (u32)Arg1;                        \
    Payload[2] = (u32)Arg2;                        \
    Payload[3] = (u32)Arg3;                        \
    Payload[4] = (u32)Arg4;                        \
    Payload[5] = (u32)Arg5;                        \
    XPm_Dbg("%s(%x, %x, %x, %x, %x)\r\n", __func__, Arg1, Arg2, Arg3, Arg4,
Arg5);
```

Description

Define LIBPM_MODULE_ID

Definition

```
#define LIBPM_MODULE_ID(0x02UL)
```

Description

Define HEADER

Definition

```
#define HEADER((len << 16U) | (LIBPM_MODULE_ID << 8U) | ((u32)ApiId))
```

Description

Define PACK_PAYLOAD0

Definition

```
#define PACK_PAYLOAD0PACK_PAYLOAD(Payload, HEADER(0UL, ApiId), 0, 0, 0, 0,
0)
```

Description

Define PACK_PAYLOAD1

Definition

```
#define PACK_PAYLOAD1PACK_PAYLOAD(Payload, HEADER(1UL, ApiId), Arg1, 0, 0,
0, 0)
```

Description

Define PACK_PAYLOAD2

Definition

```
#define PACK_PAYLOAD2PACK_PAYLOAD(Payload, HEADER(2UL, ApiId), Arg1, Arg2,
0, 0, 0)
```

Description

Define PACK_PAYLOAD3

Definition

```
#define PACK_PAYLOAD3PACK_PAYLOAD(Payload, HEADER(3UL, ApiId), Arg1, Arg2,
Arg3, 0, 0)
```

Description

Define PACK_PAYLOAD4

Definition

```
#define PACK_PAYLOAD4PACK_PAYLOAD(Payload, HEADER(4UL, ApiId), Arg1, Arg2,
Arg3, Arg4, 0)
```

Description

Define PACK_PAYLOAD5

Definition

```
#define PACK_PAYLOAD5PACK_PAYLOAD(Payload, HEADER(5UL, ApiId), Arg1, Arg2,
Arg3, Arg4, Arg5)
```

Description

Power Nodes

Definitions

Define PM_POWER_PMC

Definition

```
#define PM_POWER_PMC(0x4208001U)
```

Description

Define PM_POWER_LPD

Definition

```
#define PM_POWER_LPD(0x4210002U)
```

Description

Define PM_POWER_FPD

Definition

```
#define PM_POWER_FPD(0x420c003U)
```

Description

Define PM_POWER_NOC

Definition

```
#define PM_POWER_NOC(0x4214004U)
```

Description

Define PM_POWER_ME

Definition

```
#define PM_POWER_ME(0x421c005U)
```

Description

Define PM_POWER_PLD

Definition

```
#define PM_POWER_PLD(0x4220006U)
```

Description

Define PM_POWER_CPM

Definition

```
#define PM_POWER_CPM(0x4218007U)
```

Description

Define PM_POWER_PL_SYSMON

Definition

```
#define PM_POWER_PL_SYSMON(0x4208008U)
```

Description

Define PM_POWER_RPU0_0

Definition

```
#define PM_POWER_RPU0_0(0x4104009U)
```

Description

Define PM_POWER_GEM0

Definition

```
#define PM_POWER_GEM0(0x410400aU)
```

Description

Define PM_POWER_GEM1

Definition

```
#define PM_POWER_GEM1(0x410400bU)
```

Description

Define PM_POWER_OCM_0

Definition

```
#define PM_POWER_OCM_0(0x410400cU)
```

Description

Define PM_POWER_OCM_1

Definition

```
#define PM_POWER_OCM_1(0x410400dU)
```

Description

Define PM_POWER_OCM_2

Definition

```
#define PM_POWER_OCM_2(0x410400eU)
```

Description

Define PM_POWER_OCM_3

Definition

```
#define PM_POWER_OCM_3(0x410400fU)
```

Description

Define PM_POWER_TCM_0_A

Definition

```
#define PM_POWER_TCM_0_A(0x4104010U)
```

Description

Define PM_POWER_TCM_0_B

Definition

```
#define PM_POWER_TCM_0_B(0x4104011U)
```

Description

Define PM_POWER_TCM_1_A

Definition

```
#define PM_POWER_TCM_1_A(0x4104012U)
```

Description

Define PM_POWER_TCM_1_B

Definition

```
#define PM_POWER_TCM_1_B(0x4104013U)
```

Description

Define PM_POWER_ACPU_0

Definition

```
#define PM_POWER_ACPU_0(0x4104014U)
```

Description

Define PM_POWER_ACPU_1

Definition

```
#define PM_POWER_ACPU_1(0x4104015U)
```

Description

Define PM_POWER_L2_BANK_0

Definition

```
#define PM_POWER_L2_BANK_0(0x4104016U)
```

Description

Clock nodes

Definitions

Define PM_CLK_PMC_PLL

Definition

```
#define PM_CLK_PMC_PLL(0x8104001U)
```

Description

Define PM_CLK_APU_PLL

Definition

```
#define PM_CLK_APU_PLL(0x8104002U)
```

Description

Define PM_CLK_RPU_PLL

Definition

```
#define PM_CLK_RPU_PLL(0x8104003U)
```

Description

Define PM_CLK_CPM_PLL

Definition

```
#define PM_CLK_CPM_PLL(0x8104004U)
```

Description

Define PM_CLK_NOC_PLL

Definition

```
#define PM_CLK_NOC_PLL(0x8104005U)
```

Description

Define PM_CLK_PMC_PRESRC

Definition

```
#define PM_CLK_PMC_PRESRC(0x8208007U)
```

Description

Define PM_CLK_PMC_POSTCLK

Definition

```
#define PM_CLK_PMC_POSTCLK(0x8208008U)
```

Description

Define PM_CLK_PMC_PLL_OUT

Definition

```
#define PM_CLK_PMC_PLL_OUT(0x8208009U)
```

Description

Define PM_CLK_PPLL

Definition

```
#define PM_CLK_PPLL(0x820800aU)
```

Description

Define PM_CLK_NOC_PRESRC

Definition

```
#define PM_CLK_NOC_PRESRC(0x820800bU)
```

Description

Define PM_CLK_NOC_POSTCLK

Definition

```
#define PM_CLK_NOC_POSTCLK(0x820800cU)
```

Description

Define PM_CLK_NOC_PLL_OUT

Definition

```
#define PM_CLK_NOC_PLL_OUT(0x820800dU)
```

Description

Define PM_CLK_NPLL

Definition

```
#define PM_CLK_NPLL(0x820800eU)
```

Description

Define PM_CLK_APU_PRESRC

Definition

```
#define PM_CLK_APU_PRESRC(0x820800fU)
```

Description

Define PM_CLK_APU_POSTCLK

Definition

```
#define PM_CLK_APU_POSTCLK(0x8208010U)
```

Description

Define PM_CLK_APU_PLL_OUT

Definition

```
#define PM_CLK_APU_PLL_OUT(0x8208011U)
```

Description

Define PM_CLK_APLL

Definition

```
#define PM_CLK_APLL(0x8208012U)
```

Description

Define PM_CLK_RPU_PRESRC

Definition

```
#define PM_CLK_RPU_PRESRC(0x8208013U)
```

Description

Define PM_CLK_RPU_POSTCLK

Definition

```
#define PM_CLK_RPU_POSTCLK(0x8208014U)
```

Description

Define PM_CLK_RPU_PLL_OUT

Definition

```
#define PM_CLK_RPU_PLL_OUT(0x8208015U)
```

Description

Define PM_CLK_RPLL

Definition

```
#define PM_CLK_RPLL(0x8208016U)
```

Description

Define PM_CLK_CPM_PRESRC

Definition

```
#define PM_CLK_CPM_PRESRC(0x8208017U)
```

Description

Define PM_CLK_CPM_POSTCLK

Definition

```
#define PM_CLK_CPM_POSTCLK(0x8208018U)
```

Description

Define PM_CLK_CPM_PLL_OUT

Definition

```
#define PM_CLK_CPM_PLL_OUT(0x8208019U)
```

Description

Define PM_CLK_CPLL

Definition

```
#define PM_CLK_CPLL(0x820801aU)
```

Description

Define PM_CLK_PPLL_TO_XPD

Definition

```
#define PM_CLK_PPLL_TO_XPD(0x820801bU)
```

Description

Define PM_CLK_NPLL_TO_XPD

Definition

```
#define PM_CLK_NPLL_TO_XPD(0x820801cU)
```

Description

Define PM_CLK_APLL_TO_XPD

Definition

```
#define PM_CLK_APLL_TO_XPD(0x820801dU)
```

Description

Define PM_CLK_RPLL_TO_XPD

Definition

```
#define PM_CLK_RPLL_TO_XPD(0x820801eU)
```

Description

Define PM_CLK_EFUSE_REF

Definition

```
#define PM_CLK_EFUSE_REF(0x820801fU)
```

Description

Define PM_CLK_SYSMON_REF

Definition

```
#define PM_CLK_SYSMON_REF(0x8208020U)
```

Description

Define PM_CLK_IRO_SUSPEND_REF

Definition

```
#define PM_CLK_IRO_SUSPEND_REF(0x8208021U)
```

Description

Define PM_CLK_USB_SUSPEND

Definition

```
#define PM_CLK_USB_SUSPEND(0x8208022U)
```

Description

Define PM_CLK_SWITCH_TIMEOUT

Definition

```
#define PM_CLK_SWITCH_TIMEOUT(0x8208023U)
```

Description

Define PM_CLK_RCLK_PMC

Definition

```
#define PM_CLK_RCLK_PMC(0x8208024U)
```

Description

Define PM_CLK_RCLK_LPD

Definition

```
#define PM_CLK_RCLK_LPD(0x8208025U)
```

Description

Define PM_CLK_WDT

Definition

```
#define PM_CLK_WDT(0x8208026U)
```

Description

Define PM_CLK_TTC0

Definition

```
#define PM_CLK_TTC0(0x8208027U)
```

Description

Define PM_CLK_TTC1

Definition

```
#define PM_CLK_TTC1(0x8208028U)
```

Description

Define PM_CLK_TTC2

Definition

```
#define PM_CLK_TTC2(0x8208029U)
```

Description

Define PM_CLK_TTC3

Definition

```
#define PM_CLK_TTC3(0x820802aU)
```

Description

Define PM_CLK_GEM_TSU

Definition

```
#define PM_CLK_GEM_TSU(0x820802bU)
```

Description

Define PM_CLK_GEM_TSU_LB

Definition

```
#define PM_CLK_GEM_TSU_LB(0x820802cU)
```

Description

Define PM_CLK_MUXED_IRO_DIV2

Definition

```
#define PM_CLK_MUXED_IRO_DIV2(0x820802dU)
```

Description

Define PM_CLK_MUXED_IRO_DIV4

Definition

```
#define PM_CLK_MUXED_IRO_DIV4(0x820802eU)
```

Description

Define PM_CLK_PSM_REF

Definition

```
#define PM_CLK_PSM_REF(0x820802fU)
```

Description

Define PM_CLK_GEM0_RX

Definition

```
#define PM_CLK_GEM0_RX(0x8208030U)
```

Description

Define PM_CLK_GEM0_TX

Definition

```
#define PM_CLK_GEM0_TX(0x8208031U)
```

Description

Define PM_CLK_GEM1_RX

Definition

```
#define PM_CLK_GEM1_RX(0x8208032U)
```

Description

Define PM_CLK_GEM1_TX

Definition

```
#define PM_CLK_GEM1_TX(0x8208033U)
```

Description

Define PM_CLK_CPM_CORE_REF

Definition

```
#define PM_CLK_CPM_CORE_REF(0x8208034U)
```

Description

Define PM_CLK_CPM_LSBUS_REF

Definition

```
#define PM_CLK_CPM_LSBUS_REF(0x8208035U)
```

Description

Define PM_CLK_CPM_DBG_REF

Definition

```
#define PM_CLK_CPM_DBG_REF(0x8208036U)
```

Description

Define PM_CLK_CPM_AUX0_REF

Definition

```
#define PM_CLK_CPM_AUX0_REF(0x8208037U)
```

Description

Define PM_CLK_CPM_AUX1_REF

Definition

```
#define PM_CLK_CPM_AUX1_REF(0x8208038U)
```

Description

Define PM_CLK_QSPI_REF

Definition

```
#define PM_CLK_QSPI_REF(0x8208039U)
```

Description

Define PM_CLK_OSPI_REF

Definition

```
#define PM_CLK_OSPI_REF(0x820803aU)
```

Description

Define PM_CLK_SDIO0_REF

Definition

```
#define PM_CLK_SDIO0_REF(0x820803bU)
```

Description

Define PM_CLK_SDIO1_REF

Definition

```
#define PM_CLK_SDIO1_REF(0x820803cU)
```

Description

Define PM_CLK_PMC_LSBUS_REF

Definition

```
#define PM_CLK_PMC_LSBUS_REF(0x820803dU)
```

Description

Define PM_CLK_I2C_REF

Definition

```
#define PM_CLK_I2C_REF(0x820803eU)
```

Description

Define PM_CLK_TEST_PATTERN_REF

Definition

```
#define PM_CLK_TEST_PATTERN_REF(0x820803fU)
```

Description

Define PM_CLK_DFT_OSC_REF

Definition

```
#define PM_CLK_DFT_OSC_REF(0x8208040U)
```

Description

Define PM_CLK_PMC_PL0_REF

Definition

```
#define PM_CLK_PMC_PL0_REF(0x8208041U)
```

Description

Define PM_CLK_PMC_PL1_REF

Definition

```
#define PM_CLK_PMC_PL1_REF(0x8208042U)
```

Description

Define PM_CLK_PMC_PL2_REF

Definition

```
#define PM_CLK_PMC_PL2_REF(0x8208043U)
```

Description

Define PM_CLK_PMC_PL3_REF

Definition

```
#define PM_CLK_PMC_PL3_REF(0x8208044U)
```

Description

Define PM_CLK_CFU_REF

Definition

```
#define PM_CLK_CFU_REF(0x8208045U)
```

Description

Define PM_CLK_SPARE_REF

Definition

```
#define PM_CLK_SPARE_REF(0x8208046U)
```

Description

Define PM_CLK_NPI_REF

Definition

```
#define PM_CLK_NPI_REF(0x8208047U)
```

Description

Define PM_CLK_HSM0_REF

Definition

```
#define PM_CLK_HSM0_REF(0x8208048U)
```

Description

Define PM_CLK_HSM1_REF

Definition

```
#define PM_CLK_HSM1_REF(0x8208049U)
```

Description

Define PM_CLK_SD_DLL_REF

Definition

```
#define PM_CLK_SD_DLL_REF(0x820804aU)
```

Description

Define PM_CLK_FPD_TOP_SWITCH

Definition

```
#define PM_CLK_FPD_TOP_SWITCH(0x820804bU)
```

Description

Define PM_CLK_FPD_LSBUS

Definition

```
#define PM_CLK_FPD_LSBUS(0x820804cU)
```

Description

Define PM_CLK_ACPU

Definition

```
#define PM_CLK_ACPU(0x820804dU)
```

Description

Define PM_CLK_DBG_TRACE

Definition

```
#define PM_CLK_DBG_TRACE(0x820804eU)
```

Description

Define PM_CLK_DBG_FPD

Definition

```
#define PM_CLK_DBG_FPD(0x820804fU)
```

Description

Define PM_CLK_LPD_TOP_SWITCH

Definition

```
#define PM_CLK_LPD_TOP_SWITCH(0x8208050U)
```

Description

Define PM_CLK_ADMA

Definition

```
#define PM_CLK_ADMA(0x8208051U)
```

Description

Define PM_CLK_LPD_LSBUS

Definition

```
#define PM_CLK_LPD_LSBUS(0x8208052U)
```

Description

Define PM_CLK_CPU_R5

Definition

```
#define PM_CLK_CPU_R5(0x8208053U)
```

Description

Define PM_CLK_CPU_R5_CORE

Definition

```
#define PM_CLK_CPU_R5_CORE(0x8208054U)
```

Description

Define PM_CLK_CPU_R5_OCM

Definition

```
#define PM_CLK_CPU_R5_OCM(0x8208055U)
```

Description

Define PM_CLK_CPU_R5_OCM2

Definition

```
#define PM_CLK_CPU_R5_OCM2(0x8208056U)
```

Description

Define PM_CLK_IOU_SWITCH

Definition

```
#define PM_CLK_IOU_SWITCH(0x8208057U)
```

Description

Define PM_CLK_GEM0_REF

Definition

```
#define PM_CLK_GEM0_REF(0x8208058U)
```

Description

Define PM_CLK_GEM1_REF

Definition

```
#define PM_CLK_GEM1_REF(0x8208059U)
```

Description

Define PM_CLK_GEM_TSU_REF

Definition

```
#define PM_CLK_GEM_TSU_REF(0x820805aU)
```

Description

Define PM_CLK_USB0_BUS_REF

Definition

```
#define PM_CLK_USB0_BUS_REF(0x820805bU)
```

Description

Define PM_CLK_UART0_REF

Definition

```
#define PM_CLK_UART0_REF(0x820805cU)
```

Description

Define PM_CLK_UART1_REF

Definition

```
#define PM_CLK_UART1_REF(0x820805dU)
```

Description

Define PM_CLK_SPI0_REF

Definition

```
#define PM_CLK_SPI0_REF(0x820805eU)
```

Description

Define PM_CLK_SPI1_REF

Definition

```
#define PM_CLK_SPI1_REF(0x820805fU)
```

Description

Define PM_CLK_CAN0_REF

Definition

```
#define PM_CLK_CAN0_REF(0x8208060U)
```

Description

Define PM_CLK_CAN1_REF

Definition

```
#define PM_CLK_CAN1_REF(0x8208061U)
```

Description

Define PM_CLK_I2C0_REF

Definition

```
#define PM_CLK_I2C0_REF(0x8208062U)
```

Description

Define PM_CLK_I2C1_REF

Definition

```
#define PM_CLK_I2C1_REF(0x8208063U)
```

Description

Define PM_CLK_DBG_LPD

Definition

```
#define PM_CLK_DBG_LPD(0x8208064U)
```

Description

Define PM_CLK_TIMESTAMP_REF

Definition

```
#define PM_CLK_TIMESTAMP_REF(0x8208065U)
```

Description

Define PM_CLK_DBG_TSTMP

Definition

```
#define PM_CLK_DBG_TSTMP(0x8208066U)
```

Description

Define PM_CLK_CPM_TOPSW_REF

Definition

```
#define PM_CLK_CPM_TOPSW_REF(0x8208067U)
```

Description

Define PM_CLK_USB3_DUAL_REF

Definition

```
#define PM_CLK_USB3_DUAL_REF(0x8208068U)
```

Description

Define PM_CLK_REF_CLK

Definition

```
#define PM_CLK_REF_CLK(0x830c06aU)
```

Description

Define PM_CLK_PL_ALT_REF_CLK

Definition

```
#define PM_CLK_PL_ALT_REF_CLK(0x830c06bU)
```

Description

Define PM_CLK_MUXED_IRO

Definition

```
#define PM_CLK_MUXED_IRO(0x830c06cU)
```

Description

Define PM_CLK_PL_EXT

Definition

```
#define PM_CLK_PL_EXT(0x830c06dU)
```

Description

Define PM_CLK_PL_LB

Definition

```
#define PM_CLK_PL_LB(0x830c06eU)
```

Description

Define PM_CLK_MIO_50_OR_51

Definition

```
#define PM_CLK_MIO_50_OR_51(0x830c06fU)
```

Description

Define PM_CLK_MIO_24_OR_25

Definition

```
#define PM_CLK_MIO_24_OR_25(0x830c070U)
```

Description

Define PM_CLK_EMIO

Definition

```
#define PM_CLK_EMIO(0x830c071U)
```

Description

Define PM_CLK_MIO

Definition

```
#define PM_CLK_MIO(0x830c072U)
```

Description

Define PM_CLK_PL_PMC_ALT_REF_CLK

Definition

```
#define PM_CLK_PL_PMC_ALT_REF_CLK(0x830c076U)
```

Description

Define PM_CLK_PL_LPD_ALT_REF_CLK

Definition

```
#define PM_CLK_PL_LPD_ALT_REF_CLK(0x830c077U)
```

Description

Define PM_CLK_PL_FPD_ALT_REF_CLK

Definition

```
#define PM_CLK_PL_FPD_ALT_REF_CLK(0x830c078U)
```

Description

MIO nodes

Definitions

Define PM_STMIC_LMIO_0

Definition

```
#define PM_STMIC_LMIO_0(0x14104001U)
```

Description

Define PM_STMIC_LMIO_1

Definition

```
#define PM_STMIC_LMIO_1(0x14104002U)
```

Description

Define PM_STMIC_LMIO_2

Definition

```
#define PM_STMIC_LMIO_2(0x14104003U)
```

Description

Define PM_STMIC_LMIO_3

Definition

```
#define PM_STMIC_LMIO_3(0x14104004U)
```

Description

Define PM_STMIC_LMIO_4

Definition

```
#define PM_STMIC_LMIO_4(0x14104005U)
```

Description

Define PM_STMIC_LMIO_5

Definition

```
#define PM_STMIC_LMIO_5(0x14104006U)
```

Description

Define PM_STMIC_LMIO_6

Definition

```
#define PM_STMIC_LMIO_6(0x14104007U)
```

Description

Define PM_STMIC_LMIO_7

Definition

```
#define PM_STMIC_LMIO_7(0x14104008U)
```

Description

Define PM_STMIC_LMIO_8

Definition

```
#define PM_STMIC_LMIO_8(0x14104009U)
```

Description

Define PM_STMIC_LMIO_9

Definition

```
#define PM_STMIC_LMIO_9(0x1410400aU)
```

Description

Define PM_STMIC_LMIO_10

Definition

```
#define PM_STMIC_LMIO_10(0x1410400bU)
```

Description

Define PM_STMIC_LMIO_11

Definition

```
#define PM_STMIC_LMIO_11(0x1410400cU)
```

Description

Define PM_STMIC_LMIO_12

Definition

```
#define PM_STMIC_LMIO_12(0x1410400dU)
```

Description

Define PM_STMIC_LMIO_13

Definition

```
#define PM_STMIC_LMIO_13(0x1410400eU)
```

Description

Define PM_STMIC_LMIO_14

Definition

```
#define PM_STMIC_LMIO_14(0x1410400fU)
```

Description

Define PM_STMIC_LMIO_15

Definition

```
#define PM_STMIC_LMIO_15(0x14104010U)
```

Description

Define PM_STMIC_LMIO_16

Definition

```
#define PM_STMIC_LMIO_16(0x14104011U)
```

Description

Define PM_STMIC_LMIO_17

Definition

```
#define PM_STMIC_LMIO_17(0x14104012U)
```

Description

Define PM_STMIC_LMIO_18

Definition

```
#define PM_STMIC_LMIO_18(0x14104013U)
```

Description

Define PM_STMIC_LMIO_19

Definition

```
#define PM_STMIC_LMIO_19(0x14104014U)
```

Description

Define PM_STMIC_LMIO_20

Definition

```
#define PM_STMIC_LMIO_20(0x14104015U)
```

Description

Define PM_STMIC_LMIO_21

Definition

```
#define PM_STMIC_LMIO_21(0x14104016U)
```

Description

Define PM_STMIC_LMIO_22

Definition

```
#define PM_STMIC_LMIO_22(0x14104017U)
```

Description

Define PM_STMIC_LMIO_23

Definition

```
#define PM_STMIC_LMIO_23(0x14104018U)
```

Description

Define PM_STMIC_LMIO_24

Definition

```
#define PM_STMIC_LMIO_24(0x14104019U)
```

Description

Define PM_STMIC_LMIO_25

Definition

```
#define PM_STMIC_LMIO_25(0x1410401aU)
```

Description

Define PM_STMIC_PMIO_0

Definition

```
#define PM_STMIC_PMIO_0(0x1410801bU)
```

Description

Define PM_STMIC_PMIO_1

Definition

```
#define PM_STMIC_PMIO_1(0x1410801cU)
```

Description

Define PM_STMIC_PMIO_2

Definition

```
#define PM_STMIC_PMIO_2(0x1410801dU)
```

Description

Define PM_STMIC_PMIO_3

Definition

```
#define PM_STMIC_PMIO_3(0x1410801eU)
```

Description

Define PM_STMIC_PMIO_4

Definition

```
#define PM_STMIC_PMIO_4(0x1410801fU)
```

Description

Define PM_STMIC_PMIO_5

Definition

```
#define PM_STMIC_PMIO_5(0x14108020U)
```

Description

Define PM_STMIC_PMIO_6

Definition

```
#define PM_STMIC_PMIO_6(0x14108021U)
```

Description

Define PM_STMIC_PMIO_7

Definition

```
#define PM_STMIC_PMIO_7(0x14108022U)
```

Description

Define PM_STMIC_PMIO_8

Definition

```
#define PM_STMIC_PMIO_8(0x14108023U)
```

Description

Define PM_STMIC_PMIO_9

Definition

```
#define PM_STMIC_PMIO_9(0x14108024U)
```

Description

Define PM_STMIC_PMIO_10

Definition

```
#define PM_STMIC_PMIO_10(0x14108025U)
```

Description

Define PM_STMIC_PMIO_11

Definition

```
#define PM_STMIC_PMIO_11(0x14108026U)
```

Description

Define PM_STMIC_PMIO_12

Definition

```
#define PM_STMIC_PMIO_12(0x14108027U)
```

Description

Define PM_STMIC_PMIO_13

Definition

```
#define PM_STMIC_PMIO_13(0x14108028U)
```

Description

Define PM_STMIC_PMIO_14

Definition

```
#define PM_STMIC_PMIO_14(0x14108029U)
```

Description

Define PM_STMIC_PMIO_15

Definition

```
#define PM_STMIC_PMIO_15(0x1410802aU)
```

Description

Define PM_STMIC_PMIO_16

Definition

```
#define PM_STMIC_PMIO_16(0x1410802bU)
```

Description

Define PM_STMIC_PMIO_17

Definition

```
#define PM_STMIC_PMIO_17(0x1410802cU)
```

Description

Define PM_STMIC_PMIO_18

Definition

```
#define PM_STMIC_PMIO_18(0x1410802dU)
```

Description

Define PM_STMIC_PMIO_19

Definition

```
#define PM_STMIC_PMIO_19(0x1410802eU)
```

Description

Define PM_STMIC_PMIO_20

Definition

```
#define PM_STMIC_PMIO_20(0x1410802fU)
```

Description

Define PM_STMIC_PMIO_21

Definition

```
#define PM_STMIC_PMIO_21(0x14108030U)
```

Description

Define PM_STMIC_PMIO_22

Definition

```
#define PM_STMIC_PMIO_22(0x14108031U)
```

Description

Define PM_STMIC_PMIO_23

Definition

```
#define PM_STMIC_PMIO_23(0x14108032U)
```

Description

Define PM_STMIC_PMIO_24

Definition

```
#define PM_STMIC_PMIO_24(0x14108033U)
```

Description

Define PM_STMIC_PMIO_25

Definition

```
#define PM_STMIC_PMIO_25(0x14108034U)
```

Description

Define PM_STMIC_PMIO_26

Definition

```
#define PM_STMIC_PMIO_26(0x14108035U)
```

Description

Define PM_STMIC_PMIO_27

Definition

```
#define PM_STMIC_PMIO_27(0x14108036U)
```

Description

Define PM_STMIC_PMIO_28

Definition

```
#define PM_STMIC_PMIO_28(0x14108037U)
```

Description

Define PM_STMIC_PMIO_29

Definition

```
#define PM_STMIC_PMIO_29(0x14108038U)
```

Description

Define PM_STMIC_PMIO_30

Definition

```
#define PM_STMIC_PMIO_30(0x14108039U)
```

Description

Define PM_STMIC_PMIO_31

Definition

```
#define PM_STMIC_PMIO_31(0x1410803aU)
```

Description

Define PM_STMIC_PMIO_32

Definition

```
#define PM_STMIC_PMIO_32(0x1410803bU)
```

Description

Define PM_STMIC_PMIO_33

Definition

```
#define PM_STMIC_PMIO_33(0x1410803cU)
```

Description

Define PM_STMIC_PMIO_34

Definition

```
#define PM_STMIC_PMIO_34(0x1410803dU)
```

Description

Define PM_STMIC_PMIO_35

Definition

```
#define PM_STMIC_PMIO_35(0x1410803eU)
```

Description

Define PM_STMIC_PMIO_36

Definition

```
#define PM_STMIC_PMIO_36(0x1410803fU)
```

Description

Define PM_STMIC_PMIO_37

Definition

```
#define PM_STMIC_PMIO_37(0x14108040U)
```

Description

Define PM_STMIC_PMIO_38

Definition

```
#define PM_STMIC_PMIO_38(0x14108041U)
```

Description

Define PM_STMIC_PMIO_39

Definition

```
#define PM_STMIC_PMIO_39(0x14108042U)
```

Description

Define PM_STMIC_PMIO_40

Definition

```
#define PM_STMIC_PMIO_40(0x14108043U)
```

Description

Define PM_STMIC_PMIO_41

Definition

```
#define PM_STMIC_PMIO_41(0x14108044U)
```

Description

Define PM_STMIC_PMIO_42

Definition

```
#define PM_STMIC_PMIO_42(0x14108045U)
```

Description

Define PM_STMIC_PMIO_43

Definition

```
#define PM_STMIC_PMIO_43(0x14108046U)
```

Description

Define PM_STMIC_PMIO_44

Definition

```
#define PM_STMIC_PMIO_44(0x14108047U)
```

Description

Define PM_STMIC_PMIO_45

Definition

```
#define PM_STMIC_PMIO_45(0x14108048U)
```

Description

Define PM_STMIC_PMIO_46

Definition

```
#define PM_STMIC_PMIO_46(0x14108049U)
```

Description

Define PM_STMIC_PMIO_47

Definition

```
#define PM_STMIC_PMIO_47(0x1410804aU)
```

Description

Define PM_STMIC_PMIO_48

Definition

```
#define PM_STMIC_PMIO_48(0x1410804bU)
```

Description

Define PM_STMIC_PMIO_49

Definition

```
#define PM_STMIC_PMIO_49(0x1410804cU)
```

Description

Define PM_STMIC_PMIO_50

Definition

```
#define PM_STMIC_PMIO_50(0x1410804dU)
```

Description

Define PM_STMIC_PMIO_51

Definition

```
#define PM_STMIC_PMIO_51(0x1410804eU)
```

Description

Device nodes

Definitions

Define PM_DEV_PLD_0

Definition

```
#define PM_DEV_PLD_0(0x18700000U)
```

Description

Define PM_DEV_PMC_PROC

Definition

```
#define PM_DEV_PMC_PROC(0x18104001U)
```

Description

Define PM_DEV_PSM_PROC

Definition

```
#define PM_DEV_PSM_PROC(0x18108002U)
```

Description

Define PM_DEV_ACPU_0

Definition

```
#define PM_DEV_ACPU_0(0x1810c003U)
```

Description

Define PM_DEV_ACPU_1

Definition

```
#define PM_DEV_ACPU_1(0x1810c004U)
```

Description

Define PM_DEV_RPU0_0

Definition

```
#define PM_DEV_RPU0_0(0x18110005U)
```

Description

Define PM_DEV_RPU0_1

Definition

```
#define PM_DEV_RPU0_1(0x18110006U)
```

Description

Define PM_DEV_OCM_0

Definition

```
#define PM_DEV_OCM_0(0x18314007U)
```

Description

Define PM_DEV_OCM_1

Definition

```
#define PM_DEV_OCM_1(0x18314008U)
```

Description

Define PM_DEV_OCM_2

Definition

```
#define PM_DEV_OCM_2(0x18314009U)
```

Description

Define PM_DEV_OCM_3

Definition

```
#define PM_DEV_OCM_3(0x1831400aU)
```

Description

Define PM_DEV_TCM_0_A

Definition

```
#define PM_DEV_TCM_0_A(0x1831800bU)
```

Description

Define PM_DEV_TCM_0_B

Definition

```
#define PM_DEV_TCM_0_B(0x1831800cU)
```

Description

Define PM_DEV_TCM_1_A

Definition

```
#define PM_DEV_TCM_1_A(0x1831800dU)
```

Description

Define PM_DEV_TCM_1_B

Definition

```
#define PM_DEV_TCM_1_B(0x1831800eU)
```

Description

Define PM_DEV_L2_BANK_0

Definition

```
#define PM_DEV_L2_BANK_0(0x1831c00fU)
```

Description

Define PM_DEV_DDR_0

Definition

```
#define PM_DEV_DDR_0(0x18320010U)
```

Description

Define PM_DEV_USB_0

Definition

```
#define PM_DEV_USB_0(0x18224018U)
```

Description

Define PM_DEV_GEM_0

Definition

```
#define PM_DEV_GEM_0(0x18224019U)
```

Description

Define PM_DEV_GEM_1

Definition

```
#define PM_DEV_GEM_1(0x1822401aU)
```

Description

Define PM_DEV_SPI_0

Definition

```
#define PM_DEV_SPI_0(0x1822401bU)
```

Description

Define PM_DEV_SPI_1

Definition

```
#define PM_DEV_SPI_1(0x1822401cU)
```

Description

Define PM_DEV_I2C_0

Definition

```
#define PM_DEV_I2C_0(0x1822401dU)
```

Description

Define PM_DEV_I2C_1

Definition

```
#define PM_DEV_I2C_1(0x1822401eU)
```

Description

Define PM_DEV_CAN_FD_0

Definition

```
#define PM_DEV_CAN_FD_0(0x1822401fU)
```

Description

Define PM_DEV_CAN_FD_1

Definition

```
#define PM_DEV_CAN_FD_1(0x18224020U)
```

Description

Define PM_DEV_UART_0

Definition

```
#define PM_DEV_UART_0(0x18224021U)
```

Description

Define PM_DEV_UART_1

Definition

```
#define PM_DEV_UART_1(0x18224022U)
```

Description

Define PM_DEV_GPIO

Definition

```
#define PM_DEV_GPIO(0x18224023U)
```

Description

Define PM_DEV_TTC_0

Definition

```
#define PM_DEV_TTC_0(0x18224024U)
```

Description

Define PM_DEV_TTC_1

Definition

```
#define PM_DEV_TTC_1(0x18224025U)
```

Description

Define PM_DEV_TTC_2

Definition

```
#define PM_DEV_TTC_2(0x18224026U)
```

Description

Define PM_DEV_TTC_3

Definition

```
#define PM_DEV_TTC_3(0x18224027U)
```

Description

Define PM_DEV_SWDT_LPD

Definition

```
#define PM_DEV_SWDT_LPD(0x18224028U)
```

Description

Define PM_DEV_SWDT_FPD

Definition

```
#define PM_DEV_SWDT_FPD(0x18224029U)
```

Description

Define PM_DEV_OSPI

Definition

```
#define PM_DEV_OSPI(0x1822402aU)
```

Description

Define PM_DEV_QSPI

Definition

```
#define PM_DEV_QSPI(0x1822402bU)
```

Description

Define PM_DEV_GPIO_PMC

Definition

```
#define PM_DEV_GPIO_PMC(0x1822402cU)
```

Description

Define PM_DEV_I2C_PMC

Definition

```
#define PM_DEV_I2C_PMC(0x1822402dU)
```

Description

Define PM_DEV_SDIO_0

Definition

```
#define PM_DEV_SDIO_0(0x1822402eU)
```

Description

Define PM_DEV_SDIO_1

Definition

```
#define PM_DEV_SDIO_1(0x1822402fU)
```

Description

Define PM_DEV_RTC

Definition

```
#define PM_DEV_RTC(0x18224034U)
```

Description

Define PM_DEV_ADMA_0

Definition

```
#define PM_DEV_ADMA_0(0x18224035U)
```

Description

Define PM_DEV_ADMA_1

Definition

```
#define PM_DEV_ADMA_1(0x18224036U)
```

Description

Define PM_DEV_ADMA_2

Definition

```
#define PM_DEV_ADMA_2(0x18224037U)
```

Description

Define PM_DEV_ADMA_3

Definition

```
#define PM_DEV_ADMA_3(0x18224038U)
```

Description

Define PM_DEV_ADMA_4

Definition

```
#define PM_DEV_ADMA_4(0x18224039U)
```

Description

Define PM_DEV_ADMA_5

Definition

```
#define PM_DEV_ADMA_5(0x1822403aU)
```

Description

Define PM_DEV_ADMA_6

Definition

```
#define PM_DEV_ADMA_6(0x1822403bU)
```

Description

Define PM_DEV_ADMA_7

Definition

```
#define PM_DEV_ADMA_7(0x1822403cU)
```

Description

Define PM_DEV_IPI_0

Definition

```
#define PM_DEV_IPI_0(0x1822403dU)
```

Description

Define PM_DEV_IPI_1

Definition

```
#define PM_DEV_IPI_1(0x1822403eU)
```

Description

Define PM_DEV_IPI_2

Definition

```
#define PM_DEV_IPI_2(0x1822403fU)
```

Description

Define PM_DEV_IPI_3

Definition

```
#define PM_DEV_IPI_3(0x18224040U)
```

Description

Define PM_DEV_IPI_4

Definition

```
#define PM_DEV_IPI_4(0x18224041U)
```

Description

Define PM_DEV_IPI_5

Definition

```
#define PM_DEV_IPI_5(0x18224042U)
```

Description

Define PM_DEV_IPI_6

Definition

```
#define PM_DEV_IPI_6(0x18224043U)
```

Description

Define PM_DEV_SOC

Definition

```
#define PM_DEV_SOC(0x18428044U)
```

Description

Define PM_DEV_DDRMC_0

Definition

```
#define PM_DEV_DDRMC_0(0x18520045U)
```

Description

Define PM_DEV_DDRMC_1

Definition

```
#define PM_DEV_DDRMC_1(0x18520046U)
```

Description

Define PM_DEV_DDRMC_2

Definition

```
#define PM_DEV_DDRMC_2(0x18520047U)
```

Description

Define PM_DEV_DDRMC_3

Definition

```
#define PM_DEV_DDRMC_3(0x18520048U)
```

Description

Define PM_DEV_GT_0

Definition

```
#define PM_DEV_GT_0(0x1862c049U)
```

Description

Define PM_DEV_GT_1

Definition

```
#define PM_DEV_GT_1(0x1862c04aU)
```

Description

Define PM_DEV_GT_2

Definition

```
#define PM_DEV_GT_2(0x1862c04bU)
```

Description

Define PM_DEV_GT_3

Definition

```
#define PM_DEV_GT_3(0x1862c04cU)
```

Description

Define PM_DEV_GT_4

Definition

```
#define PM_DEV_GT_4(0x1862c04dU)
```

Description

Define PM_DEV_GT_5

Definition

```
#define PM_DEV_GT_5(0x1862c04eU)
```

Description

Define PM_DEV_GT_6

Definition

```
#define PM_DEV_GT_6(0x1862c04fU)
```

Description

Define PM_DEV_GT_7

Definition

```
#define PM_DEV_GT_7(0x1862c050U)
```

Description

Define PM_DEV_GT_8

Definition

```
#define PM_DEV_GT_8(0x1862c051U)
```

Description

Define PM_DEV_GT_9

Definition

```
#define PM_DEV_GT_9(0x1862c052U)
```

Description

Define PM_DEV_GT_10

Definition

```
#define PM_DEV_GT_10(0x1862c053U)
```

Description

Define PM_DEV_EFUSE_CACHE

Definition

```
#define PM_DEV_EFUSE_CACHE(0x18330054U)
```

Description

Define PM_DEV_AMS_ROOT

Definition

```
#define PM_DEV_AMS_ROOT(0x18224055U)
```

Description

Define PM_DEV_AIE

Definition

```
#define PM_DEV_AIE(0x18224072U)
```

Description

Define PM_DEV_IPI_PMC

Definition

```
#define PM_DEV_IPI_PMC(0x18224073U)
```

Description

Subsystem nodes

Definitions

Define PM_SUBSYS_DEFAULT

Definition

```
#define PM_SUBSYS_DEFAULT(0x1c000000U)
```

Description

Define PM_SUBSYS_PMC

Definition

```
#define PM_SUBSYS_PMC(0x1c000001U)
```

Description

Data Structure Index

The following is a list of data structures:

- XPm_DeviceStatus
- XPm_Master
- XPm_NodeStatus
- XPm_Notifier
- XPm_Proc
- pm_acknowledge
- pm_init_suspend

pm_acknowledge

Declaration

```
typedef struct
{
  u8 received,
  u32 node,
  XStatus status,
  u32 opp,
  bool received,
  enum XPmNodeId node
} pm_acknowledge;
```

Table 138: **Structure pm_acknowledge member description**

Member	Description
received	Has acknowledge argument been received?
node	Node argument about which the acknowledge is
status	Acknowledged status

Table 138: **Structure pm_acknowledge member description** *(cont'd)*

Member	Description
opp	Operating point of node in question
received	Has acknowledge argument been received?
node	Node argument about which the acknowledge is

pm_init_suspend

Declaration

```
typedef struct
{
  u8 received,
  enum XPmSuspendReason reason,
  u32 latency,
  u32 state,
  u32 timeout,
  bool received
} pm_init_suspend;
```

Table 139: **Structure pm_init_suspend member description**

Member	Description
received	Has init suspend callback been received/handled
reason	Reason of initializing suspend
latency	Maximum allowed latency
state	Targeted sleep/suspend state
timeout	Period of time the client has to response
received	Has init suspend callback been received/handled

XPm_DeviceStatus

Contains the device status information.

Declaration

```
typedef struct
{
  u32 Status,
  u32 Requirement,
  u32 Usage
} XPm_DeviceStatus;
```

Table 140: **Structure XPm_DeviceStatus member description**

Member	Description
Status	Device power state
Requirement	Requirements placed on the device by the caller
Usage	Usage info (which subsystem is using the device)

XPm_Master

`XPm_Master` - Master structure

Declaration

```
typedef struct
{
  enum XPmNodeId node_id,
  const u32 pwrctl,
  const u32 pwrdn_mask,
  XIpiPsu * ipi
} XPm_Master;
```

Table 141: **Structure XPm_Master member description**

Member	Description
node_id	Node ID
pwrctl	
pwrdn_mask	< Power Control Register Address Power Down Mask
ipi	IPI Instance

XPm_NodeStatus

`XPm_NodeStatus` - struct containing node status information

Declaration

```
typedef struct
{
  u32 status,
  u32 requirements,
  u32 usage
} XPm_NodeStatus;
```

Table 142: **Structure XPm_NodeStatus member description**

Member	Description
status	Node power state
requirements	Current requirements asserted on the node (slaves only)

Table 142: **Structure XPm_NodeStatus member description** *(cont'd)*

Member	Description
usage	Usage information (which master is currently using the slave)

XPm_Notifier

XPm_Notifier - Notifier structure registered with a callback by app

Declaration

```
typedef struct
{
  void(*const callback)(struct XPm_Ntfier *const notifier),
  const u32 node,
  enum XPmNotifyEvent event,
  u32 flags,
  u32 oppoint,
  u32 received,
  struct XPm_Ntfier * next,
  enum XPmNodeId node
} XPm_Notifier;
```

Table 143: **Structure XPm_Notifier member description**

Member	Description
callback	Custom callback handler to be called when the notification is received. The custom handler would execute from interrupt context, it shall return quickly and must not block! (enables event-driven notifications)
node	Node argument (the node to receive notifications about)
event	Event argument (the event type to receive notifications about)
flags	Flags
oppoint	Operating point of node in question. Contains the value updated when the last event notification is received. User shall not modify this value while the notifier is registered.
received	How many times the notification has been received - to be used by application (enables polling). User shall not modify this value while the notifier is registered.
next	Pointer to next notifier in linked list. Must not be modified while the notifier is registered. User shall not ever modify this value.
node	Node argument (the node to receive notifications about)

XPm_Proc

XPm_Proc - Processor structure

Declaration

```
typedef struct
{
  const u32 DevId,
  const u32 PwrCtrl,
  const u32 PwrDwnMask,
  XIpiPsu * Ipi
} XPm_Proc;
```

Table 144: **Structure XPm_Proc member description**

Member	Description
DevId	Device ID
PwrCtrl	Power Control Register Address
PwrDwnMask	Power Down Mask
Ipi	IPI Instance

Error Status

This section lists the Power management specific return error statuses.

Definitions

Define XST_PM_INTERNAL

Definition

```
#define XST_PM_INTERNAL2000L
```

Description

An internal error occurred while performing the requested operation

Define XST_PM_CONFLICT

Definition

```
#define XST_PM_CONFLICT2001L
```

Description

Conflicting requirements have been asserted when more than one processing cluster is using the same PM slave

Define XST_PM_NO_ACCESS

Definition

```
#define XST_PM_NO_ACCESS2002L
```

Description

The processing cluster does not have access to the requested node or operation

Define XST_PM_INVALID_NODE

Definition

```
#define XST_PM_INVALID_NODE2003L
```

Description

The API function does not apply to the node passed as argument

Define XST_PM_DOUBLE_REQ

Definition

```
#define XST_PM_DOUBLE_REQ2004L
```

Description

A processing cluster has already been assigned access to a PM slave and has issued a duplicate request for that PM slave

Define XST_PM_ABORT_SUSPEND

Definition

```
#define XST_PM_ABORT_SUSPEND2005L
```

Description

The target processing cluster has aborted suspend

Define XST_PM_TIMEOUT

Definition

```
#define XST_PM_TIMEOUT2006L
```

Description

A timeout occurred while performing the requested operation

Define XST_PM_NODE_USED

Definition

```
#define XST_PM_NODE_USED2007L
```

Description

Slave request cannot be granted since node is non-shareable and used

Reset Nodes

Definitions

Define PM_RST_PMC_POR

Definition

```
#define PM_RST_PMC_POR(0xc30c001U)
```

Description

Define PM_RST_PMC

Definition

```
#define PM_RST_PMC(0xc410002U)
```

Description

Define PM_RST_PS_POR

Definition

```
#define PM_RST_PS_POR(0xc30c003U)
```

Description

Define PM_RST_PL_POR

Definition

```
#define PM_RST_PL_POR(0xc30c004U)
```

Description

Define PM_RST_NOC_POR

Definition

```
#define PM_RST_NOC_POR(0xc30c005U)
```

Description

Define PM_RST_FPD_POR

Definition

```
#define PM_RST_FPD_POR(0xc30c006U)
```

Description

Define PM_RST_ACPU_0_POR

Definition

```
#define PM_RST_ACPU_0_POR(0xc30c007U)
```

Description

Define PM_RST_ACPU_1_POR

Definition

```
#define PM_RST_ACPU_1_POR(0xc30c008U)
```

Description

Define PM_RST_OCM2_POR

Definition

```
#define PM_RST_OCM2_POR(0xc30c009U)
```

Description

Define PM_RST_PS_SRST

Definition

```
#define PM_RST_PS_SRST(0xc41000aU)
```

Description

Define PM_RST_PL_SRST

Definition

```
#define PM_RST_PL_SRST(0xc41000bU)
```

Description

Define PM_RST_NOC

Definition

```
#define PM_RST_NOC(0xc41000cU)
```

Description

Define PM_RST_NPI

Definition

```
#define PM_RST_NPI(0xc41000dU)
```

Description

Define PM_RST_SYS_RST_1

Definition

```
#define PM_RST_SYS_RST_1(0xc41000eU)
```

Description

Define PM_RST_SYS_RST_2

Definition

```
#define PM_RST_SYS_RST_2(0xc41000fU)
```

Description

Define PM_RST_SYS_RST_3

Definition

```
#define PM_RST_SYS_RST_3(0xc410010U)
```

Description

Define PM_RST_FPD

Definition

```
#define PM_RST_FPD(0xc410011U)
```

Description

Define PM_RST_PL0

Definition

```
#define PM_RST_PL0(0xc410012U)
```

Description

Define PM_RST_PL1

Definition

```
#define PM_RST_PL1(0xc410013U)
```

Description

Define PM_RST_PL2

Definition

```
#define PM_RST_PL2(0xc410014U)
```

Description

Define PM_RST_PL3

Definition

```
#define PM_RST_PL3(0xc410015U)
```

Description

Define PM_RST_APU

Definition

```
#define PM_RST_APU(0xc410016U)
```

Description

Define PM_RST_ACPU_0

Definition

```
#define PM_RST_ACPU_0(0xc410017U)
```

Description

Define PM_RST_ACPU_1

Definition

```
#define PM_RST_ACPU_1(0xc410018U)
```

Description

Define PM_RST_ACPU_L2

Definition

```
#define PM_RST_ACPU_L2(0xc410019U)
```

Description

Define PM_RST_ACPU_GIC

Definition

```
#define PM_RST_ACPU_GIC(0xc41001aU)
```

Description

Define PM_RST_RPU_ISLAND

Definition

```
#define PM_RST_RPU_ISLAND(0xc41001bU)
```

Description

Define PM_RST_RPU_AMBA

Definition

```
#define PM_RST_RPU_AMBA(0xc41001cU)
```

Description

Define PM_RST_R5_0

Definition

```
#define PM_RST_R5_0(0xc41001dU)
```

Description

Define PM_RST_R5_1

Definition

```
#define PM_RST_R5_1(0xc41001eU)
```

Description

Define PM_RST_SYSMON_PMC_SEQ_RST

Definition

```
#define PM_RST_SYSMON_PMC_SEQ_RST(0xc41001fU)
```

Description

Define PM_RST_SYSMON_PMC_CFG_RST

Definition

```
#define PM_RST_SYSMON_PMC_CFG_RST(0xc410020U)
```

Description

Define PM_RST_SYSMON_FPD_CFG_RST

Definition

```
#define PM_RST_SYSMON_FPD_CFG_RST(0xc410021U)
```

Description

Define PM_RST_SYSMON_FPD_SEQ_RST

Definition

```
#define PM_RST_SYSMON_FPD_SEQ_RST(0xc410022U)
```

Description

Define PM_RST_SYSMON_LPD

Definition

```
#define PM_RST_SYSMON_LPD(0xc410023U)
```

Description

Define PM_RST_PDMA_RST1

Definition

```
#define PM_RST_PDMA_RST1(0xc410024U)
```

Description

Define PM_RST_PDMA_RST0

Definition

```
#define PM_RST_PDMA_RST0(0xc410025U)
```

Description

Define PM_RST_ADMA

Definition

```
#define PM_RST_ADMA(0xc410026U)
```

Description

Define PM_RST_TIMESTAMP

Definition

```
#define PM_RST_TIMESTAMP(0xc410027U)
```

Description

Define PM_RST_OCM

Definition

```
#define PM_RST_OCM(0xc410028U)
```

Description

Define PM_RST_OCM2_RST

Definition

```
#define PM_RST_OCM2_RST(0xc410029U)
```

Description

Define PM_RST_IPI

Definition

```
#define PM_RST_IPI(0xc41002aU)
```

Description

Define PM_RST_SBI

Definition

```
#define PM_RST_SBI(0xc41002bU)
```

Description

Define PM_RST_LPD

Definition

```
#define PM_RST_LPD(0xc41002cU)
```

Description

Define PM_RST_QSPI

Definition

```
#define PM_RST_QSPI(0xc10402dU)
```

Description

Define PM_RST_OSPI

Definition

```
#define PM_RST_OSPI(0xc10402eU)
```

Description

Define PM_RST_SDIO_0

Definition

```
#define PM_RST_SDIO_0(0xc10402fU)
```

Description

Define PM_RST_SDIO_1

Definition

```
#define PM_RST_SDIO_1(0xc104030U)
```

Description

Define PM_RST_I2C_PMC

Definition

```
#define PM_RST_I2C_PMC(0xc104031U)
```

Description

Define PM_RST_GPIO_PMC

Definition

```
#define PM_RST_GPIO_PMC(0xc104032U)
```

Description

Define PM_RST_GEM_0

Definition

```
#define PM_RST_GEM_0(0xc104033U)
```

Description

Define PM_RST_GEM_1

Definition

```
#define PM_RST_GEM_1(0xc104034U)
```

Description

Define PM_RST_SPARE

Definition

```
#define PM_RST_SPARE(0xc104035U)
```

Description

Define PM_RST_USB_0

Definition

```
#define PM_RST_USB_0(0xc104036U)
```

Description

Define PM_RST_UART_0

Definition

```
#define PM_RST_UART_0(0xc104037U)
```

Description

Define PM_RST_UART_1

Definition

```
#define PM_RST_UART_1(0xc104038U)
```

Description

Define PM_RST_SPI_0

Definition

```
#define PM_RST_SPI_0(0xc104039U)
```

Description

Define PM_RST_SPI_1

Definition

```
#define PM_RST_SPI_1(0xc10403aU)
```

Description

Define PM_RST_CAN_FD_0

Definition

```
#define PM_RST_CAN_FD_0(0xc10403bU)
```

Description

Define PM_RST_CAN_FD_1

Definition

```
#define PM_RST_CAN_FD_1(0xc10403cU)
```

Description

Define PM_RST_I2C_0

Definition

```
#define PM_RST_I2C_0(0xc10403dU)
```

Description

Define PM_RST_I2C_1

Definition

```
#define PM_RST_I2C_1(0xc10403eU)
```

Description

Define PM_RST_GPIO_LPD

Definition

```
#define PM_RST_GPIO_LPD(0xc10403fU)
```

Description

Define PM_RST_TTC_0

Definition

```
#define PM_RST_TTC_0(0xc104040U)
```

Description

Define PM_RST_TTC_1

Definition

```
#define PM_RST_TTC_1(0xc104041U)
```

Description

Define PM_RST_TTC_2

Definition

```
#define PM_RST_TTC_2(0xc104042U)
```

Description

Define PM_RST_TTC_3

Definition

```
#define PM_RST_TTC_3(0xc104043U)
```

Description

Define PM_RST_SWDT_FPD

Definition

```
#define PM_RST_SWDT_FPD(0xc104044U)
```

Description

Define PM_RST_SWDT_LPD

Definition

```
#define PM_RST_SWDT_LPD(0xc104045U)
```

Description

Define PM_RST_USB

Definition

```
#define PM_RST_USB(0xc104046U)
```

Description

Define PM_RST_DPC

Definition

```
#define PM_RST_DPC(0xc208047U)
```

Description

Define PM_RST_PMCDBG

Definition

```
#define PM_RST_PMCDBG(0xc208048U)
```

Description

Define PM_RST_DBG_TRACE

Definition

```
#define PM_RST_DBG_TRACE(0xc208049U)
```

Description

Define PM_RST_DBG_FPD

Definition

```
#define PM_RST_DBG_FPD(0xc20804aU)
```

Description

Define PM_RST_DBG_TSTMP

Definition

```
#define PM_RST_DBG_TSTMP(0xc20804bU)
```

Description

Define PM_RST_RPU0_DBG

Definition

```
#define PM_RST_RPU0_DBG(0xc20804cU)
```

Description

Define PM_RST_RPU1_DBG

Definition

```
#define PM_RST_RPU1_DBG(0xc20804dU)
```

Description

Define PM_RST_HSDP

Definition

```
#define PM_RST_HSDP(0xc20804eU)
```

Description

Define PM_RST_DBG_LPD

Definition

```
#define PM_RST_DBG_LPD(0xc20804fU)
```

Description

Define PM_RST_CPM_POR

Definition

```
#define PM_RST_CPM_POR(0xc30c050U)
```

Description

Define PM_RST_CPM

Definition

```
#define PM_RST_CPM(0xc410051U)
```

Description

Define PM_RST_CPMDBG

Definition

```
#define PM_RST_CPMDBG(0xc208052U)
```

Description

Define PM_RST_PCIE_CFG

Definition

```
#define PM_RST_PCIE_CFG(0xc410053U)
```

Description

Define PM_RST_PCIE_CORE0

Definition

```
#define PM_RST_PCIE_CORE0(0xc410054U)
```

Description

Define PM_RST_PCIE_CORE1

Definition

```
#define PM_RST_PCIE_CORE1(0xc410055U)
```

Description

Define PM_RST_PCIE_DMA

Definition

```
#define PM_RST_PCIE_DMA(0xc410056U)
```

Description

Define PM_RST_CMN

Definition

```
#define PM_RST_CMN(0xc410057U)
```

Description

Define PM_RST_L2_0

Definition

```
#define PM_RST_L2_0(0xc410058U)
```

Description

Define PM_RST_L2_1

Definition

```
#define PM_RST_L2_1(0xc410059U)
```

Description

Define PM_RST_ADDR_REMAP

Definition

```
#define PM_RST_ADDR_REMAP(0xc41005aU)
```

Description

Define PM_RST_CPI0

Definition

```
#define PM_RST_CPI0(0xc41005bU)
```

Description

Define PM_RST_CPI1

Definition

```
#define PM_RST_CPI1(0xc41005cU)
```

Description

Define PM_RST_AIE_ARRAY

Definition

```
#define PM_RST_AIE_ARRAY(0xc10405eU)
```

Description

Define PM_RST_AIE_SHIM

Definition

```
#define PM_RST_AIE_SHIM(0xc10405fU)
```

Description

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Documentation Navigator and Design Hubs

Documentation Navigator (DocNav) provides access to documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the IDE, select **Help → Documentation and Tutorials**.
- On Windows, select **Start → All Programs → Xilinx Design Tools → DocNav**.
- At the Linux command prompt, enter `docnav`.

Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- On the website, see the Design Hubs page.

Note: For more information on DocNav, see the Documentation Navigator page on the website.

Please Read: Important Legal Notices

AUTOMOTIVE APPLICATIONS DISCLAIMER

Copyright