Xilinx Standalone Library Documentation

XilPM Library v3.5

UG1225 (v2021.2) October 13, 2021

XILINX®

Table of Contents

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	XPm_GetBootStatus	void
void	XPm_SuspendFinalize	void
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
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

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

Type	Name	Arguments
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	u32 type u32 subtype
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
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

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

Type	Name	Arguments
XStatus	XPm_GetOpCharacteristic	const enum XPmNodeId node const enum XPmOpCharType type u32 *const result
XStatus	XPm_ResetAssert	const enum XPmReset reset const enum XPmResetAction resetaction
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
XStatus	XPm_ClockSetDivider	const enum XPmClock clk const u32 divider
XStatus	XPm_ClockGetOneDivider	const enum XPmClock clk u32 *const divider const u32 divId

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

Type	Name	Arguments
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 const enum XPmApiId api
XStatus	XPm_PinCtrlRequest	const u32 pin
XStatus	XPm_PinCtrlRelease	const u32 pin
XStatus	XPm_PinCtrlSetFunction	const u32 pin const enum XPmPinFn fn

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

Type	Name	Arguments
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
XStatus	XPm_DevIoctl	const u32 deviceId const pm_ioctl_id ioctlId const u32 arg1 const u32 arg2 u32 *const response

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

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 3: **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 4: **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 5: **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 6: **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 7: **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 8: **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 9: **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 10: **XPm_SystemShutdown Arguments**

Type	Name	Description
u32	type	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
u32	subtype	Restart subtype (SYSTEM or PS_ONLY or SUBSYSTEM)

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 11: **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 12: **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 13: **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 14: **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 15: **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 15: **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 16: **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 17: **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 18: **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 19: **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 20: **XPm_GetOpCharacteristic Arguments**

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

Table 20: **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 21: **XPm_ResetAssert Arguments**

Type	Name	Description
const enum XPmReset	reset	ID of the reset line
const enum XPmResetAction	resetaction	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 22: **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 23: **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 24: **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 25: **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 26: **XPm_MmioRead Arguments**

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

Table 26: **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 27: **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 28: **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 29: **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 30: **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 31: **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 32: **XPm_ClockGetOneDivider Arguments**

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

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 33: **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 34: **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 35: **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 36: **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 37: **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 38: **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 39: **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 40: **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 41: **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 42: **XPm_PinCtrlAction Arguments**

Type	Name	Description
const u32	pin	PIN identifier (index from range 0-77)
const enum `XPmApiId`	api	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 43: **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 44: **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 45: **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 46: **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 47: **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 48: **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

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 49: **XPm_DevIoctl Arguments**

Type	Name	Description
const u32	deviceId	ID of the device

Table 49: **XPm_DevIoctl Arguments** *(cont'd)*

Type	Name	Description
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

Enumerations

Enumeration XPmApiId

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

Table 50: **Enumeration XPmApiId Values**

Value	Description
PM_GET_API_VERSION	0x1
PM_SET_CONFIGURATION	0x2
PM_GET_NODE_STATUS	0x3
PM_GET_OP_CHARACTERISTIC	0x4
PM_REGISTER_NOTIFIER	0x5
PM_REQUEST_SUSPEND	0x6
PM_SELF_SUSPEND	0x7
PM_FORCE_POWERDOWN	0x8
PM_ABORT_SUSPEND	0x9
PM_REQUEST_WAKEUP	0xA
PM_SET_WAKEUP_SOURCE	0xB
PM_SYSTEM_SHUTDOWN	0xC
PM_REQUEST_NODE	0xD
PM_RELEASE_NODE	0xE
PM_SET_REQUIREMENT	0xF
PM_SET_MAX_LATENCY	0x10
PM_RESET_ASSERT	0x11
PM_RESET_GET_STATUS	0x12
PM_MMIO_WRITE	0x13

Table 50: **Enumeration XPmApiId Values** *(cont'd)*

Value	Description
PM_MMIO_READ	0x14
PM_INIT_FINALIZE	0x15
PM_FPGA_LOAD	0x16
PM_FPGA_GET_STATUS	0x17
PM_GET_CHIPID	0x18
PM_SECURE_SHA	0x1A
PM_SECURE_RSA	0x1B
PM_PINCTRL_REQUEST	0x1C
PM_PINCTRL_RELEASE	0x1D
PM_PINCTRL_GET_FUNCTION	0x1E
PM_PINCTRL_SET_FUNCTION	0x1F
PM_PINCTRL_CONFIG_PARAM_GET	0x20
PM_PINCTRL_CONFIG_PARAM_SET	0x21
PM_IOCTL	0x22
PM_QUERY_DATA	0x23
PM_CLOCK_ENABLE	0x24
PM_CLOCK_DISABLE	0x25
PM_CLOCK_GETSTATE	0x26
PM_CLOCK_SETDIVIDER	0x27
PM_CLOCK_GETDIVIDER	0x28
PM_CLOCK_SETRATE	0x29
PM_CLOCK_GETRATE	0x2A
PM_CLOCK_SETPARENT	0x2B
PM_CLOCK_GETPARENT	0x2C
PM_SECURE_IMAGE	0x2D
PM_FPGA_READ	0x2E
PM_SECURE_AES	0x2F
PM_PLL_SET_PARAMETER	0x30
PM_PLL_GET_PARAMETER	0x31
PM_PLL_SET_MODE	0x32
PM_PLL_GET_MODE	0x33
PM_REGISTER_ACCESS	0x34
PM_EFUSE_ACCESS	0x35
PM_API_MAX	0x36

Enumeration XPmApiCbId

PM API Callback ID

Table 51: **Enumeration XPmApiCbId Values**

Value	Description
PM_INIT_SUSPEND_CB	Suspend callback
PM_ACKNOWLEDGE_CB	Acknowledge callback
PM_NOTIFY_CB	Notify callback
PM_NOTIFY_STL_NO_OP	Notify STL No OP

Enumeration XPmNodeId

PM Node ID

Table 52: **Enumeration XPmNodeId Values**

Value	Description
NODE_UNKNOWN	0x0
NODE_APU	0x1
NODE_APU_0	0x2
NODE_APU_1	0x3
NODE_APU_2	0x4
NODE_APU_3	0x5
NODE_RPU	0x6
NODE_RPU_0	0x7
NODE_RPU_1	0x8
NODE_PLD	0x9
NODE_FPD	0xA
NODE_OCM_BANK_0	0xB
NODE_OCM_BANK_1	0xC
NODE_OCM_BANK_2	0xD
NODE_OCM_BANK_3	0xE
NODE_TCM_0_A	0xF
NODE_TCM_0_B	0x10
NODE_TCM_1_A	0x11
NODE_TCM_1_B	0x12
NODE_L2	0x13
NODE_GPU_PP_0	0x14
NODE_GPU_PP_1	0x15
NODE_USB_0	0x16
NODE_USB_1	0x17
NODE_TTC_0	0x18
NODE_TTC_1	0x19
NODE_TTC_2	0x1A

Table 52: **Enumeration XPmNodeId Values** *(cont'd)*

Value	Description
NODE_TTC_3	0x1B
NODE_SATA	0x1C
NODE_ETH_0	0x1D
NODE_ETH_1	0x1E
NODE_ETH_2	0x1F
NODE_ETH_3	0x20
NODE_UART_0	0x21
NODE_UART_1	0x22
NODE_SPI_0	0x23
NODE_SPI_1	0x24
NODE_I2C_0	0x25
NODE_I2C_1	0x26
NODE_SD_0	0x27
NODE_SD_1	0x28
NODE_DP	0x29
NODE_GDMA	0x2A
NODE_ADMA	0x2B
NODE_NAND	0x2C
NODE_QSPI	0x2D
NODE_GPIO	0x2E
NODE_CAN_0	0x2F
NODE_CAN_1	0x30
NODE_EXTERN	0x31
NODE_APLL	0x32
NODE_VPLL	0x33
NODE_DPLL	0x34
NODE_RPLL	0x35
NODE_IOPLL	0x36
NODE_DDR	0x37
NODE_IPI_APU	0x38
NODE_IPI_RPU_0	0x39
NODE_GPU	0x3A
NODE_PCIE	0x3B
NODE_PCAP	0x3C
NODE_RTC	0x3D
NODE_LPD	0x3E
NODE_VCU	0x3F
NODE_IPI_RPU_1	0x40
NODE_IPI_PL_0	0x41

Table 52: **Enumeration XPmNodeId Values** *(cont'd)*

Value	Description
NODE_IPI_PL_1	0x42
NODE_IPI_PL_2	0x43
NODE_IPI_PL_3	0x44
NODE_PL	0x45
NODE_ID_MAX	0x46

Enumeration XPmRequestAck

PM Acknowledge Request

Table 53: **Enumeration XPmRequestAck Values**

Value	Description
REQUEST_ACK_NO	No Ack
REQUEST_ACK_BLOCKING	Blocking Ack
REQUEST_ACK_NON_BLOCKING	Non blocking Ack
REQUEST_ACK_CB_CERROR	Callback Error

Enumeration XPmAbortReason

PM Abort Reasons

Table 54: **Enumeration XPmAbortReason Values**

Value	Description
ABORT_REASON_WKUP_EVENT	Wakeup Event
ABORT_REASON_PU_BUSY	Processor Busy
ABORT_REASON_NO_PWRDN	No Powerdown
ABORT_REASON_UNKNOWN	Unknown Reason

Enumeration XPmSuspendReason

PM Suspend Reasons

Table 55: **Enumeration XPmSuspendReason Values**

Value	Description
SUSPEND_REASON_PU_REQ	Processor request
SUSPEND_REASON_ALERT	Alert
SUSPEND_REASON_SYS_SHUTDOWN	System shutdown

Enumeration XPmRamState

PM RAM States

Table 56: **Enumeration XPmRamState Values**

Value	Description
PM_RAM_STATE_OFF	Off
PM_RAM_STATE_RETENTION	Retention
PM_RAM_STATE_ON	On

Enumeration XPmOpCharType

PM Operating Characteristic

Table 57: **Enumeration XPmOpCharType Values**

Value	Description
PM_OPCHAR_TYPE_POWER	Operating characteristic ID power
PM_OPCHAR_TYPE_TEMP	Operating characteristic ID temperature
PM_OPCHAR_TYPE_LATENCY	Operating characteristic ID latency

Definitions

Define PM_VERSION_MAJOR

Definition

```
#define PM_VERSION_MAJOR1
```

Description

PM Version Number macros

Define PM_VERSION_MINOR

Definition

```
#define PM_VERSION_MINOR1
```

Description

PM Version Number macros

Define PM_VERSION

Definition

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

Description

PM Version Number macros

Define PM_CAP_ACCESS

Definition

```
#define PM_CAP_ACCESS0x1U
```

Description

Capabilities for RAM

Define PM_CAP_CONTEXT

Definition

```
#define PM_CAP_CONTEXT0x2U
```

Description

Capabilities for RAM

Define PM_CAP_WAKEUP

Definition

```
#define PM_CAP_WAKEUP0x4U
```

Description

Capabilities for RAM

Define NODE_STATE_OFF

Definition

```
#define NODE_STATE_OFF0
```

Description

Node State

Define NODE_STATE_ON

Definition

```
#define NODE_STATE_ON1
```

Description

Node State

Define PROC_STATE_FORCEDOFF

Definition

```
#define PROC_STATE_FORCEDOFF0
```

Description

Processor's state

Define PROC_STATE_ACTIVE

Definition

```
#define PROC_STATE_ACTIVE1
```

Description

Processor's state

Define PROC_STATE_SLEEP

Definition

```
#define PROC_STATE_SLEEP2
```

Description

Processor's state

Define PROC_STATE_SUSPENDING

Definition

```
#define PROC_STATE_SUSPENDING3
```

Description

Processor's state

Define MAX_LATENCY

Definition

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

Description

Maximum Latency/QOS

Define MAX_QOS

Definition

```
#define MAX_QOS100U
```

Description

Maximum Latency/QOS

Define PMF_SHUTDOWN_TYPE_SHUTDOWN

Definition

```
#define PMF_SHUTDOWN_TYPE_SHUTDOWN0U
```

Description

System shutdown/Restart

Define PMF_SHUTDOWN_TYPE_RESET

Definition

```
#define PMF_SHUTDOWN_TYPE_RESET1U
```

Description

System shutdown/Restart

Define PMF_SHUTDOWN_SUBTYPE_SUBSYSTEM

Definition

```
#define PMF_SHUTDOWN_SUBTYPE_SUBSYSTEM0U
```

Description

System shutdown/Restart

Define PMF_SHUTDOWN_SUBTYPE_PS_ONLY

Definition

```
#define PMF_SHUTDOWN_SUBTYPE_PS_ONLY1U
```

Description

System shutdown/Restart

Define PMF_SHUTDOWN_SUBTYPE_SYSTEM

Definition

```
#define PMF_SHUTDOWN_SUBTYPE_SYSTEM2U
```

Description

System shutdown/Restart

Error Status

This section lists the Power management specific return error statuses.

Enumerations

Enumeration XPmBootStatus

Boot Status

Table 58: **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

Enumeration XPmResetAction

PM Reset Action types

Table 59: **Enumeration XPmResetAction Values**

Value	Description
XILPM_RESET_ACTION_RELEASE	Reset action release
XILPM_RESET_ACTION_ASSERT	Reset action assert
XILPM_RESET_ACTION_PULSE	Reset action pulse

Enumeration XPmReset

PM Reset Line IDs

Table 60: **Enumeration XPmReset Values**

Value	Description
XILPM_RESET_PCIE_CFG	Reset ID PCIE_CFG
XILPM_RESET_PCIE_BRIDGE	Reset ID PCIE_BRIDGE
XILPM_RESET_PCIE_CTRL	Reset ID PCIE_CTRL
XILPM_RESET_DP	Reset ID DP
XILPM_RESET_SWDT_CRF	Reset ID SWDT_CRF
XILPM_RESET_AFI_FM5	Reset ID AFI_FM5
XILPM_RESET_AFI_FM4	Reset ID AFI_FM4
XILPM_RESET_AFI_FM3	Reset ID AFI_FM3
XILPM_RESET_AFI_FM2	Reset ID AFI_FM2
XILPM_RESET_AFI_FM1	Reset ID AFI_FM1
XILPM_RESET_AFI_FM0	Reset ID AFI_FM0
XILPM_RESET_GDMA	Reset ID GDMA
XILPM_RESET_GPU_PP1	Reset ID GPU_PP1
XILPM_RESET_GPU_PP0	Reset ID GPU_PP0
XILPM_RESET_GPU	Reset ID GPU
XILPM_RESET_GT	Reset ID GT
XILPM_RESET_SATA	Reset ID SATA
XILPM_RESET_ACPU3_PWRON	Reset ID ACPU3_PWRON
XILPM_RESET_ACPU2_PWRON	Reset ID ACPU2_PWRON
XILPM_RESET_ACPU1_PWRON	Reset ID ACPU1_PWRON
XILPM_RESET_ACPU0_PWRON	Reset ID ACPU0_PWRON
XILPM_RESET_APU_L2	Reset ID APU_L2
XILPM_RESET_ACPU3	Reset ID ACPU3
XILPM_RESET_ACPU2	Reset ID ACPU2
XILPM_RESET_ACPU1	Reset ID ACPU1
XILPM_RESET_ACPU0	Reset ID ACPU0
XILPM_RESET_DDR	Reset ID DDR
XILPM_RESET_APM_FPD	Reset ID APM_FPD
XILPM_RESET_SOFT	Reset ID SOFT
XILPM_RESET_GEM0	Reset ID GEM0
XILPM_RESET_GEM1	Reset ID GEM1
XILPM_RESET_GEM2	Reset ID GEM2
XILPM_RESET_GEM3	Reset ID GEM3
XILPM_RESET_QSPI	Reset ID QSPI
XILPM_RESET_UART0	Reset ID UART0
XILPM_RESET_UART1	Reset ID UART1
XILPM_RESET_SPI0	Reset ID SPI0
XILPM_RESET_SPI1	Reset ID SPI1
XILPM_RESET_SDIO0	Reset ID SDIO0

Table 60: **Enumeration XPmReset Values** *(cont'd)*

Value	Description
XILPM_RESET_SDIO1	Reset ID SDIO1
XILPM_RESET_CAN0	Reset ID CAN0
XILPM_RESET_CAN1	Reset ID CAN1
XILPM_RESET_I2C0	Reset ID I2C0
XILPM_RESET_I2C1	Reset ID I2C1
XILPM_RESET_TTC0	Reset ID TTC0
XILPM_RESET_TTC1	Reset ID TTC1
XILPM_RESET_TTC2	Reset ID TTC2
XILPM_RESET_TTC3	Reset ID TTC3
XILPM_RESET_SWDT_CRL	Reset ID SWDT_CRL
XILPM_RESET_NAND	Reset ID NAND
XILPM_RESET_ADMA	Reset ID ADMA
XILPM_RESET_GPIO	Reset ID GPIO
XILPM_RESET_IOU_CC	Reset ID IOU_CC
XILPM_RESET_TIMESTAMP	Reset ID TIMESTAMP
XILPM_RESET_RPU_R50	Reset ID RPU_R50
XILPM_RESET_RPU_R51	Reset ID RPU_R51
XILPM_RESET_RPU_AMBA	Reset ID RPU_AMBA
XILPM_RESET_OCM	Reset ID OCM
XILPM_RESET_RPU_PGE	Reset ID RPU_PGE
XILPM_RESET_USB0_CORERESET	Reset ID USB0_CORERESE
XILPM_RESET_USB1_CORERESET	Reset ID USB1_CORERESE
XILPM_RESET_USB0_HIBERRESET	Reset ID USB0_HIBERRES
XILPM_RESET_USB1_HIBERRESET	Reset ID USB1_HIBERRES
XILPM_RESET_USB0_APB	Reset ID USB0_APB
XILPM_RESET_USB1_APB	Reset ID USB1_APB
XILPM_RESET_IPI	Reset ID IPI
XILPM_RESET_APM_LPD	Reset ID APM_LPD
XILPM_RESET_RTC	Reset ID RTC
XILPM_RESET_SYSMON	Reset ID SYSMON
XILPM_RESET_AFI_FM6	Reset ID AFI_FM6
XILPM_RESET_LPD_SWDT	Reset ID LPD_SWDT
XILPM_RESET_FPD	Reset ID FPD
XILPM_RESET_RPU_DBG1	Reset ID RPU_DBG1
XILPM_RESET_RPU_DBG0	Reset ID RPU_DBG0
XILPM_RESET_DBG_LPD	Reset ID DBG_LPD
XILPM_RESET_DBG_FPD	Reset ID DBG_FPD
XILPM_RESET_APLL	Reset ID APLL
XILPM_RESET_DPLL	Reset ID DPLL

Table 60: **Enumeration XPmReset Values** *(cont'd)*

Value	Description
XILPM_RESET_VPLL	Reset ID VPLL
XILPM_RESET_IOPLL	Reset ID IOPLL
XILPM_RESET_RPLL	Reset ID RPLL
XILPM_RESET_GPO3_PL_0	Reset ID GPO3_PL_0
XILPM_RESET_GPO3_PL_1	Reset ID GPO3_PL_1
XILPM_RESET_GPO3_PL_2	Reset ID GPO3_PL_2
XILPM_RESET_GPO3_PL_3	Reset ID GPO3_PL_3
XILPM_RESET_GPO3_PL_4	Reset ID GPO3_PL_4
XILPM_RESET_GPO3_PL_5	Reset ID GPO3_PL_5
XILPM_RESET_GPO3_PL_6	Reset ID GPO3_PL_6
XILPM_RESET_GPO3_PL_7	Reset ID GPO3_PL_7
XILPM_RESET_GPO3_PL_8	Reset ID GPO3_PL_8
XILPM_RESET_GPO3_PL_9	Reset ID GPO3_PL_9
XILPM_RESET_GPO3_PL_10	Reset ID GPO3_PL_10
XILPM_RESET_GPO3_PL_11	Reset ID GPO3_PL_11
XILPM_RESET_GPO3_PL_12	Reset ID GPO3_PL_12
XILPM_RESET_GPO3_PL_13	Reset ID GPO3_PL_13
XILPM_RESET_GPO3_PL_14	Reset ID GPO3_PL_14
XILPM_RESET_GPO3_PL_15	Reset ID GPO3_PL_15
XILPM_RESET_GPO3_PL_16	Reset ID GPO3_PL_16
XILPM_RESET_GPO3_PL_17	Reset ID GPO3_PL_17
XILPM_RESET_GPO3_PL_18	Reset ID GPO3_PL_18
XILPM_RESET_GPO3_PL_19	Reset ID GPO3_PL_19
XILPM_RESET_GPO3_PL_20	Reset ID GPO3_PL_20
XILPM_RESET_GPO3_PL_21	Reset ID GPO3_PL_21
XILPM_RESET_GPO3_PL_22	Reset ID GPO3_PL_22
XILPM_RESET_GPO3_PL_23	Reset ID GPO3_PL_23
XILPM_RESET_GPO3_PL_24	Reset ID GPO3_PL_24
XILPM_RESET_GPO3_PL_25	Reset ID GPO3_PL_25
XILPM_RESET_GPO3_PL_26	Reset ID GPO3_PL_26
XILPM_RESET_GPO3_PL_27	Reset ID GPO3_PL_27
XILPM_RESET_GPO3_PL_28	Reset ID GPO3_PL_28
XILPM_RESET_GPO3_PL_29	Reset ID GPO3_PL_29
XILPM_RESET_GPO3_PL_30	Reset ID GPO3_PL_30
XILPM_RESET_GPO3_PL_31	Reset ID GPO3_PL_31
XILPM_RESET_RPU_LS	Reset ID RPU_LS
XILPM_RESET_PS_ONLY	Reset ID PS_ONLY
XILPM_RESET_PL	Reset ID PL
XILPM_RESET_GPIO5_EMIO_92	Reset ID GPIO5_EMIO_92

Table 60: **Enumeration XPmReset Values** *(cont'd)*

Value	Description
XILPM_RESET_GPIO5_EMIO_93	Reset ID GPIO5_EMIO_93
XILPM_RESET_GPIO5_EMIO_94	Reset ID GPIO5_EMIO_94
XILPM_RESET_GPIO5_EMIO_95	Reset ID GPIO5_EMIO_95

Enumeration XPmNotifyEvent

PM Notify Events Enum

Table 61: **Enumeration XPmNotifyEvent Values**

Value	Description
EVENT_STATE_CHANGE	State change event
EVENT_ZERO_USERS	Zero user event
EVENT_CPU_IDLE_FORCE_PWRDWN	CPU idle event during force power down

Enumeration XPmClock

PM Clock IDs

Table 62: **Enumeration XPmClock Values**

Value	Description
PM_CLOCK_IOPLL	Clock ID IOPLL
PM_CLOCK_RPLL	Clock ID RPLL
PM_CLOCK_APLL	Clock ID APLL
PM_CLOCK_DPLL	Clock ID DPLL
PM_CLOCK_VPLL	Clock ID VPLL
PM_CLOCK_IOPLL_TO_FPD	Clock ID IOPLL_TO_FPD
PM_CLOCK_RPLL_TO_FPD	Clock ID RPLL_TO_FPD
PM_CLOCK_APLL_TO_LPD	Clock ID APLL_TO_LPD
PM_CLOCK_DPLL_TO_LPD	Clock ID DPLL_TO_LPD
PM_CLOCK_VPLL_TO_LPD	Clock ID VPLL_TO_LPD
PM_CLOCK_ACPU	Clock ID ACPU
PM_CLOCK_ACPU_HALF	Clock ID ACPU_HALF
PM_CLOCK_DBG_FPD	Clock ID DBG_FPD
PM_CLOCK_DBG_LPD	Clock ID DBG_LPD
PM_CLOCK_DBG_TRACE	Clock ID DBG_TRACE
PM_CLOCK_DBG_TSTMP	Clock ID DBG_TSTMP
PM_CLOCK_DP_VIDEO_REF	Clock ID DP_VIDEO_REF
PM_CLOCK_DP_AUDIO_REF	Clock ID DP_AUDIO_REF

Table 62: **Enumeration XPmClock Values** *(cont'd)*

Value	Description
PM_CLOCK_DP_STC_REF	Clock ID DP_STC_REF
PM_CLOCK_GDMA_REF	Clock ID GDMA_REF
PM_CLOCK_DPDMA_REF	Clock ID DPDMA_REF
PM_CLOCK_DDR_REF	Clock ID DDR_REF
PM_CLOCK_SATA_REF	Clock ID SATA_REF
PM_CLOCK_PCIE_REF	Clock ID PCIE_REF
PM_CLOCK_GPU_REF	Clock ID GPU_REF
PM_CLOCK_GPU_PP0_REF	Clock ID GPU_PP0_REF
PM_CLOCK_GPU_PP1_REF	Clock ID GPU_PP1_REF
PM_CLOCK_TOPSW_MAIN	Clock ID TOPSW_MAIN
PM_CLOCK_TOPSW_LSBUS	Clock ID TOPSW_LSBUS
PM_CLOCK_GTGREF0_REF	Clock ID GTGREF0_REF
PM_CLOCK_LPD_SWITCH	Clock ID LPD_SWITCH
PM_CLOCK_LPD_LSBUS	Clock ID LPD_LSBUS
PM_CLOCK_USB0_BUS_REF	Clock ID USB0_BUS_REF
PM_CLOCK_USB1_BUS_REF	Clock ID USB1_BUS_REF
PM_CLOCK_USB3_DUAL_REF	Clock ID USB3_DUAL_REF
PM_CLOCK_USB0	Clock ID USB0
PM_CLOCK_USB1	Clock ID USB1
PM_CLOCK_CPU_R5	Clock ID CPU_R5
PM_CLOCK_CPU_R5_CORE	Clock ID CPU_R5_CORE
PM_CLOCK_CSU_SPB	Clock ID CSU_SPB
PM_CLOCK_CSU_PLL	Clock ID CSU_PLL
PM_CLOCK_PCAP	Clock ID PCAP
PM_CLOCK_IOU_SWITCH	Clock ID IOU_SWITCH
PM_CLOCK_GEM_TSU_REF	Clock ID GEM_TSU_REF
PM_CLOCK_GEM_TSU	Clock ID GEM_TSU
PM_CLOCK_GEM0_TX	Clock ID GEM0_TX
PM_CLOCK_GEM1_TX	Clock ID GEM1_TX
PM_CLOCK_GEM2_TX	Clock ID GEM2_TX
PM_CLOCK_GEM3_TX	Clock ID GEM3_TX
PM_CLOCK_GEM0_RX	Clock ID GEM0_RX
PM_CLOCK_GEM1_RX	Clock ID GEM1_RX
PM_CLOCK_GEM2_RX	Clock ID GEM2_RX
PM_CLOCK_GEM3_RX	Clock ID GEM3_RX
PM_CLOCK_QSPI_REF	Clock ID QSPI_REF
PM_CLOCK_SDIO0_REF	Clock ID SDIO0_REF
PM_CLOCK_SDIO1_REF	Clock ID SDIO1_REF
PM_CLOCK_UART0_REF	Clock ID UART0_REF

Table 62: **Enumeration XPmClock Values** *(cont'd)*

Value	Description
PM_CLOCK_UART1_REF	Clock ID UART1_REF
PM_CLOCK_SPI0_REF	Clock ID SPI0_REF
PM_CLOCK_SPI1_REF	Clock ID SPI1_REF
PM_CLOCK_NAND_REF	Clock ID NAND_REF
PM_CLOCK_I2C0_REF	Clock ID I2C0_REF
PM_CLOCK_I2C1_REF	Clock ID I2C1_REF
PM_CLOCK_CAN0_REF	Clock ID CAN0_REF
PM_CLOCK_CAN1_REF	Clock ID CAN1_REF
PM_CLOCK_CAN0	Clock ID CAN0
PM_CLOCK_CAN1	Clock ID CAN1
PM_CLOCK_DLL_REF	Clock ID DLL_REF
PM_CLOCK_ADMA_REF	Clock ID ADMA_REF
PM_CLOCK_TIMESTAMP_REF	Clock ID TIMESTAMP_REF
PM_CLOCK_AMS_REF	Clock ID AMS_REF
PM_CLOCK_PL0_REF	Clock ID PL0_REF
PM_CLOCK_PL1_REF	Clock ID PL1_REF
PM_CLOCK_PL2_REF	Clock ID PL2_REF
PM_CLOCK_PL3_REF	Clock ID PL3_REF
PM_CLOCK_WDT	Clock ID WDT
PM_CLOCK_IOPLL_INT	Clock ID IOPLL_INT
PM_CLOCK_IOPLL_PRE_SRC	Clock ID IOPLL_PRE_SRC
PM_CLOCK_IOPLL_HALF	Clock ID IOPLL_HALF
PM_CLOCK_IOPLL_INT_MUX	Clock ID IOPLL_INT_MUX
PM_CLOCK_IOPLL_POST_SRC	Clock ID IOPLL_POST_SRC
PM_CLOCK_RPLL_INT	Clock ID RPLL_INT
PM_CLOCK_RPLL_PRE_SRC	Clock ID RPLL_PRE_SRC
PM_CLOCK_RPLL_HALF	Clock ID RPLL_HALF
PM_CLOCK_RPLL_INT_MUX	Clock ID RPLL_INT_MUX
PM_CLOCK_RPLL_POST_SRC	Clock ID RPLL_POST_SRC
PM_CLOCK_APLL_INT	Clock ID APLL_INT
PM_CLOCK_APLL_PRE_SRC	Clock ID APLL_PRE_SRC
PM_CLOCK_APLL_HALF	Clock ID APLL_HALF
PM_CLOCK_APLL_INT_MUX	Clock ID APLL_INT_MUX
PM_CLOCK_APLL_POST_SRC	Clock ID APLL_POST_SRC
PM_CLOCK_DPLL_INT	Clock ID DPLL_INT
PM_CLOCK_DPLL_PRE_SRC	Clock ID DPLL_PRE_SRC
PM_CLOCK_DPLL_HALF	Clock ID DPLL_HALF
PM_CLOCK_DPLL_INT_MUX	Clock ID DPLL_INT_MUX
PM_CLOCK_DPLL_POST_SRC	Clock ID DPLL_POST_SRC

Table 62: **Enumeration XPmClock Values** *(cont'd)*

Value	Description
PM_CLOCK_VPLL_INT	Clock ID VPLL_INT
PM_CLOCK_VPLL_PRE_SRC	Clock ID VPLL_PRE_SRC
PM_CLOCK_VPLL_HALF	Clock ID VPLL_HALF
PM_CLOCK_VPLL_INT_MUX	Clock ID VPLL_INT_MUX
PM_CLOCK_VPLL_POST_SRC	Clock ID VPLL_POST_SRC
PM_CLOCK_CAN0_MIO	Clock ID CAN0_MIO
PM_CLOCK_CAN1_MIO	Clock ID CAN1_MIO
PM_CLOCK_ACPU_FULL	Clock ID ACPU_FULL
PM_CLOCK_GEM0_REF	Clock ID GEM0_REF
PM_CLOCK_GEM1_REF	Clock ID GEM1_REF
PM_CLOCK_GEM2_REF	Clock ID GEM2_REF
PM_CLOCK_GEM3_REF	Clock ID GEM3_REF
PM_CLOCK_GEM0_REF_UNGATED	Clock ID GEM0_REF_UNGATED
PM_CLOCK_GEM1_REF_UNGATED	Clock ID GEM1_REF_UNGATED
PM_CLOCK_GEM2_REF_UNGATED	Clock ID GEM2_REF_UNGATED
PM_CLOCK_GEM3_REF_UNGATED	Clock ID GEM3_REF_UNGATED
PM_CLOCK_EXT_PSS_REF	Clock ID EXT_PSS_REF
PM_CLOCK_EXT_VIDEO	Clock ID EXT_VIDEO
PM_CLOCK_EXT_PSS_ALT_REF	Clock ID EXT_PSS_ALT_REF
PM_CLOCK_EXT_AUX_REF	Clock ID EXT_AUX_REF
PM_CLOCK_EXT_GT_CRX_REF	Clock ID EXT_GT_CRX_REF
PM_CLOCK_EXT_SWDT0	Clock ID EXT_SWDT0
PM_CLOCK_EXT_SWDT1	Clock ID EXT_SWDT1
PM_CLOCK_EXT_GEM0_TX_EMIO	Clock ID EXT_GEM0_TX_EMIO
PM_CLOCK_EXT_GEM1_TX_EMIO	Clock ID EXT_GEM1_TX_EMIO
PM_CLOCK_EXT_GEM2_TX_EMIO	Clock ID EXT_GEM2_TX_EMIO
PM_CLOCK_EXT_GEM3_TX_EMIO	Clock ID EXT_GEM3_TX_EMIO
PM_CLOCK_EXT_GEM0_RX_EMIO	Clock ID EXT_GEM0_RX_EMIO
PM_CLOCK_EXT_GEM1_RX_EMIO	Clock ID EXT_GEM1_RX_EMIO
PM_CLOCK_EXT_GEM2_RX_EMIO	Clock ID EXT_GEM2_RX_EMIO
PM_CLOCK_EXT_GEM3_RX_EMIO	Clock ID EXT_GEM3_RX_EMIO
PM_CLOCK_EXT_MIO50_OR_MIO51	Clock ID EXT_MIO50_OR_MIO51
PM_CLOCK_EXT_MIO0	Clock ID EXT_MIO0
PM_CLOCK_EXT_MIO1	Clock ID EXT_MIO1
PM_CLOCK_EXT_MIO2	Clock ID EXT_MIO2
PM_CLOCK_EXT_MIO3	Clock ID EXT_MIO3
PM_CLOCK_EXT_MIO4	Clock ID EXT_MIO4
PM_CLOCK_EXT_MIO5	Clock ID EXT_MIO5
PM_CLOCK_EXT_MIO6	Clock ID EXT_MIO6

Table 62: **Enumeration XPmClock Values** *(cont'd)*

Value	Description
PM_CLOCK_EXT_MIO7	Clock ID EXT_MIO7
PM_CLOCK_EXT_MIO8	Clock ID EXT_MIO8
PM_CLOCK_EXT_MIO9	Clock ID EXT_MIO9
PM_CLOCK_EXT_MIO10	Clock ID EXT_MIO10
PM_CLOCK_EXT_MIO11	Clock ID EXT_MIO11
PM_CLOCK_EXT_MIO12	Clock ID EXT_MIO12
PM_CLOCK_EXT_MIO13	Clock ID EXT_MIO13
PM_CLOCK_EXT_MIO14	Clock ID EXT_MIO14
PM_CLOCK_EXT_MIO15	Clock ID EXT_MIO15
PM_CLOCK_EXT_MIO16	Clock ID EXT_MIO16
PM_CLOCK_EXT_MIO17	Clock ID EXT_MIO17
PM_CLOCK_EXT_MIO18	Clock ID EXT_MIO18
PM_CLOCK_EXT_MIO19	Clock ID EXT_MIO19
PM_CLOCK_EXT_MIO20	Clock ID EXT_MIO20
PM_CLOCK_EXT_MIO21	Clock ID EXT_MIO21
PM_CLOCK_EXT_MIO22	Clock ID EXT_MIO22
PM_CLOCK_EXT_MIO23	Clock ID EXT_MIO23
PM_CLOCK_EXT_MIO24	Clock ID EXT_MIO24
PM_CLOCK_EXT_MIO25	Clock ID EXT_MIO25
PM_CLOCK_EXT_MIO26	Clock ID EXT_MIO26
PM_CLOCK_EXT_MIO27	Clock ID EXT_MIO27
PM_CLOCK_EXT_MIO28	Clock ID EXT_MIO28
PM_CLOCK_EXT_MIO29	Clock ID EXT_MIO29
PM_CLOCK_EXT_MIO30	Clock ID EXT_MIO30
PM_CLOCK_EXT_MIO31	Clock ID EXT_MIO31
PM_CLOCK_EXT_MIO32	Clock ID EXT_MIO32
PM_CLOCK_EXT_MIO33	Clock ID EXT_MIO33
PM_CLOCK_EXT_MIO34	Clock ID EXT_MIO34
PM_CLOCK_EXT_MIO35	Clock ID EXT_MIO35
PM_CLOCK_EXT_MIO36	Clock ID EXT_MIO36
PM_CLOCK_EXT_MIO37	Clock ID EXT_MIO37
PM_CLOCK_EXT_MIO38	Clock ID EXT_MIO38
PM_CLOCK_EXT_MIO39	Clock ID EXT_MIO39
PM_CLOCK_EXT_MIO40	Clock ID EXT_MIO40
PM_CLOCK_EXT_MIO41	Clock ID EXT_MIO41
PM_CLOCK_EXT_MIO42	Clock ID EXT_MIO42
PM_CLOCK_EXT_MIO43	Clock ID EXT_MIO43
PM_CLOCK_EXT_MIO44	Clock ID EXT_MIO44
PM_CLOCK_EXT_MIO45	Clock ID EXT_MIO45

Table 62: **Enumeration XPmClock Values** *(cont'd)*

Value	Description
PM_CLOCK_EXT_MIO46	Clock ID EXT_MIO46
PM_CLOCK_EXT_MIO47	Clock ID EXT_MIO47
PM_CLOCK_EXT_MIO48	Clock ID EXT_MIO48
PM_CLOCK_EXT_MIO49	Clock ID EXT_MIO49
PM_CLOCK_EXT_MIO50	Clock ID EXT_MIO50
PM_CLOCK_EXT_MIO51	Clock ID EXT_MIO51
PM_CLOCK_EXT_MIO52	Clock ID EXT_MIO52
PM_CLOCK_EXT_MIO53	Clock ID EXT_MIO53
PM_CLOCK_EXT_MIO54	Clock ID EXT_MIO54
PM_CLOCK_EXT_MIO55	Clock ID EXT_MIO55
PM_CLOCK_EXT_MIO56	Clock ID EXT_MIO56
PM_CLOCK_EXT_MIO57	Clock ID EXT_MIO57
PM_CLOCK_EXT_MIO58	Clock ID EXT_MIO58
PM_CLOCK_EXT_MIO59	Clock ID EXT_MIO59
PM_CLOCK_EXT_MIO60	Clock ID EXT_MIO60
PM_CLOCK_EXT_MIO61	Clock ID EXT_MIO61
PM_CLOCK_EXT_MIO62	Clock ID EXT_MIO62
PM_CLOCK_EXT_MIO63	Clock ID EXT_MIO63
PM_CLOCK_EXT_MIO64	Clock ID EXT_MIO64
PM_CLOCK_EXT_MIO65	Clock ID EXT_MIO65
PM_CLOCK_EXT_MIO66	Clock ID EXT_MIO66
PM_CLOCK_EXT_MIO67	Clock ID EXT_MIO67
PM_CLOCK_EXT_MIO68	Clock ID EXT_MIO68
PM_CLOCK_EXT_MIO69	Clock ID EXT_MIO69
PM_CLOCK_EXT_MIO70	Clock ID EXT_MIO70
PM_CLOCK_EXT_MIO71	Clock ID EXT_MIO71
PM_CLOCK_EXT_MIO72	Clock ID EXT_MIO72
PM_CLOCK_EXT_MIO73	Clock ID EXT_MIO73
PM_CLOCK_EXT_MIO74	Clock ID EXT_MIO74
PM_CLOCK_EXT_MIO75	Clock ID EXT_MIO75
PM_CLOCK_EXT_MIO76	Clock ID EXT_MIO76
PM_CLOCK_EXT_MIO77	Clock ID EXT_MIO77

Enumeration XPmPllParam

PLL parameters

Table 63: **Enumeration XPmPllParam Values**

Value	Description
PM_PLL_PARAM_ID_DIV2	PLL param ID DIV2
PM_PLL_PARAM_ID_FBDIV	PLL param ID FBDIV
PM_PLL_PARAM_ID_DATA	PLL param ID DATA
PM_PLL_PARAM_ID_PRE_SRC	PLL param ID PRE_SRC
PM_PLL_PARAM_ID_POST_SRC	PLL param ID POST_SRC
PM_PLL_PARAM_ID_LOCK_DLY	PLL param ID LOCK_DLY
PM_PLL_PARAM_ID_LOCK_CNT	PLL param ID LOCK_CNT
PM_PLL_PARAM_ID_LFHF	PLL param ID LFHF
PM_PLL_PARAM_ID_CP	PLL param ID CP
PM_PLL_PARAM_ID_RES	PLL param ID RES

Enumeration XPmPllMode

PLL Modes

Table 64: **Enumeration XPmPllMode Values**

Value	Description
PM_PLL_MODE_INTEGER	PLL mode integer
PM_PLL_MODE_FRACTIONAL	PLL mode fractional
PM_PLL_MODE_RESET	PLL mode reset

Enumeration XPmPinFn

Pin Function IDs

Table 65: **Enumeration XPmPinFn Values**

Value	Description
PINCTRL_FUNC_CAN0	Pin Function CAN0
PINCTRL_FUNC_CAN1	Pin Function CAN1
PINCTRL_FUNC_ETHERNET0	Pin Function ETHERNET0
PINCTRL_FUNC_ETHERNET1	Pin Function ETHERNET1
PINCTRL_FUNC_ETHERNET2	Pin Function ETHERNET2
PINCTRL_FUNC_ETHERNET3	Pin Function ETHERNET3
PINCTRL_FUNC_GEMTSU0	Pin Function GEMTSU0
PINCTRL_FUNC_GPIO0	Pin Function GPIO0
PINCTRL_FUNC_I2C0	Pin Function I2C0
PINCTRL_FUNC_I2C1	Pin Function I2C1
PINCTRL_FUNC_MDIO0	Pin Function MDIO0

Table 65: **Enumeration XPmPinFn Values** *(cont'd)*

Value	Description
PINCTRL_FUNC_MDIO1	Pin Function MDIO1
PINCTRL_FUNC_MDIO2	Pin Function MDIO2
PINCTRL_FUNC_MDIO3	Pin Function MDIO3
PINCTRL_FUNC_QSPI0	Pin Function QSPI0
PINCTRL_FUNC_QSPI_FBCLK	Pin Function QSPI_FBCLK
PINCTRL_FUNC_QSPI_SS	Pin Function QSPI_SS
PINCTRL_FUNC_SPI0	Pin Function SPI0
PINCTRL_FUNC_SPI1	Pin Function SPI1
PINCTRL_FUNC_SPI0_SS	Pin Function SPI0_SS
PINCTRL_FUNC_SPI1_SS	Pin Function SPI1_SS
PINCTRL_FUNC_SDIO0	Pin Function SDIO0
PINCTRL_FUNC_SDIO0_PC	Pin Function SDIO0_PC
PINCTRL_FUNC_SDIO0_CD	Pin Function SDIO0_CD
PINCTRL_FUNC_SDIO0_WP	Pin Function SDIO0_WP
PINCTRL_FUNC_SDIO1	Pin Function SDIO1
PINCTRL_FUNC_SDIO1_PC	Pin Function SDIO1_PC
PINCTRL_FUNC_SDIO1_CD	Pin Function SDIO1_CD
PINCTRL_FUNC_SDIO1_WP	Pin Function SDIO1_WP
PINCTRL_FUNC_NAND0	Pin Function NAND0
PINCTRL_FUNC_NAND0_CE	Pin Function NAND0_CE
PINCTRL_FUNC_NAND0_RB	Pin Function NAND0_RB
PINCTRL_FUNC_NAND0_DQS	Pin Function NAND0_DQS
PINCTRL_FUNC_TTC0_CLK	Pin Function TTC0_CLK
PINCTRL_FUNC_TTC0_WAV	Pin Function TTC0_WAV
PINCTRL_FUNC_TTC1_CLK	Pin Function TTC1_CLK
PINCTRL_FUNC_TTC1_WAV	Pin Function TTC1_WAV
PINCTRL_FUNC_TTC2_CLK	Pin Function TTC2_CLK
PINCTRL_FUNC_TTC2_WAV	Pin Function TTC2_WAV
PINCTRL_FUNC_TTC3_CLK	Pin Function TTC3_CLK
PINCTRL_FUNC_TTC3_WAV	Pin Function TTC3_WAV
PINCTRL_FUNC_UART0	Pin Function UART0
PINCTRL_FUNC_UART1	Pin Function UART1
PINCTRL_FUNC_USB0	Pin Function USB0
PINCTRL_FUNC_USB1	Pin Function USB1
PINCTRL_FUNC_SWDT0_CLK	Pin Function SWDT0_CLK
PINCTRL_FUNC_SWDT0_RST	Pin Function SWDT0_RST
PINCTRL_FUNC_SWDT1_CLK	Pin Function SWDT1_CLK
PINCTRL_FUNC_SWDT1_RST	Pin Function SWDT1_RST
PINCTRL_FUNC_PMU0	Pin Function PMU0

Table 65: **Enumeration XPmPinFn Values** *(cont'd)*

Value	Description
PINCTRL_FUNC_PCIE0	Pin Function PCIE0
PINCTRL_FUNC_CSU0	Pin Function CSU0
PINCTRL_FUNC_DPAUX0	Pin Function DPAUX0
PINCTRL_FUNC_PJTAG0	Pin Function PJTAG0
PINCTRL_FUNC_TRACE0	Pin Function TRACE0
PINCTRL_FUNC_TRACE0_CLK	Pin Function TRACE0_CLK
PINCTRL_FUNC_TESTSCAN0	Pin Function TESTSCAN0

Enumeration XPmPinParam

PIN Control Parameters

Table 66: **Enumeration XPmPinParam Values**

Value	Description
PINCTRL_CONFIG_SLEW_RATE	Pin config slew rate
PINCTRL_CONFIG_BIAS_STATUS	Pin config bias status
PINCTRL_CONFIG_PULL_CTRL	Pin config pull control
PINCTRL_CONFIG_SCHMITT_CMOS	Pin config schmitt CMOS
PINCTRL_CONFIG_DRIVE_STRENGTH	Pin config drive strength
PINCTRL_CONFIG_VOLTAGE_STATUS	Pin config voltage status

Enumeration pm_ioctl_id

IOCTL IDs

Table 67: **Enumeration pm_ioctl_id Values**

Value	Description
IOCTL_GET_RPU_OPER_MODE	Get RPU mode
IOCTL_SET_RPU_OPER_MODE	Set RPU mode
IOCTL_RPU_BOOT_ADDR_CONFIG	RPU boot address config
IOCTL_TCM_COMB_CONFIG	TCM config
IOCTL_SET_TAPDELAY_BYPASS	TAP delay bypass
IOCTL_SET_SGMII_MODE	SGMII mode
IOCTL_SD_DLL_RESET	SD DLL reset
IOCTL_SET_SD_TAPDELAY	SD TAP delay
IOCTL_SET_PLL_FRAC_MODE	Set PLL frac mode
IOCTL_GET_PLL_FRAC_MODE	Get PLL frac mode
IOCTL_SET_PLL_FRAC_DATA	Set PLL frac data

Table 67: **Enumeration pm_ioctl_id Values** *(cont'd)*

Value	Description
IOCTL_GET_PLL_FRAC_DATA	Get PLL frac data
IOCTL_WRITE_GGS	Write GGS
IOCTL_READ_GGS	Read GGS
IOCTL_WRITE_PGGS	Write PGGS
IOCTL_READ_PGGS	Read PGGS
IOCTL_ULPI_RESET	ULPI reset
IOCTL_SET_BOOT_HEALTH_STATUS	Set boot status
IOCTL_AFI	AFI
IOCTL_PROBE_COUNTER_READ	Probe counter read
IOCTL_PROBE_COUNTER_WRITE	Probe counter write
IOCTL_OSPI_MUX_SELECT	OSPI mux select
IOCTL_USB_SET_STATE	USB set state
IOCTL_GET_LAST_RESET_REASON	Get last reset reason
IOCTL_AIE_ISR_CLEAR	AIE ISR clear
IOCTL_REGISTER_SGI	Register SGI to ATF
IOCTL_SET_FEATURE_CONFIG	Set runtime feature config
IOCTL_GET_FEATURE_CONFIG	Get runtime feature config
IOCTL_GET_RPU_OPER_MODE	Get RPU mode
IOCTL_SET_RPU_OPER_MODE	Set RPU mode
IOCTL_RPU_BOOT_ADDR_CONFIG	RPU boot address config
IOCTL_TCM_COMB_CONFIG	TCM config
IOCTL_SET_TAPDELAY_BYPASS	TAP delay bypass
IOCTL_SET_SGMII_MODE	SGMII mode
IOCTL_SD_DLL_RESET	SD DLL reset
IOCTL_SET_SD_TAPDELAY	SD TAP delay
IOCTL_SET_PLL_FRAC_MODE	Set PLL frac mode
IOCTL_GET_PLL_FRAC_MODE	Get PLL frac mode
IOCTL_SET_PLL_FRAC_DATA	Set PLL frac data
IOCTL_GET_PLL_FRAC_DATA	Get PLL frac data
IOCTL_WRITE_GGS	Write GGS
IOCTL_READ_GGS	Read GGS
IOCTL_WRITE_PGGS	Write PGGS
IOCTL_READ_PGGS	Read PGGS
IOCTL_ULPI_RESET	ULPI reset
IOCTL_SET_BOOT_HEALTH_STATUS	Set boot status
IOCTL_AFI	AFI
IOCTL_PROBE_COUNTER_READ	Probe counter read
IOCTL_PROBE_COUNTER_WRITE	Probe counter write
IOCTL_OSPI_MUX_SELECT	OSPI mux select

Table 67: **Enumeration pm_ioctl_id Values** *(cont'd)*

Value	Description
IOCTL_USB_SET_STATE	USB set state
IOCTL_GET_LAST_RESET_REASON	Get last reset reason
IOCTL_AIE_ISR_CLEAR	AIE ISR clear
IOCTL_REGISTER_SGI	Register SGI to ATF
IOCTL_SET_FEATURE_CONFIG	Set feature config
IOCTL_GET_FEATURE_CONFIG	Get feature config

Enumeration pm_feature_id

IOCTL IDs

Table 68: **Enumeration pm_feature_id Values**

Value	Description
XPM_FEATURE_INVALID	Invalid ID
XPM_FEATURE_OVERTEMP_STATUS	Over temperature status
XPM_FEATURE_OVERTEMP_VALUE	Over temperature limit
XPM_FEATURE_EXTWDT_STATUS	External watchdog status
XPM_FEATURE_EXTWDT_VALUE	External watchdog interval

Definitions

Define XST_PM_INTERNAL

Definition

```
#define XST_PM_INTERNAL2000L
```

Description

Power management specific return error status 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

XilPM Versal ACAP APIs

Power Management (XilPM) provides the Embedded Energy Management Interface (EEMI) APIs for power management on ACAP devices. For more details about EEMI, see the*Embedded Energy Management Interface EEMI API Reference 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 69: **Quick Function Reference**

Type	Name	Arguments
XStatus	XPm_InitXilpm	XIpiPsu * IpiInst
enum	XPm_GetBootStatus	void
XStatus	XPm_GetChipID	u32 * IDCode u32 * Version
XStatus	XPm_GetApiVersion	u32 * Version
XStatus	XPm_RequestNode	const u32 DeviceId const u32 Capabilities const u32 QoS const u32 Ack
XStatus	XPm_ReleaseNode	const u32 DeviceId
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

Table 69: **Quick Function Reference** *(cont'd)*

Type	Name	Arguments
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
XStatus	XPm_ClockGetStatus	const u32 ClockId u32 *const State
XStatus	XPm_ClockSetDivider	const u32 ClockId const u32 Divider

Table 69: **Quick Function Reference** *(cont'd)*

Type	Name	Arguments
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
XStatus	XPm_ClockGetRate	const u32 ClockId u32 *const Rate
XStatus	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
XStatus	XPm_RequestWakeUp	const u32 TargetDevId const u8 SetAddress const u64 Address const u32 Ack
void	XPm_SuspendFinalize	void

Table 69: **Quick Function Reference** *(cont'd)*

Type	Name	Arguments
XStatus	XPm_RequestSuspend	const u32 TargetSubsystemId const u32 Ack const u32 Latency const u32 State
XStatus	XPm_AbortSuspend	const enum XPmAbortReason 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	const u32 QueryId const u32 Arg1 const u32 Arg2 const u32 Arg3 u32 *const Data
XStatus	XPm_SetMaxLatency	const u32 DeviceId const u32 Latency
XStatus	XPm_GetOpCharacteristic	const u32 DeviceId const enum XPmOpCharType Type u32 *const Result
XStatus	XPm_InitFinalize	void
XStatus	XPm_RegisterNotifier	XPm_Notifier *const Notifier
XStatus	XPm_UnregisterNotifier	XPm_Notifier *const Notifier
void	XPm_InitSuspendCb	const enum XPmSuspendReason Reason const u32 Latency const u32 State const u32 Timeout

Table 69: **Quick Function Reference** *(cont'd)*

Type	Name	Arguments
void	XPm_AcknowledgeCb	const u32 Node const XStatus Status const u32 Oppoint
void	XPm_NotifyCb	const u32 Node const u32 Event const u32 Oppoint
XStatus	XPm_FeatureCheck	const u32 FeatureId u32 * Version

Functions

XPm_InitXilpm

Initialize xilpm library.

Prototype

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

Parameters

The following table lists the `XPm_InitXilpm` function arguments.

Table 70: **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 71: **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 72: **XPm_GetApiVersion Arguments**

Type	Name	Description
u32 *	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 73: **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 74: **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 75: **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 76: **XPm_GetNodeStatus Arguments**

Type	Name	Description
const u32	DeviceId	Device for which status is requested
`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 77: **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 78: **XPm_ResetGetStatus Arguments**

Type	Name	Description
const u32	ResetId	Reset ID

Table 78: **XPm_ResetGetStatus Arguments** *(cont'd)*

Type	Name	Description
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 79: **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 80: **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 81: **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 82: **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
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 83: **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
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 84: **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 85: **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 86: **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 87: **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 88: **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 89: **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 90: **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 91: **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 92: **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

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

Parameters

The following table lists the `XPm_ClockGetRate` function arguments.

Table 93: **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

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

Parameters

The following table lists the `XPm_ClockSetRate` function arguments.

Table 94: **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 95: **XPm_PllSetParameter Arguments**

Type	Name	Description
const u32	ClockId	Clock ID
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 96: **XPm_PllGetParameter Arguments**

Type	Name	Description
const u32	ClockId	Clock ID

Table 96: **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 97: **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 98: **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 99: **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 100: **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 101: **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 102: **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_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 103: **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 104: **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 105: **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 106: **XPm_Query Arguments**

Type	Name	Description
const u32	QueryId	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

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

Parameters

The following table lists the `XPm_SetMaxLatency` function arguments.

Table 107: **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 108: **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

```
XStatus 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

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

Parameters

The following table lists the `XPm_RegisterNotifier` function arguments.

Table 109: **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

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

Parameters

The following table lists the `XPm_UnregisterNotifier` function arguments.

Table 110: **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 111: **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 111: **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 112: **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 u32 Event, const u32 Oppoint);
```

Parameters

The following table lists the `XPm_NotifyCb` function arguments.

Table 113: **XPm_NotifyCb Arguments**

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

Returns

None

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 114: **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

Table 115: **Enumeration XPmAbortReason Values**

Value	Description
ABORT_REASON_WKUP_EVENT	Wakeup Event
ABORT_REASON_PU_BUSY	Processor Busy
ABORT_REASON_NO_PWRDN	No Powerdown
ABORT_REASON_UNKNOWN	Unknown Reason

Enumeration XPmBootStatus

Boot status enumeration.

Table 116: **Enumeration XPmBootStatus Values**

Value	Description
PM_INITIAL_BOOT	Boot is a fresh system startup
PM_RESUME	Boot is a resume
PM_BOOT_ERROR	Error. Cause cannot be identified

Enumeration XPmRequestAck

PM Acknowledge Request Types

Table 117: **Enumeration XPmRequestAck Values**

Value	Description
REQUEST_ACK_NO	No Ack
REQUEST_ACK_BLOCKING	Blocking Ack
REQUEST_ACK_NON_BLOCKING	Non blocking Ack
REQUEST_ACK_CB_CERROR	Callback Error

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
PM_CAP_SECURE	Secure access type (non-secure/secure)
PM_CAP_COHERENT	Device Coherency
PM_CAP_VIRTUALIZED	Device Virtualization

Enumeration XPmDeviceUsage

Usage status, returned by PmGetNodeStatus

Table 119: **Enumeration XPmDeviceUsage Values**

Value	Description
PM_USAGE_CURRENT_SUBSYSTEM	Current subsystem is using
PM_USAGE_OTHER_SUBSYSTEM	Other subsystem is using

Enumeration XPmResetActions

Reset configuration argument

Table 120: **Enumeration XPmResetActions Values**

Value	Description
PM_RESET_ACTION_RELEASE	Reset action release
PM_RESET_ACTION_ASSERT	Reset action assert
PM_RESET_ACTION_PULSE	Reset action pulse

Enumeration XPmSuspendReason

Suspend reasons

Table 121: **Enumeration XPmSuspendReason Values**

Value	Description
SUSPEND_REASON_PU_REQ	Processor request
SUSPEND_REASON_ALERT	Alert
SUSPEND_REASON_SYS_SHUTDOWN	System shutdown

Enumeration XPmApiCbId_t

PM API callback IDs

Table 122: **Enumeration XPmApiCbId_t Values**

Value	Description
PM_INIT_SUSPEND_CB	Suspend callback
PM_ACKNOWLEDGE_CB	Acknowledge callback
PM_NOTIFY_CB	Notify callback

Enumeration pm_query_id

Query IDs

Table 123: **Enumeration pm_query_id Values**

Value	Description
XPM_QID_INVALID	Invalid Query ID
XPM_QID_CLOCK_GET_NAME	Get clock name
XPM_QID_CLOCK_GET_TOPOLOGY	Get clock topology
XPM_QID_CLOCK_GET_FIXEDFACTOR_PARAMS	Get clock fixedfactor parameter
XPM_QID_CLOCK_GET_MUXSOURCES	Get clock mux sources
XPM_QID_CLOCK_GET_ATTRIBUTES	Get clock attributes
XPM_QID_PINCTRL_GET_NUM_PINS	Get total pins
XPM_QID_PINCTRL_GET_NUM_FUNCTIONS	Get total pin functions
XPM_QID_PINCTRL_GET_NUM_FUNCTION_GROUPS	Get total pin function groups
XPM_QID_PINCTRL_GET_FUNCTION_NAME	Get pin function name
XPM_QID_PINCTRL_GET_FUNCTION_GROUPS	Get pin function groups
XPM_QID_PINCTRL_GET_PIN_GROUPS	Get pin groups
XPM_QID_CLOCK_GET_NUM_CLOCKS	Get number of clocks
XPM_QID_CLOCK_GET_MAX_DIVISOR	Get max clock divisor
XPM_QID_PLD_GET_PARENT	Get PLD parent

Enumeration PmPinFunIds

Pin Function IDs

Table 124: **Enumeration PmPinFunIds Values**

Value	Description
PIN_FUNC_SPI0	Pin function ID of SPI0
PIN_FUNC_SPI0_SS	Pin function ID of SPI0_SS

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

Value	Description
PIN_FUNC_SPI1	Pin function ID of SPI1
PIN_FUNC_SPI1_SS	Pin function ID of SPI1_SS
PIN_FUNC_CAN0	Pin function ID of CAN0
PIN_FUNC_CAN1	Pin function ID of CAN1
PIN_FUNC_I2C0	Pin function ID of I2C0
PIN_FUNC_I2C1	Pin function ID of I2C1
PIN_FUNC_I2C_PMC	Pin function ID of I2C_PMC
PIN_FUNC_TTC0_CLK	Pin function ID of TTC0_CLK
PIN_FUNC_TTC0_WAV	Pin function ID of TTC0_WAV
PIN_FUNC_TTC1_CLK	Pin function ID of TTC1_CLK
PIN_FUNC_TTC1_WAV	Pin function ID of TTC1_WAV
PIN_FUNC_TTC2_CLK	Pin function ID of TTC2_CLK
PIN_FUNC_TTC2_WAV	Pin function ID of TTC2_WAV
PIN_FUNC_TTC3_CLK	Pin function ID of TTC3_CLK
PIN_FUNC_TTC3_WAV	Pin function ID of TTC3_WAV
PIN_FUNC_WWDT0	Pin function ID of WWDT0
PIN_FUNC_WWDT1	Pin function ID of WWDT1
PIN_FUNC_SYSMON_I2C0	Pin function ID of SYSMON_I2C0
PIN_FUNC_SYSMON_I2C0_ALERT	Pin function ID of SYSMON_I2C0_AL
PIN_FUNC_UART0	Pin function ID of UART0
PIN_FUNC_UART0_CTRL	Pin function ID of UART0_CTRL
PIN_FUNC_UART1	Pin function ID of UART1
PIN_FUNC_UART1_CTRL	Pin function ID of UART1_CTRL
PIN_FUNC_GPIO0	Pin function ID of GPIO0
PIN_FUNC_GPIO1	Pin function ID of GPIO1
PIN_FUNC_GPIO2	Pin function ID of GPIO2
PIN_FUNC_EMIO0	Pin function ID of EMIO0
PIN_FUNC_GEM0	Pin function ID of GEM0
PIN_FUNC_GEM1	Pin function ID of GEM1
PIN_FUNC_TRACE0	Pin function ID of TRACE0
PIN_FUNC_TRACE0_CLK	Pin function ID of TRACE0_CLK
PIN_FUNC_MDIO0	Pin function ID of MDIO0
PIN_FUNC_MDIO1	Pin function ID of MDIO1
PIN_FUNC_GEM_TSU0	Pin function ID of GEM_TSU0
PIN_FUNC_PCIE0	Pin function ID of PCIE0
PIN_FUNC_SMAP0	Pin function ID of SMAP0
PIN_FUNC_USB0	Pin function ID of USB0
PIN_FUNC_SD0	Pin function ID of SD0
PIN_FUNC_SD0_PC	Pin function ID of SD0_PC

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

Value	Description
PIN_FUNC_SD0_CD	Pin function ID of SD0_CD
PIN_FUNC_SD0_WP	Pin function ID of SD0_WP
PIN_FUNC_SD1	Pin function ID of SD1
PIN_FUNC_SD1_PC	Pin function ID of SD1_PC
PIN_FUNC_SD1_CD	Pin function ID of SD1_CD
PIN_FUNC_SD1_WP	Pin function ID of SD1_WP
PIN_FUNC_OSPI0	Pin function ID of OSPI0
PIN_FUNC_OSPI0_SS	Pin function ID of OSPI0_SS
PIN_FUNC_QSPI0	Pin function ID of QSPI0
PIN_FUNC_QSPI0_FBCLK	Pin function ID of QSPI0_FBCLK
PIN_FUNC_QSPI0_SS	Pin function ID of QSPI0_SS
PIN_FUNC_TEST_CLK	Pin function ID of TEST_CLK
PIN_FUNC_TEST_SCAN	Pin function ID of TEST_SCAN
PIN_FUNC_TAMPER_TRIGGER	Pin function ID of TAMPER_TRIGGER
MAX_FUNCTION	Max Pin function

Enumeration pm_pinctrl_config_param

Pin Control Configuration

Table 125: **Enumeration pm_pinctrl_config_param Values**

Value	Description
PINCTRL_CONFIG_SLEW_RATE	Pin config slew rate
PINCTRL_CONFIG_BIAS_STATUS	Pin config bias status
PINCTRL_CONFIG_PULL_CTRL	Pin config pull control
PINCTRL_CONFIG_SCHMITT_CMOS	Pin config schmitt CMOS
PINCTRL_CONFIG_DRIVE_STRENGTH	Pin config drive strength
PINCTRL_CONFIG_VOLTAGE_STATUS	Pin config voltage status
PINCTRL_CONFIG_TRI_STATE	Pin config tri state
PINCTRL_CONFIG_MAX	Max Pin config

Enumeration pm_pinctrl_slew_rate

Pin Control Slew Rate

Table 126: **Enumeration pm_pinctrl_slew_rate Values**

Value	Description
PINCTRL_SLEW_RATE_FAST	Fast slew rate
PINCTRL_SLEW_RATE_SLOW	Slow slew rate

Enumeration pm_pinctrl_bias_status

Pin Control Bias Status

Table 127: **Enumeration pm_pinctrl_bias_status Values**

Value	Description
PINCTRL_BIAS_DISABLE	Bias disable
PINCTRL_BIAS_ENABLE	Bias enable

Enumeration pm_pinctrl_pull_ctrl

Pin Control Pull Control

Table 128: **Enumeration pm_pinctrl_pull_ctrl Values**

Value	Description
PINCTRL_BIAS_PULL_DOWN	Bias pull-down
PINCTRL_BIAS_PULL_UP	Bias pull-up

Enumeration pm_pinctrl_schmitt_cmos

Pin Control Input Type

Table 129: **Enumeration pm_pinctrl_schmitt_cmos Values**

Value	Description
PINCTRL_INPUT_TYPE_CMOS	Input type CMOS
PINCTRL_INPUT_TYPE_SCHMITT	Input type SCHMITT

Enumeration pm_pinctrl_drive_strength

Pin Control Drive Strength

Table 130: **Enumeration pm_pinctrl_drive_strength Values**

Value	Description
PINCTRL_DRIVE_STRENGTH_TRISTATE	tri-state
PINCTRL_DRIVE_STRENGTH_4MA	4mA
PINCTRL_DRIVE_STRENGTH_8MA	8mA
PINCTRL_DRIVE_STRENGTH_12MA	12mA
PINCTRL_DRIVE_STRENGTH_MAX	Max value

Enumeration pm_pinctrl_tri_state

Pin Control Tri State

Table 131: **Enumeration pm_pinctrl_tri_state Values**

Value	Description
PINCTRL_TRI_STATE_DISABLE	Tri state disable
PINCTRL_TRI_STATE_ENABLE	Tri state enable

Enumeration pm_ioctl_id

IOCTL IDs

Table 132: **Enumeration pm_ioctl_id Values**

Value	Description
IOCTL_GET_RPU_OPER_MODE	Get RPU mode
IOCTL_SET_RPU_OPER_MODE	Set RPU mode
IOCTL_RPU_BOOT_ADDR_CONFIG	RPU boot address config
IOCTL_TCM_COMB_CONFIG	TCM config
IOCTL_SET_TAPDELAY_BYPASS	TAP delay bypass
IOCTL_SET_SGMII_MODE	SGMII mode
IOCTL_SD_DLL_RESET	SD DLL reset
IOCTL_SET_SD_TAPDELAY	SD TAP delay
IOCTL_SET_PLL_FRAC_MODE	Set PLL frac mode
IOCTL_GET_PLL_FRAC_MODE	Get PLL frac mode
IOCTL_SET_PLL_FRAC_DATA	Set PLL frac data
IOCTL_GET_PLL_FRAC_DATA	Get PLL frac data
IOCTL_WRITE_GGS	Write GGS
IOCTL_READ_GGS	Read GGS
IOCTL_WRITE_PGGS	Write PGGS
IOCTL_READ_PGGS	Read PGGS

Table 132: **Enumeration pm_ioctl_id Values** *(cont'd)*

Value	Description
IOCTL_ULPI_RESET	ULPI reset
IOCTL_SET_BOOT_HEALTH_STATUS	Set boot status
IOCTL_AFI	AFI
IOCTL_PROBE_COUNTER_READ	Probe counter read
IOCTL_PROBE_COUNTER_WRITE	Probe counter write
IOCTL_OSPI_MUX_SELECT	OSPI mux select
IOCTL_USB_SET_STATE	USB set state
IOCTL_GET_LAST_RESET_REASON	Get last reset reason
IOCTL_AIE_ISR_CLEAR	AIE ISR clear
IOCTL_REGISTER_SGI	Register SGI to ATF
IOCTL_SET_FEATURE_CONFIG	Set runtime feature config
IOCTL_GET_FEATURE_CONFIG	Get runtime feature config
IOCTL_GET_RPU_OPER_MODE	Get RPU mode
IOCTL_SET_RPU_OPER_MODE	Set RPU mode
IOCTL_RPU_BOOT_ADDR_CONFIG	RPU boot address config
IOCTL_TCM_COMB_CONFIG	TCM config
IOCTL_SET_TAPDELAY_BYPASS	TAP delay bypass
IOCTL_SET_SGMII_MODE	SGMII mode
IOCTL_SD_DLL_RESET	SD DLL reset
IOCTL_SET_SD_TAPDELAY	SD TAP delay
IOCTL_SET_PLL_FRAC_MODE	Set PLL frac mode
IOCTL_GET_PLL_FRAC_MODE	Get PLL frac mode
IOCTL_SET_PLL_FRAC_DATA	Set PLL frac data
IOCTL_GET_PLL_FRAC_DATA	Get PLL frac data
IOCTL_WRITE_GGS	Write GGS
IOCTL_READ_GGS	Read GGS
IOCTL_WRITE_PGGS	Write PGGS
IOCTL_READ_PGGS	Read PGGS
IOCTL_ULPI_RESET	ULPI reset
IOCTL_SET_BOOT_HEALTH_STATUS	Set boot status
IOCTL_AFI	AFI
IOCTL_PROBE_COUNTER_READ	Probe counter read
IOCTL_PROBE_COUNTER_WRITE	Probe counter write
IOCTL_OSPI_MUX_SELECT	OSPI mux select
IOCTL_USB_SET_STATE	USB set state
IOCTL_GET_LAST_RESET_REASON	Get last reset reason
IOCTL_AIE_ISR_CLEAR	AIE ISR clear
IOCTL_REGISTER_SGI	Register SGI to ATF
IOCTL_SET_FEATURE_CONFIG	Set feature config

Table 132: **Enumeration pm_ioctl_id Values** *(cont'd)*

Value	Description
IOCTL_GET_FEATURE_CONFIG	Get feature config

Enumeration XPm_PllConfigParams

PLL parameters

Table 133: **Enumeration XPm_PllConfigParams Values**

Value	Description
PM_PLL_PARAM_ID_DIV2	PLL param ID DIV2
PM_PLL_PARAM_ID_FBDIV	PLL param ID FBDIV
PM_PLL_PARAM_ID_DATA	PLL param ID DATA
PM_PLL_PARAM_ID_PRE_SRC	PLL param ID PRE_SRC
PM_PLL_PARAM_ID_POST_SRC	PLL param ID POST_SRC
PM_PLL_PARAM_ID_LOCK_DLY	PLL param ID LOCK_DLY
PM_PLL_PARAM_ID_LOCK_CNT	PLL param ID LOCK_CNT
PM_PLL_PARAM_ID_LFHF	PLL param ID LFHF
PM_PLL_PARAM_ID_CP	PLL param ID CP
PM_PLL_PARAM_ID_RES	PLL param ID RES
PM_PLL_PARAM_MAX	PLL param ID max

Enumeration XPmPllMode

PLL modes

Table 134: **Enumeration XPmPllMode Values**

Value	Description
PM_PLL_MODE_INTEGER	PLL mode integer
PM_PLL_MODE_FRACTIONAL	PLL mode fractional
PM_PLL_MODE_RESET	PLL mode reset

Enumeration XPmInitFunctions

PM init node functions

Table 135: **Enumeration XPmInitFunctions Values**

Value	Description
FUNC_INIT_START	Function ID INIT_START

Table 135: **Enumeration XPmInitFunctions Values** *(cont'd)*

Value	Description
FUNC_INIT_FINISH	Function ID INIT_FINISH
FUNC_SCAN_CLEAR	Function ID SCAN_CLEAR
FUNC_BISR	Function ID BISR
FUNC_LBIST	Function ID LBIST
FUNC_MEM_INIT	Function ID MEM_INIT
FUNC_MBIST_CLEAR	Function ID MBIST_CLEAR
FUNC_HOUSECLEAN_PL	Function ID HOUSECLEAN_PL
FUNC_HOUSECLEAN_COMPLETE	Function ID HOUSECLEAN_COMPLETE
FUNC_MAX_COUNT_PMINIT	Function ID MAX

Enumeration XPmOpCharType

PM operating characteristic types

Table 136: **Enumeration XPmOpCharType Values**

Value	Description
PM_OPCHAR_TYPE_POWER	Operating characteristic ID power
PM_OPCHAR_TYPE_TEMP	Operating characteristic ID temperature
PM_OPCHAR_TYPE_LATENCY	Operating characteristic ID latency

Enumeration XPmNotifyEvent

PM notify events

Table 137: **Enumeration XPmNotifyEvent Values**

Value	Description
EVENT_STATE_CHANGE	State change event
EVENT_ZERO_USERS	Zero user event
EVENT_CPU_IDLE_FORCE_PWRDWN	CPU idle event during force power down

Enumeration XPm_ApiId

PM API IDs

Table 138: **Enumeration XPm_ApiId Values**

Value	Description
PM_API_MIN	0x0

Table 138: **Enumeration XPm_ApiId Values** *(cont'd)*

Value	Description
PM_GET_API_VERSION	0x1
PM_SET_CONFIGURATION	0x2
PM_GET_NODE_STATUS	0x3
PM_GET_OP_CHARACTERISTIC	0x4
PM_REGISTER_NOTIFIER	0x5
PM_REQUEST_SUSPEND	0x6
PM_SELF_SUSPEND	0x7
PM_FORCE_POWERDOWN	0x8
PM_ABORT_SUSPEND	0x9
PM_REQUEST_WAKEUP	0xA
PM_SET_WAKEUP_SOURCE	0xB
PM_SYSTEM_SHUTDOWN	0xC
PM_REQUEST_NODE	0xD
PM_RELEASE_NODE	0xE
PM_SET_REQUIREMENT	0xF
PM_SET_MAX_LATENCY	0x10
PM_RESET_ASSERT	0x11
PM_RESET_GET_STATUS	0x12
PM_MMIO_WRITE	0x13
PM_MMIO_READ	0x14
PM_INIT_FINALIZE	0x15
PM_FPGA_LOAD	0x16
PM_FPGA_GET_STATUS	0x17
PM_GET_CHIPID	0x18
PM_SECURE_RSA_AES	0x19
PM_SECURE_SHA	0x1A
PM_SECURE_RSA	0x1B
PM_PINCTRL_REQUEST	0x1C
PM_PINCTRL_RELEASE	0x1D
PM_PINCTRL_GET_FUNCTION	0x1E
PM_PINCTRL_SET_FUNCTION	0x1F
PM_PINCTRL_CONFIG_PARAM_GET	0x20
PM_PINCTRL_CONFIG_PARAM_SET	0x21
PM_IOCTL	0x22
PM_QUERY_DATA	0x23
PM_CLOCK_ENABLE	0x24
PM_CLOCK_DISABLE	0x25
PM_CLOCK_GETSTATE	0x26
PM_CLOCK_SETDIVIDER	0x27

Table 138: **Enumeration XPm_ApiId Values** *(cont'd)*

Value	Description
PM_CLOCK_GETDIVIDER	0x28
PM_CLOCK_SETRATE	0x29
PM_CLOCK_GETRATE	0x2A
PM_CLOCK_SETPARENT	0x2B
PM_CLOCK_GETPARENT	0x2C
PM_SECURE_IMAGE	0x2D
PM_FPGA_READ	0x2E
PM_API_RESERVED_1	0x2F
PM_PLL_SET_PARAMETER	0x30
PM_PLL_GET_PARAMETER	0x31
PM_PLL_SET_MODE	0x32
PM_PLL_GET_MODE	0x33
PM_REGISTER_ACCESS	0x34
PM_EFUSE_ACCESS	0x35
PM_ADD_SUBSYSTEM	0x36
PM_DESTROY_SUBSYSTEM	0x37
PM_DESCRIBE_NODES	0x38
PM_ADD_NODE	0x39
PM_ADD_NODE_PARENT	0x3A
PM_ADD_NODE_NAME	0x3B
PM_ADD_REQUIREMENT	0x3C
PM_SET_CURRENT_SUBSYSTEM	0x3D
PM_INIT_NODE	0x3E
PM_FEATURE_CHECK	0x3F
PM_ISO_CONTROL	0x40
PM_ACTIVATE_SUBSYSTEM	0x41
PM_API_MAX	0x42

Definitions

Define PM_VERSION_MAJOR

Definition

```
#define PM_VERSION_MAJOR1UL
```

Description

PM Version Number

Define PM_VERSION_MINOR

Definition

```
#define PM_VERSION_MINOR0UL
```

Description

PM Version Number

Define PM_VERSION

Definition

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

Description

PM Version Number

Define XPM_MAX_CAPABILITY

Definition

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

Description

Requirement limits

Define XPM_MAX_LATENCY

Definition

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

Description

Requirement limits

Define XPM_MAX_QOS

Definition

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

Description

Requirement limits

Define XPM_MIN_CAPABILITY

Definition

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

Description

Requirement limits

Define XPM_MIN_LATENCY

Definition

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

Description

Requirement limits

Define XPM_MIN_QOS

Definition

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

Description

Requirement limits

Define XPM_DEF_CAPABILITY

Definition

```
#define XPM_DEF_CAPABILITY
                XPM_MAX_CAPABILITY
```

Description

Requirement limits

Define XPM_DEF_LATENCY

Definition

```
#define XPM_DEF_LATENCY
                XPM_MAX_LATENCY
```

Description

Requirement limits

Define XPM_DEF_QOS

Definition

```
#define XPM_DEF_QOS
                XPM_MAX_QOS
```

Description

Requirement limits

Define NODE_STATE_OFF

Definition

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

Description

Device node status

Define NODE_STATE_ON

Definition

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

Description

Device node status

Define PROC_STATE_SLEEP

Definition

```
#define PROC_STATE_SLEEP
                NODE_STATE_OFF
```

Description

Processor node status

Define PROC_STATE_ACTIVE

Definition

```
#define PROC_STATE_ACTIVE
                NODE_STATE_ON
```

Description

Processor node status

Define PROC_STATE_FORCEDOFF

Definition

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

Description

Processor node status

Define PROC_STATE_SUSPENDING

Definition

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

Description

Processor node status

Define PM_SHUTDOWN_TYPE_SHUTDOWN

Definition

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

Description

System shutdown macros

Define PM_SHUTDOWN_TYPE_RESET

Definition

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

Description

System shutdown macros

Define PM_SHUTDOWN_SUBTYPE_RST_SUBSYSTEM

Definition

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

Description

System shutdown macros

Define PM_SHUTDOWN_SUBTYPE_RST_PS_ONLY

Definition

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

Description

System shutdown macros

Define PM_SHUTDOWN_SUBTYPE_RST_SYSTEM

Definition

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

Description

System shutdown macros

Define PM_SUSPEND_STATE_CPU_IDLE

Definition

```
#define PM_SUSPEND_STATE_CPU_IDLE0x0U
```

Description

State arguments of the self suspend

Define PM_SUSPEND_STATE_SUSPEND_TO_RAM

Definition

```
#define PM_SUSPEND_STATE_SUSPEND_TO_RAM0xFU
```

Description

State arguments of the self suspend

Define XPM_RPU_MODE_LOCKSTEP

Definition

```
#define XPM_RPU_MODE_LOCKSTEP0U
```

Description

RPU operation mode

Define XPM_RPU_MODE_SPLIT

Definition

```
#define XPM_RPU_MODE_SPLIT1U
```

Description

RPU operation mode

Define XPM_RPU_BOOTMEM_LOVEC

Definition

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

Description

RPU Boot memory

Define XPM_RPU_BOOTMEM_HIVEC

Definition

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

Description

RPU Boot memory

Define XPM_RPU_TCM_SPLIT

Definition

```
#define XPM_RPU_TCM_SPLIT0U
```

Description

RPU TCM mode

Define XPM_RPU_TCM_COMB

Definition

```
#define XPM_RPU_TCM_COMB1U
```

Description

RPU TCM mode

Define XPM_BOOT_HEALTH_STATUS_MASK

Definition

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

Description

Boot health status mask

Define XPM_TAPDELAY_QSPI

Definition

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

Description

Tap delay signal type

Define XPM_TAPDELAY_BYPASS_DISABLE

Definition

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

Description

Tap delay bypass

Define XPM_TAPDELAY_BYPASS_ENABLE

Definition

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

Description

Tap delay bypass

Define XPM_OSPI_MUX_SEL_DMA

Definition

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

Description

Ospi AXI Mux select

Define XPM_OSPI_MUX_SEL_LINEAR

Definition

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

Description

Ospi AXI Mux select

Define XPM_OSPI_MUX_GET_MODE

Definition

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

Description

Ospi AXI Mux select

Define XPM_TAPDELAY_INPUT

Definition

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

Description

Tap delay type

Define XPM_TAPDELAY_OUTPUT

Definition

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

Description

Tap delay type

Define XPM_DLL_RESET_ASSERT

Definition

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

Description

Dll reset type

Define XPM_DLL_RESET_RELEASE

Definition

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

Description

Dll reset type

Define XPM_DLL_RESET_PULSE

Definition

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

Description

Dll reset type

Define XPM_RESET_REASON_EXT_POR

Definition

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

Description

Reset Reason

Define XPM_RESET_REASON_SW_POR

Definition

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

Description

Reset Reason

Define XPM_RESET_REASON_SLR_POR

Definition

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

Description

Reset Reason

Define XPM_RESET_REASON_ERR_POR

Definition

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

Description

Reset Reason

Define XPM_RESET_REASON_DAP_SRST

Definition

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

Description

Reset Reason

Define XPM_RESET_REASON_ERR_SRST

Definition

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

Description

Reset Reason

Define XPM_RESET_REASON_SW_SRST

Definition

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

Description

Reset Reason

Define XPM_RESET_REASON_SLR_SRST

Definition

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

Description

Reset Reason

Define XPM_RESET_REASON_INVALID

Definition

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

Description

Reset Reason

Define XPM_PROBE_COUNTER_TYPE_LAR_LSR

Definition

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

Description

Probe Counter Type

Define XPM_PROBE_COUNTER_TYPE_MAIN_CTL

Definition

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

Description

Probe Counter Type

Define XPM_PROBE_COUNTER_TYPE_CFG_CTL

Definition

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

Description

Probe Counter Type

Define XPM_PROBE_COUNTER_TYPE_STATE_PERIOD

Definition

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

Description

Probe Counter Type

Define XPM_PROBE_COUNTER_TYPE_PORT_SEL

Definition

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

Description

Probe Counter Type

Define XPM_PROBE_COUNTER_TYPE_SRC

Definition

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

Description

Probe Counter Type

Define XPM_PROBE_COUNTER_TYPE_VAL

Definition

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

Description

Probe Counter Type

Define XST_API_BASE_VERSION

Definition

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

Description

PM API versions

Define XST_API_QUERY_DATA_VERSION

Definition

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

Description

PM API versions

Define XST_API_REG_NOTIFIER_VERSION

Definition

```
#define XST_API_REG_NOTIFIER_VERSION(2U)
```

Description

PM API versions

Define HOUSECLEAN_DISABLE_DEFAULT_MASK

Definition

```
#define HOUSECLEAN_DISABLE_DEFAULT_MASK(0x0000U)
```

Description

Houseclean Disable Masks

Define HOUSECLEAN_DISABLE_SCAN_CLEAR_MASK

Definition

```
#define HOUSECLEAN_DISABLE_SCAN_CLEAR_MASK(0x0001U)
```

Description

Houseclean Disable Masks

Define HOUSECLEAN_DISABLE_BISR_MASK

Definition

```
#define HOUSECLEAN_DISABLE_BISR_MASK(0x0002U)
```

Description

Houseclean Disable Masks

Define HOUSECLEAN_DISABLE_LBIST_MASK

Definition

```
#define HOUSECLEAN_DISABLE_LBIST_MASK(0x0004U)
```

Description

Houseclean Disable Masks

Define HOUSECLEAN_DISABLE_MBIST_CLEAR_MASK

Definition

```
#define HOUSECLEAN_DISABLE_MBIST_CLEAR_MASK(0x0008U)
```

Description

Houseclean Disable Masks

Define HOUSECLEAN_DISABLE_PL_HC_MASK

Definition

```
#define HOUSECLEAN_DISABLE_PL_HC_MASK(0x0010U)
```

Description

Houseclean Disable Masks

Power Nodes

Definitions

Define PM_POWER_PMC

Definition

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

Description

Versal Power Nodes

Define PM_POWER_LPD

Definition

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

Description

Versal Power Nodes

Define PM_POWER_FPD

Definition

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

Description

Versal Power Nodes

Define PM_POWER_NOC

Definition

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

Description

Versal Power Nodes

Define PM_POWER_ME

Definition

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

Description

Versal Power Nodes

Define PM_POWER_PLD

Definition

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

Description

Versal Power Nodes

Define PM_POWER_CPM

Definition

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

Description

Versal Power Nodes

Define PM_POWER_PL_SYSMON

Definition

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

Description

Versal Power Nodes

Define PM_POWER_RPU0_0

Definition

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

Description

Versal Power Nodes

Define PM_POWER_GEM0

Definition

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

Description

Versal Power Nodes

Define PM_POWER_GEM1

Definition

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

Description

Versal Power Nodes

Define PM_POWER_OCM_0

Definition

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

Description

Versal Power Nodes

Define PM_POWER_OCM_1

Definition

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

Description

Versal Power Nodes

Define PM_POWER_OCM_2

Definition

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

Description

Versal Power Nodes

Define PM_POWER_OCM_3

Definition

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

Description

Versal Power Nodes

Define PM_POWER_TCM_0_A

Definition

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

Description

Versal Power Nodes

Define PM_POWER_TCM_0_B

Definition

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

Description

Versal Power Nodes

Define PM_POWER_TCM_1_A

Definition

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

Description

Versal Power Nodes

Define PM_POWER_TCM_1_B

Definition

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

Description

Versal Power Nodes

Define PM_POWER_ACPU_0

Definition

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

Description

Versal Power Nodes

Define PM_POWER_ACPU_1

Definition

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

Description

Versal Power Nodes

Define PM_POWER_L2_BANK_0

Definition

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

Description

Versal Power Nodes

Reset Nodes

Definitions

Define PM_RST_PMC_POR

Definition

```
#define PM_RST_PMC_POR(0xc30c001U)
```

Description

Versal Reset Nodes

Define PM_RST_PMC

Definition

```
#define PM_RST_PMC(0xc410002U)
```

Description

Versal Reset Nodes

Define PM_RST_PS_POR

Definition

```
#define PM_RST_PS_POR(0xc30c003U)
```

Description

Versal Reset Nodes

Define PM_RST_PL_POR

Definition

```
#define PM_RST_PL_POR(0xc30c004U)
```

Description

Versal Reset Nodes

Define PM_RST_NOC_POR

Definition

```
#define PM_RST_NOC_POR(0xc30c005U)
```

Description

Versal Reset Nodes

Define PM_RST_FPD_POR

Definition

```
#define PM_RST_FPD_POR(0xc30c006U)
```

Description

Versal Reset Nodes

Define PM_RST_ACPU_0_POR

Definition

```
#define PM_RST_ACPU_0_POR(0xc30c007U)
```

Description

Versal Reset Nodes

Define PM_RST_ACPU_1_POR

Definition

```
#define PM_RST_ACPU_1_POR(0xc30c008U)
```

Description

Versal Reset Nodes

Define PM_RST_OCM2_POR

Definition

```
#define PM_RST_OCM2_POR(0xc30c009U)
```

Description

Versal Reset Nodes

Define PM_RST_PS_SRST

Definition

```
#define PM_RST_PS_SRST(0xc41000aU)
```

Description

Versal Reset Nodes

Define PM_RST_PL_SRST

Definition

```
#define PM_RST_PL_SRST(0xc41000bU)
```

Description

Versal Reset Nodes

Define PM_RST_NOC

Definition

```
#define PM_RST_NOC(0xc41000cU)
```

Description

Versal Reset Nodes

Define PM_RST_NPI

Definition

```
#define PM_RST_NPI(0xc41000dU)
```

Description

Versal Reset Nodes

Define PM_RST_SYS_RST_1

Definition

```
#define PM_RST_SYS_RST_1(0xc41000eU)
```

Description

Versal Reset Nodes

Define PM_RST_SYS_RST_2

Definition

```
#define PM_RST_SYS_RST_2(0xc41000fU)
```

Description

Versal Reset Nodes

Define PM_RST_SYS_RST_3

Definition

```
#define PM_RST_SYS_RST_3(0xc410010U)
```

Description

Versal Reset Nodes

Define PM_RST_FPD

Definition

```
#define PM_RST_FPD(0xc410011U)
```

Description

Versal Reset Nodes

Define PM_RST_PL0

Definition

```
#define PM_RST_PL0(0xc410012U)
```

Description

Versal Reset Nodes

Define PM_RST_PL1

Definition

```
#define PM_RST_PL1(0xc410013U)
```

Description

Versal Reset Nodes

Define PM_RST_PL2

Definition

```
#define PM_RST_PL2(0xc410014U)
```

Description

Versal Reset Nodes

Define PM_RST_PL3

Definition

```
#define PM_RST_PL3(0xc410015U)
```

Description

Versal Reset Nodes

Define PM_RST_APU

Definition

```
#define PM_RST_APU(0xc410016U)
```

Description

Versal Reset Nodes

Define PM_RST_ACPU_0

Definition

```
#define PM_RST_ACPU_0(0xc410017U)
```

Description

Versal Reset Nodes

Define PM_RST_ACPU_1

Definition

```
#define PM_RST_ACPU_1(0xc410018U)
```

Description

Versal Reset Nodes

Define PM_RST_ACPU_L2

Definition

```
#define PM_RST_ACPU_L2(0xc410019U)
```

Description

Versal Reset Nodes

Define PM_RST_ACPU_GIC

Definition

```
#define PM_RST_ACPU_GIC(0xc41001aU)
```

Description

Versal Reset Nodes

Define PM_RST_RPU_ISLAND

Definition

```
#define PM_RST_RPU_ISLAND(0xc41001bU)
```

Description

Versal Reset Nodes

Define PM_RST_RPU_AMBA

Definition

```
#define PM_RST_RPU_AMBA(0xc41001cU)
```

Description

Versal Reset Nodes

Define PM_RST_R5_0

Definition

```
#define PM_RST_R5_0(0xc41001dU)
```

Description

Versal Reset Nodes

Define PM_RST_R5_1

Definition

```
#define PM_RST_R5_1(0xc41001eU)
```

Description

Versal Reset Nodes

Define PM_RST_SYSMON_PMC_SEQ_RST

Definition

```
#define PM_RST_SYSMON_PMC_SEQ_RST(0xc41001fU)
```

Description

Versal Reset Nodes

Define PM_RST_SYSMON_PMC_CFG_RST

Definition

```
#define PM_RST_SYSMON_PMC_CFG_RST(0xc410020U)
```

Description

Versal Reset Nodes

Define PM_RST_SYSMON_FPD_CFG_RST

Definition

```
#define PM_RST_SYSMON_FPD_CFG_RST(0xc410021U)
```

Description

Versal Reset Nodes

Define PM_RST_SYSMON_FPD_SEQ_RST

Definition

```
#define PM_RST_SYSMON_FPD_SEQ_RST(0xc410022U)
```

Description

Versal Reset Nodes

Define PM_RST_SYSMON_LPD

Definition

```
#define PM_RST_SYSMON_LPD(0xc410023U)
```

Description

Versal Reset Nodes

Define PM_RST_PDMA_RST1

Definition

```
#define PM_RST_PDMA_RST1(0xc410024U)
```

Description

Versal Reset Nodes

Define PM_RST_PDMA_RST0

Definition

```
#define PM_RST_PDMA_RST0(0xc410025U)
```

Description

Versal Reset Nodes

Define PM_RST_ADMA

Definition

```
#define PM_RST_ADMA(0xc410026U)
```

Description

Versal Reset Nodes

Define PM_RST_TIMESTAMP

Definition

```
#define PM_RST_TIMESTAMP(0xc410027U)
```

Description

Versal Reset Nodes

Define PM_RST_OCM

Definition

```
#define PM_RST_OCM(0xc410028U)
```

Description

Versal Reset Nodes

Define PM_RST_OCM2_RST

Definition

```
#define PM_RST_OCM2_RST(0xc410029U)
```

Description

Versal Reset Nodes

Define PM_RST_IPI

Definition

```
#define PM_RST_IPI(0xc41002aU)
```

Description

Versal Reset Nodes

Define PM_RST_SBI

Definition

```
#define PM_RST_SBI(0xc41002bU)
```

Description

Versal Reset Nodes

Define PM_RST_LPD

Definition

```
#define PM_RST_LPD(0xc41002cU)
```

Description

Versal Reset Nodes

Define PM_RST_QSPI

Definition

```
#define PM_RST_QSPI(0xc10402dU)
```

Description

Versal Reset Nodes

Define PM_RST_OSPI

Definition

```
#define PM_RST_OSPI(0xc10402eU)
```

Description

Versal Reset Nodes

Define PM_RST_SDIO_0

Definition

```
#define PM_RST_SDIO_0(0xc10402fU)
```

Description

Versal Reset Nodes

Define PM_RST_SDIO_1

Definition

```
#define PM_RST_SDIO_1(0xc104030U)
```

Description

Versal Reset Nodes

Define PM_RST_I2C_PMC

Definition

```
#define PM_RST_I2C_PMC(0xc104031U)
```

Description

Versal Reset Nodes

Define PM_RST_GPIO_PMC

Definition

```
#define PM_RST_GPIO_PMC(0xc104032U)
```

Description

Versal Reset Nodes

Define PM_RST_GEM_0

Definition

```
#define PM_RST_GEM_0(0xc104033U)
```

Description

Versal Reset Nodes

Define PM_RST_GEM_1

Definition

```
#define PM_RST_GEM_1(0xc104034U)
```

Description

Versal Reset Nodes

Define PM_RST_SPARE

Definition

```
#define PM_RST_SPARE(0xc104035U)
```

Description

Versal Reset Nodes

Define PM_RST_USB_0

Definition

```
#define PM_RST_USB_0(0xc104036U)
```

Description

Versal Reset Nodes

Define PM_RST_UART_0

Definition

```
#define PM_RST_UART_0(0xc104037U)
```

Description

Versal Reset Nodes

Define PM_RST_UART_1

Definition

```
#define PM_RST_UART_1(0xc104038U)
```

Description

Versal Reset Nodes

Define PM_RST_SPI_0

Definition

```
#define PM_RST_SPI_0(0xc104039U)
```

Description

Versal Reset Nodes

Define PM_RST_SPI_1

Definition

```
#define PM_RST_SPI_1(0xc10403aU)
```

Description

Versal Reset Nodes

Define PM_RST_CAN_FD_0

Definition

```
#define PM_RST_CAN_FD_0(0xc10403bU)
```

Description

Versal Reset Nodes

Define PM_RST_CAN_FD_1

Definition

```
#define PM_RST_CAN_FD_1(0xc10403cU)
```

Description

Versal Reset Nodes

Define PM_RST_I2C_0

Definition

```
#define PM_RST_I2C_0(0xc10403dU)
```

Description

Versal Reset Nodes

Define PM_RST_I2C_1

Definition

```
#define PM_RST_I2C_1(0xc10403eU)
```

Description

Versal Reset Nodes

Define PM_RST_GPIO_LPD

Definition

```
#define PM_RST_GPIO_LPD(0xc10403fU)
```

Description

Versal Reset Nodes

Define PM_RST_TTC_0

Definition

```
#define PM_RST_TTC_0(0xc104040U)
```

Description

Versal Reset Nodes

Define PM_RST_TTC_1

Definition

```
#define PM_RST_TTC_1(0xc104041U)
```

Description

Versal Reset Nodes

Define PM_RST_TTC_2

Definition

```
#define PM_RST_TTC_2(0xc104042U)
```

Description

Versal Reset Nodes

Define PM_RST_TTC_3

Definition

```
#define PM_RST_TTC_3(0xc104043U)
```

Description

Versal Reset Nodes

Define PM_RST_SWDT_FPD

Definition

```
#define PM_RST_SWDT_FPD(0xc104044U)
```

Description

Versal Reset Nodes

Define PM_RST_SWDT_LPD

Definition

```
#define PM_RST_SWDT_LPD(0xc104045U)
```

Description

Versal Reset Nodes

Define PM_RST_USB

Definition

```
#define PM_RST_USB(0xc104046U)
```

Description

Versal Reset Nodes

Define PM_RST_DPC

Definition

```
#define PM_RST_DPC(0xc208047U)
```

Description

Versal Reset Nodes

Define PM_RST_PMCDBG

Definition

```
#define PM_RST_PMCDBG(0xc208048U)
```

Description

Versal Reset Nodes

Define PM_RST_DBG_TRACE

Definition

```
#define PM_RST_DBG_TRACE(0xc208049U)
```

Description

Versal Reset Nodes

Define PM_RST_DBG_FPD

Definition

```
#define PM_RST_DBG_FPD(0xc20804aU)
```

Description

Versal Reset Nodes

Define PM_RST_DBG_TSTMP

Definition

```
#define PM_RST_DBG_TSTMP(0xc20804bU)
```

Description

Versal Reset Nodes

Define PM_RST_RPU0_DBG

Definition

```
#define PM_RST_RPU0_DBG(0xc20804cU)
```

Description

Versal Reset Nodes

Define PM_RST_RPU1_DBG

Definition

```
#define PM_RST_RPU1_DBG(0xc20804dU)
```

Description

Versal Reset Nodes

Define PM_RST_HSDP

Definition

```
#define PM_RST_HSDP(0xc20804eU)
```

Description

Versal Reset Nodes

Define PM_RST_DBG_LPD

Definition

```
#define PM_RST_DBG_LPD(0xc20804fU)
```

Description

Versal Reset Nodes

Define PM_RST_CPM_POR

Definition

```
#define PM_RST_CPM_POR(0xc30c050U)
```

Description

Versal Reset Nodes

Define PM_RST_CPM

Definition

```
#define PM_RST_CPM(0xc410051U)
```

Description

Versal Reset Nodes

Define PM_RST_CPMDBG

Definition

```
#define PM_RST_CPMDBG(0xc208052U)
```

Description

Versal Reset Nodes

Define PM_RST_PCIE_CFG

Definition

```
#define PM_RST_PCIE_CFG(0xc410053U)
```

Description

Versal Reset Nodes

Define PM_RST_PCIE_CORE0

Definition

```
#define PM_RST_PCIE_CORE0(0xc410054U)
```

Description

Versal Reset Nodes

Define PM_RST_PCIE_CORE1

Definition

```
#define PM_RST_PCIE_CORE1(0xc410055U)
```

Description

Versal Reset Nodes

Define PM_RST_PCIE_DMA

Definition

```
#define PM_RST_PCIE_DMA(0xc410056U)
```

Description

Versal Reset Nodes

Define PM_RST_CMN

Definition

```
#define PM_RST_CMN(0xc410057U)
```

Description

Versal Reset Nodes

Define PM_RST_L2_0

Definition

```
#define PM_RST_L2_0(0xc410058U)
```

Description

Versal Reset Nodes

Define PM_RST_L2_1

Definition

```
#define PM_RST_L2_1(0xc410059U)
```

Description

Versal Reset Nodes

Define PM_RST_ADDR_REMAP

Definition

```
#define PM_RST_ADDR_REMAP(0xc41005aU)
```

Description

Versal Reset Nodes

Define PM_RST_CPI0

Definition

```
#define PM_RST_CPI0(0xc41005bU)
```

Description

Versal Reset Nodes

Define PM_RST_CPI1

Definition

```
#define PM_RST_CPI1(0xc41005cU)
```

Description

Versal Reset Nodes

Define PM_RST_AIE_ARRAY

Definition

```
#define PM_RST_AIE_ARRAY(0xc10405eU)
```

Description

Versal Reset Nodes

Define PM_RST_AIE_SHIM

Definition

```
#define PM_RST_AIE_SHIM(0xc10405fU)
```

Description

Versal Reset Nodes

Clock Nodes

Definitions

Define PM_CLK_PMC_PLL

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_APU_PLL

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_RPU_PLL

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CPM_PLL

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_NOC_PLL

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PMC_PRESRC

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PMC_POSTCLK

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PMC_PLL_OUT

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PPLL

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_NOC_PRESRC

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_NOC_POSTCLK

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_NOC_PLL_OUT

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_NPLL

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_APU_PRESRC

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_APU_POSTCLK

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_APU_PLL_OUT

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_APLL

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_RPU_PRESRC

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_RPU_POSTCLK

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_RPU_PLL_OUT

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_RPLL

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CPM_PRESRC

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CPM_POSTCLK

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CPM_PLL_OUT

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CPLL

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PPLL_TO_XPD

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_NPLL_TO_XPD

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_APLL_TO_XPD

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_RPLL_TO_XPD

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_EFUSE_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_SYSMON_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_IRO_SUSPEND_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_USB_SUSPEND

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_SWITCH_TIMEOUT

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_RCLK_PMC

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_RCLK_LPD

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_WDT

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_TTC0

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_TTC1

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_TTC2

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_TTC3

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_GEM_TSU

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_GEM_TSU_LB

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_MUXED_IRO_DIV2

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_MUXED_IRO_DIV4

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PSM_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_GEM0_RX

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_GEM0_TX

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_GEM1_RX

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_GEM1_TX

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CPM_CORE_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CPM_LSBUS_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CPM_DBG_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CPM_AUX0_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CPM_AUX1_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_QSPI_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_OSPI_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_SDIO0_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_SDIO1_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PMC_LSBUS_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_I2C_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_TEST_PATTERN_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_DFT_OSC_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PMC_PL0_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PMC_PL1_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PMC_PL2_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PMC_PL3_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CFU_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_SPARE_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_NPI_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_HSM0_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_HSM1_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_SD_DLL_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_FPD_TOP_SWITCH

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_FPD_LSBUS

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_ACPU

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_DBG_TRACE

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_DBG_FPD

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_LPD_TOP_SWITCH

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_ADMA

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_LPD_LSBUS

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CPU_R5

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CPU_R5_CORE

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CPU_R5_OCM

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CPU_R5_OCM2

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_IOU_SWITCH

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_GEM0_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_GEM1_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_GEM_TSU_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_USB0_BUS_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_UART0_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_UART1_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_SPI0_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_SPI1_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CAN0_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CAN1_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_I2C0_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_I2C1_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_DBG_LPD

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_TIMESTAMP_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_DBG_TSTMP

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_CPM_TOPSW_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_USB3_DUAL_REF

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_REF_CLK

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PL_ALT_REF_CLK

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_MUXED_IRO

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PL_EXT

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PL_LB

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_MIO_50_OR_51

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_MIO_24_OR_25

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_EMIO

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_MIO

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PL_PMC_ALT_REF_CLK

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PL_LPD_ALT_REF_CLK

Definition

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

Description

Versal Clock Nodes

Define PM_CLK_PL_FPD_ALT_REF_CLK

Definition

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

Description

Versal Clock Nodes

MIO Nodes

Definitions

Define PM_STMIC_LMIO_0

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_1

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_2

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_3

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_4

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_5

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_6

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_7

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_8

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_9

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_10

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_11

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_12

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_13

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_14

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_15

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_16

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_17

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_18

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_19

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_20

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_21

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_22

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_23

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_24

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_LMIO_25

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_0

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_1

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_2

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_3

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_4

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_5

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_6

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_7

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_8

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_9

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_10

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_11

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_12

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_13

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_14

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_15

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_16

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_17

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_18

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_19

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_20

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_21

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_22

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_23

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_24

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_25

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_26

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_27

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_28

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_29

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_30

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_31

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_32

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_33

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_34

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_35

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_36

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_37

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_38

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_39

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_40

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_41

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_42

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_43

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_44

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_45

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_46

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_47

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_48

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_49

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_50

Definition

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

Description

Versal MIO Nodes

Define PM_STMIC_PMIO_51

Definition

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

Description

Versal MIO Nodes

Device Nodes

Definitions

Define PM_DEV_PLD_0

Definition

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

Description

Versal Device Nodes

Define PM_DEV_PMC_PROC

Definition

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

Description

Versal Device Nodes

Define PM_DEV_PSM_PROC

Definition

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

Description

Versal Device Nodes

Define PM_DEV_ACPU_0

Definition

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

Description

Versal Device Nodes

Define PM_DEV_ACPU_1

Definition

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

Description

Versal Device Nodes

Define PM_DEV_RPU0_0

Definition

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

Description

Versal Device Nodes

Define PM_DEV_RPU0_1

Definition

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

Description

Versal Device Nodes

Define PM_DEV_OCM_0

Definition

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

Description

Versal Device Nodes

Define PM_DEV_OCM_1

Definition

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

Description

Versal Device Nodes

Define PM_DEV_OCM_2

Definition

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

Description

Versal Device Nodes

Define PM_DEV_OCM_3

Definition

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

Description

Versal Device Nodes

Define PM_DEV_TCM_0_A

Definition

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

Description

Versal Device Nodes

Define PM_DEV_TCM_0_B

Definition

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

Description

Versal Device Nodes

Define PM_DEV_TCM_1_A

Definition

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

Description

Versal Device Nodes

Define PM_DEV_TCM_1_B

Definition

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

Description

Versal Device Nodes

Define PM_DEV_L2_BANK_0

Definition

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

Description

Versal Device Nodes

Define PM_DEV_DDR_0

Definition

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

Description

Versal Device Nodes

Define PM_DEV_USB_0

Definition

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

Description

Versal Device Nodes

Define PM_DEV_GEM_0

Definition

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

Description

Versal Device Nodes

Define PM_DEV_GEM_1

Definition

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

Description

Versal Device Nodes

Define PM_DEV_SPI_0

Definition

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

Description

Versal Device Nodes

Define PM_DEV_SPI_1

Definition

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

Description

Versal Device Nodes

Define PM_DEV_I2C_0

Definition

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

Description

Versal Device Nodes

Define PM_DEV_I2C_1

Definition

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

Description

Versal Device Nodes

Define PM_DEV_CAN_FD_0

Definition

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

Description

Versal Device Nodes

Define PM_DEV_CAN_FD_1

Definition

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

Description

Versal Device Nodes

Define PM_DEV_UART_0

Definition

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

Description

Versal Device Nodes

Define PM_DEV_UART_1

Definition

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

Description

Versal Device Nodes

Define PM_DEV_GPIO

Definition

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

Description

Versal Device Nodes

Define PM_DEV_TTC_0

Definition

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

Description

Versal Device Nodes

Define PM_DEV_TTC_1

Definition

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

Description

Versal Device Nodes

Define PM_DEV_TTC_2

Definition

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

Description

Versal Device Nodes

Define PM_DEV_TTC_3

Definition

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

Description

Versal Device Nodes

Define PM_DEV_SWDT_LPD

Definition

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

Description

Versal Device Nodes

Define PM_DEV_SWDT_FPD

Definition

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

Description

Versal Device Nodes

Define PM_DEV_OSPI

Definition

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

Description

Versal Device Nodes

Define PM_DEV_QSPI

Definition

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

Description

Versal Device Nodes

Define PM_DEV_GPIO_PMC

Definition

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

Description

Versal Device Nodes

Define PM_DEV_I2C_PMC

Definition

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

Description

Versal Device Nodes

Define PM_DEV_SDIO_0

Definition

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

Description

Versal Device Nodes

Define PM_DEV_SDIO_1

Definition

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

Description

Versal Device Nodes

Define PM_DEV_RTC

Definition

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

Description

Versal Device Nodes

Define PM_DEV_ADMA_0

Definition

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

Description

Versal Device Nodes

Define PM_DEV_ADMA_1

Definition

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

Description

Versal Device Nodes

Define PM_DEV_ADMA_2

Definition

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

Description

Versal Device Nodes

Define PM_DEV_ADMA_3

Definition

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

Description

Versal Device Nodes

Define PM_DEV_ADMA_4

Definition

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

Description

Versal Device Nodes

Define PM_DEV_ADMA_5

Definition

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

Description

Versal Device Nodes

Define PM_DEV_ADMA_6

Definition

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

Description

Versal Device Nodes

Define PM_DEV_ADMA_7

Definition

```
#define PM_DEV_ADMA_7(0x1822403cU)
```

Description

Versal Device Nodes

Define PM_DEV_IPI_0

Definition

```
#define PM_DEV_IPI_0(0x1822403dU)
```

Description

Versal Device Nodes

Define PM_DEV_IPI_1

Definition

```
#define PM_DEV_IPI_1(0x1822403eU)
```

Description

Versal Device Nodes

Define PM_DEV_IPI_2

Definition

```
#define PM_DEV_IPI_2(0x1822403fU)
```

Description

Versal Device Nodes

Define PM_DEV_IPI_3

Definition

```
#define PM_DEV_IPI_3(0x18224040U)
```

Description

Versal Device Nodes

Define PM_DEV_IPI_4

Definition

```
#define PM_DEV_IPI_4(0x18224041U)
```

Description

Versal Device Nodes

Define PM_DEV_IPI_5

Definition

```
#define PM_DEV_IPI_5(0x18224042U)
```

Description

Versal Device Nodes

Define PM_DEV_IPI_6

Definition

```
#define PM_DEV_IPI_6(0x18224043U)
```

Description

Versal Device Nodes

Define PM_DEV_SOC

Definition

```
#define PM_DEV_SOC(0x18428044U)
```

Description

Versal Device Nodes

Define PM_DEV_DDRMC_0

Definition

```
#define PM_DEV_DDRMC_0(0x18520045U)
```

Description

Versal Device Nodes

Define PM_DEV_DDRMC_1

Definition

```
#define PM_DEV_DDRMC_1(0x18520046U)
```

Description

Versal Device Nodes

Define PM_DEV_DDRMC_2

Definition

```
#define PM_DEV_DDRMC_2(0x18520047U)
```

Description

Versal Device Nodes

Define PM_DEV_DDRMC_3

Definition

```
#define PM_DEV_DDRMC_3(0x18520048U)
```

Description

Versal Device Nodes

Define PM_DEV_GT_0

Definition

```
#define PM_DEV_GT_0(0x1862c049U)
```

Description

Versal Device Nodes

Define PM_DEV_GT_1

Definition

```
#define PM_DEV_GT_1(0x1862c04aU)
```

Description

Versal Device Nodes

Define PM_DEV_GT_2

Definition

```
#define PM_DEV_GT_2(0x1862c04bU)
```

Description

Versal Device Nodes

Define PM_DEV_GT_3

Definition

```
#define PM_DEV_GT_3(0x1862c04cU)
```

Description

Versal Device Nodes

Define PM_DEV_GT_4

Definition

```
#define PM_DEV_GT_4(0x1862c04dU)
```

Description

Versal Device Nodes

Define PM_DEV_GT_5

Definition

```
#define PM_DEV_GT_5(0x1862c04eU)
```

Description

Versal Device Nodes

Define PM_DEV_GT_6

Definition

```
#define PM_DEV_GT_6(0x1862c04fU)
```

Description

Versal Device Nodes

Define PM_DEV_GT_7

Definition

```
#define PM_DEV_GT_7(0x1862c050U)
```

Description

Versal Device Nodes

Define PM_DEV_GT_8

Definition

```
#define PM_DEV_GT_8(0x1862c051U)
```

Description

Versal Device Nodes

Define PM_DEV_GT_9

Definition

```
#define PM_DEV_GT_9(0x1862c052U)
```

Description

Versal Device Nodes

Define PM_DEV_GT_10

Definition

```
#define PM_DEV_GT_10(0x1862c053U)
```

Description

Versal Device Nodes

Define PM_DEV_EFUSE_CACHE

Definition

```
#define PM_DEV_EFUSE_CACHE(0x18330054U)
```

Description

Versal Device Nodes

Define PM_DEV_AMS_ROOT

Definition

```
#define PM_DEV_AMS_ROOT(0x18224055U)
```

Description

Versal Device Nodes

Define PM_DEV_AIE

Definition

```
#define PM_DEV_AIE(0x18224072U)
```

Description

Versal Device Nodes

Define PM_DEV_IPI_PMC

Definition

```
#define PM_DEV_IPI_PMC(0x18224073U)
```

Description

Versal Device Nodes

Define PM_DEV_GGS_0

Definition

```
#define PM_DEV_GGS_0(0x18248000U)
```

Description

Versal Virtual Device Nodes

Define PM_DEV_GGS_1

Definition

```
#define PM_DEV_GGS_1(0x18248001U)
```

Description

Versal Virtual Device Nodes

Define PM_DEV_GGS_2

Definition

```
#define PM_DEV_GGS_2(0x18248002U)
```

Description

Versal Virtual Device Nodes

Define PM_DEV_GGS_3

Definition

```
#define PM_DEV_GGS_3(0x18248003U)
```

Description

Versal Virtual Device Nodes

Define PM_DEV_PGGS_0

Definition

```
#define PM_DEV_PGGS_0(0x1824c004U)
```

Description

Versal Virtual Device Nodes

Define PM_DEV_PGGS_1

Definition

```
#define PM_DEV_PGGS_1(0x1824c005U)
```

Description

Versal Virtual Device Nodes

Define PM_DEV_PGGS_2

Definition

```
#define PM_DEV_PGGS_2(0x1824c006U)
```

Description

Versal Virtual Device Nodes

Define PM_DEV_PGGS_3

Definition

```
#define PM_DEV_PGGS_3(0x1824c007U)
```

Description

Versal Virtual Device Nodes

Define PM_DEV_HB_MON_0

Definition

```
#define PM_DEV_HB_MON_0(0x18250000U)
```

Description

Versal Virtual Device Nodes

Define PM_DEV_HB_MON_1

Definition

```
#define PM_DEV_HB_MON_1(0x18250001U)
```

Description

Versal Virtual Device Nodes

Define PM_DEV_HB_MON_2

Definition

```
#define PM_DEV_HB_MON_2(0x18250002U)
```

Description

Versal Virtual Device Nodes

Define PM_DEV_HB_MON_3

Definition

```
#define PM_DEV_HB_MON_3(0x18250003U)
```

Description

Versal Virtual Device Nodes

Subsystem Nodes

Definitions

Define PM_SUBSYS_DEFAULT

Definition

```
#define PM_SUBSYS_DEFAULT(0x1c000000U)
```

Description

Versal Subsystem Nodes

Define PM_SUBSYS_PMC

Definition

```
#define PM_SUBSYS_PMC(0x1c000001U)
```

Description

Versal Subsystem Nodes

Event Node IDs

Definitions

Define XPM_NODETYPE_EVENT_ERROR_PMC_ERR1

Definition

```
#define XPM_NODETYPE_EVENT_ERROR_PMC_ERR1(0x28100000U)
```

Description

Error Event Node Ids

Define XPM_NODETYPE_EVENT_ERROR_PMC_ERR2

Definition

```
#define XPM_NODETYPE_EVENT_ERROR_PMC_ERR2(0x28104000U)
```

Description

Error Event Node Ids

Define XPM_NODETYPE_EVENT_ERROR_PSM_ERR1

Definition

```
#define XPM_NODETYPE_EVENT_ERROR_PSM_ERR1(0x28108000U)
```

Description

Error Event Node Ids

Define XPM_NODETYPE_EVENT_ERROR_PSM_ERR2

Definition

```
#define XPM_NODETYPE_EVENT_ERROR_PSM_ERR2(0x2810C000U)
```

Description

Error Event Node Ids

Define XPM_NODETYPE_EVENT_ERROR_SW_ERR

Definition

```
#define XPM_NODETYPE_EVENT_ERROR_SW_ERR(0x28110000U)
```

Description

Error Event Node Ids

Error Event Mask

Error Event Mask for PMC ERR1

Error Events belong to PMC ERR1 Node.

Definitions

Define XPM_EVENT_ERROR_MASK_BOOT_CR

Definition

```
#define XPM_EVENT_ERROR_MASK_BOOT_CR(0x00000001U)
```

Description

Error event mask for PMC Boot Correctable Error. Set by ROM code during ROM execution during Boot.

Define XPM_EVENT_ERROR_MASK_BOOT_NCR

Definition

```
#define XPM_EVENT_ERROR_MASK_BOOT_NCR(0x00000002U)
```

Description

Error event mask for PMC Boot Non-Correctable Error. Set by ROM code during ROM execution during Boot.

Define XPM_EVENT_ERROR_MASK_FW_CR

Definition

```
#define XPM_EVENT_ERROR_MASK_FW_CR(0x00000004U)
```

Description

Error event mask for PMC Firmware Boot Correctable Error. Set by PLM during firmware execution during Boot.

Define XPM_EVENT_ERROR_MASK_FW_NCR

Definition

```
#define XPM_EVENT_ERROR_MASK_FW_NCR(0x00000008U)
```

Description

Error event mask for PMC Firmware Boot Non-Correctable Error. Set by PLM during firmware execution during Boot.

Define XPM_EVENT_ERROR_MASK_GSW_CR

Definition

```
#define XPM_EVENT_ERROR_MASK_GSW_CR(0x00000010U)
```

Description

Error event mask for General Software Correctable Error. Set by any processors after Boot.

Define XPM_EVENT_ERROR_MASK_GSW_NCR

Definition

```
#define XPM_EVENT_ERROR_MASK_GSW_NCR(0x00000020U)
```

Description

Error event mask for General Software Non-Correctable Error. Set by any processors after Boot.

Define XPM_EVENT_ERROR_MASK_CFU

Definition

```
#define XPM_EVENT_ERROR_MASK_CFU(0x00000040U)
```

Description

Error event mask for CFU Error.

Define XPM_EVENT_ERROR_MASK_CFRAME

Definition

```
#define XPM_EVENT_ERROR_MASK_CFRAME(0x00000080U)
```

Description

Error event mask for CFRAME Error.

Define XPM_EVENT_ERROR_MASK_PMC_PSM_CR

Definition

```
#define XPM_EVENT_ERROR_MASK_PMC_PSM_CR(0x00000100U)
```

Description

Error event mask for PSM Correctable Error, Summary from PSM Error Management.

Define XPM_EVENT_ERROR_MASK_PMC_PSM_NCR

Definition

```
#define XPM_EVENT_ERROR_MASK_PMC_PSM_NCR(0x00000200U)
```

Description

Error event mask for PSM Non-Correctable Error, Summary from PSM Error Management.

Define XPM_EVENT_ERROR_MASK_DDRMB_CR

Definition

```
#define XPM_EVENT_ERROR_MASK_DDRMB_CR(0x00000400U)
```

Description

Error event mask for DDRMC MB Correctable ECC Error.

Define XPM_EVENT_ERROR_MASK_DDRMB_NCR

Definition

```
#define XPM_EVENT_ERROR_MASK_DDRMB_NCR(0x00000800U)
```

Description

Error event mask for DDRMC MB Non-Correctable ECC Error.

Define XPM_EVENT_ERROR_MASK_NOCTYPE1_CR

Definition

```
#define XPM_EVENT_ERROR_MASK_NOCTYPE1_CR(0x00001000U)
```

Description

Error event mask for NoC Type1 Correctable Error.

Define XPM_EVENT_ERROR_MASK_NOCTYPE1_NCR

Definition

```
#define XPM_EVENT_ERROR_MASK_NOCTYPE1_NCR(0x00002000U)
```

Description

Error event mask for NoC Type1 Non-Correctable Error.

Define XPM_EVENT_ERROR_MASK_NOCUSER

Definition

```
#define XPM_EVENT_ERROR_MASK_NOCUSER(0x00004000U)
```

Description

Error event mask for NoC User Error.

Define XPM_EVENT_ERROR_MASK_MMCM

Definition

```
#define XPM_EVENT_ERROR_MASK_MMCM(0x00008000U)
```

Description

Error event mask for MMCM Lock Error.

Define XPM_EVENT_ERROR_MASK_AIE_CR

Definition

```
#define XPM_EVENT_ERROR_MASK_AIE_CR(0x00010000U)
```

Description

Error event mask for ME Correctable Error.

Define XPM_EVENT_ERROR_MASK_AIE_NCR

Definition

```
#define XPM_EVENT_ERROR_MASK_AIE_NCR(0x00020000U)
```

Description

Error event mask for ME Non-Correctable Error.

Define XPM_EVENT_ERROR_MASK_DDRMC_CR

Definition

```
#define XPM_EVENT_ERROR_MASK_DDRMC_CR(0x00040000U)
```

Description

Error event mask for DDRMC MC Correctable ECC Error.

Define XPM_EVENT_ERROR_MASK_DDRMC_NCR

Definition

```
#define XPM_EVENT_ERROR_MASK_DDRMC_NCR(0x00080000U)
```

Description

Error event mask for DDRMC MC Non-Correctable ECC Error.

Define XPM_EVENT_ERROR_MASK_GT_CR

Definition

```
#define XPM_EVENT_ERROR_MASK_GT_CR(0x00100000U)
```

Description

Error event mask for GT Correctable Error.

Define XPM_EVENT_ERROR_MASK_GT_NCR

Definition

```
#define XPM_EVENT_ERROR_MASK_GT_NCR(0x00200000U)
```

Description

Error event mask for GT Non-Correctable Error.

Define XPM_EVENT_ERROR_MASK_PLSMON_CR

Definition

```
#define XPM_EVENT_ERROR_MASK_PLSMON_CR(0x00400000U)
```

Description

Error event mask for PL Sysmon Correctable Error.

Define XPM_EVENT_ERROR_MASK_PLSMON_NCR

Definition

```
#define XPM_EVENT_ERROR_MASK_PLSMON_NCR(0x00800000U)
```

Description

Error event mask for PL Sysmon Non-Correctable Error.

Define XPM_EVENT_ERROR_MASK_PL0

Definition

```
#define XPM_EVENT_ERROR_MASK_PL0(0x01000000U)
```

Description

Error event mask for User defined PL generic error.

Define XPM_EVENT_ERROR_MASK_PL1

Definition

```
#define XPM_EVENT_ERROR_MASK_PL1(0x02000000U)
```

Description

Error event mask for User defined PL generic error.

Define XPM_EVENT_ERROR_MASK_PL2

Definition

```
#define XPM_EVENT_ERROR_MASK_PL2(0x04000000U)
```

Description

Error event mask for User defined PL generic error.

Define XPM_EVENT_ERROR_MASK_PL3

Definition

```
#define XPM_EVENT_ERROR_MASK_PL3(0x08000000U)
```

Description

Error event mask for User defined PL generic error.

Define XPM_EVENT_ERROR_MASK_NPIROOT

Definition

```
#define XPM_EVENT_ERROR_MASK_NPIROOT(0x10000000U)
```

Description

Error event mask for NPI Root Error.

Define XPM_EVENT_ERROR_MASK_SSIT3

Definition

```
#define XPM_EVENT_ERROR_MASK_SSIT3(0x20000000U)
```

Description

Error event mask for SSIT Error from Slave SLR1, Only used in Master SLR.

Define XPM_EVENT_ERROR_MASK_SSIT4

Definition

```
#define XPM_EVENT_ERROR_MASK_SSIT4(0x40000000U)
```

Description

Error event mask for SSIT Error from Slave SLR2, Only used in Master SLR.

Define XPM_EVENT_ERROR_MASK_SSIT5

Definition

```
#define XPM_EVENT_ERROR_MASK_SSIT5(0x80000000U)
```

Description

Error event mask for SSIT Error from Slave SLR3, Only used in Master SLR.

Error Event Mask for PMC ERR2

Error Events belong to PMC ERR2 Node.

Definitions

Define XPM_EVENT_ERROR_MASK_PMCAPB

Definition

```
#define XPM_EVENT_ERROR_MASK_PMCAPB(0x00000001U)
```

Description

Error event mask for General purpose PMC error, can be triggered by any of the following peripherals:,

- PMC Global Regsiters,- PMC Clock & Reset (CRP),- PMC IOU Secure SLCR,
- PMC IOU SLCR,- BBRAM Controller,- PMC Analog Control Registers,
- RTC Control Registers.

Define XPM_EVENT_ERROR_MASK_PMCROM

Definition

```
#define XPM_EVENT_ERROR_MASK_PMCROM(0x00000002U)
```

Description

Error event mask for PMC ROM Validation Error.

Define XPM_EVENT_ERROR_MASK_MB_FATAL0

Definition

```
#define XPM_EVENT_ERROR_MASK_MB_FATAL0(0x00000004U)
```

Description

Error event mask for PMC PPU0 MB TMR Fatal Error.

Define XPM_EVENT_ERROR_MASK_MB_FATAL1

Definition

```
#define XPM_EVENT_ERROR_MASK_MB_FATAL1(0x00000008U)
```

Description

Error event mask for PMC PPU1 MB TMR Fatal Error.

Define XPM_EVENT_ERROR_MASK_PMCPAR

Definition

```
#define XPM_EVENT_ERROR_MASK_PMCPAR(0x00000010U)
```

Description

Error event mask for PMC Switch and PMC IOU Parity Errors.

Define XPM_EVENT_ERROR_MASK_PMC_CR

Definition

```
#define XPM_EVENT_ERROR_MASK_PMC_CR(0x00000020U)
```

Description

Error event mask for PMC Correctable Errors:, PPU0 RAM correctable error.,PPU1 instruction RAM correctable error., PPU1 data RAM correctable error.

Define XPM_EVENT_ERROR_MASK_PMC_NCR

Definition

```
#define XPM_EVENT_ERROR_MASK_PMC_NCR(0x00000040U)
```

Description

Error event mask for PMC Non-Correctable Errors:, PPU0 RAM non-correctable error.,PPU1 instruction RAM non-correctable error., PPU1 data RAM non-correctable error.,PRAM non-correctable error.

Define XPM_EVENT_ERROR_MASK_PMCSMON0

Definition

```
#define XPM_EVENT_ERROR_MASK_PMCSMON0(0x00000080U)
```

Description

Error event mask for PMC Temperature Shutdown Alert and Power Supply Failure Detection Errors from PMC Sysmon alarm[0]. Indicates an alarm condition on any of SUPPLY0 to SUPPLY31.

Define XPM_EVENT_ERROR_MASK_PMCSMON1

Definition

```
#define XPM_EVENT_ERROR_MASK_PMCSMON1(0x00000100U)
```

Description

Error event mask for PMC Temperature Shutdown Alert and Power Supply Failure Detection Errors from PMC Sysmon alarm[1]. Indicates an alarm condition on any of SUPPLY32 to SUPPLY63.

Define XPM_EVENT_ERROR_MASK_PMCSMON2

Definition

```
#define XPM_EVENT_ERROR_MASK_PMCSMON2(0x00000200U)
```

Description

Error event mask for PMC Temperature Shutdown Alert and Power Supply Failure Detection Errors from PMC Sysmon alarm[2]. Indicates an alarm condition on any of SUPPLY64 to SUPPLY95.

Define XPM_EVENT_ERROR_MASK_PMCSMON3

Definition

```
#define XPM_EVENT_ERROR_MASK_PMCSMON3(0x00000400U)
```

Description

Error event mask for PMC Temperature Shutdown Alert and Power Supply Failure Detection Errors from PMC Sysmon alarm[3]. Indicates an alarm condition on any of SUPPLY96 to SUPPLY127.

Define XPM_EVENT_ERROR_MASK_PMCSMON4

Definition

```
#define XPM_EVENT_ERROR_MASK_PMCSMON4(0x00000800U)
```

Description

Error event mask for PMC Temperature Shutdown Alert and Power Supply Failure Detection Errors from PMC Sysmon alarm[4]. Indicates an alarm condition on any of SUPPLY128 to SUPPLY159.

Define XPM_EVENT_ERROR_MASK_PMCSMON8

Definition

```
#define XPM_EVENT_ERROR_MASK_PMCSMON8(0x00008000U)
```

Description

Error event mask for PMC Temperature Shutdown Alert and Power Supply Failure Detection Errors from PMC Sysmon alarm[8]. Indicates an over-temperature alarm.

Define XPM_EVENT_ERROR_MASK_PMCSMON9

Definition

```
#define XPM_EVENT_ERROR_MASK_PMCSMON9(0x00010000U)
```

Description

Error event mask for PMC Temperature Shutdown Alert and Power Supply Failure Detection Errors from PMC Sysmon alarm[9]. Indicates a device temperature alarm.

Define XPM_EVENT_ERROR_MASK_CFI

Definition

```
#define XPM_EVENT_ERROR_MASK_CFI(0x00020000U)
```

Description

Error event mask for CFI Non-Correctable Error.

Define XPM_EVENT_ERROR_MASK_SEUCRC

Definition

```
#define XPM_EVENT_ERROR_MASK_SEUCRC(0x00040000U)
```

Description

Error event mask for CFRAME SEU CRC Error.

Define XPM_EVENT_ERROR_MASK_SEUECC

Definition

```
#define XPM_EVENT_ERROR_MASK_SEUECC(0x00080000U)
```

Description

Error event mask for CFRAME SEU ECC Error.

Define XPM_EVENT_ERROR_MASK_RTCALARM

Definition

```
#define XPM_EVENT_ERROR_MASK_RTCALARM(0x00400000U)
```

Description

Error event mask for RTC Alarm Error.

Define XPM_EVENT_ERROR_MASK_NPLL

Definition

```
#define XPM_EVENT_ERROR_MASK_NPLL(0x00800000U)
```

Description

Error event mask for PMC NPLL Lock Error, This error can be unmasked after the NPLL is locked to alert when the NPLL loses lock.

Define XPM_EVENT_ERROR_MASK_PPLL

Definition

```
#define XPM_EVENT_ERROR_MASK_PPLL(0x01000000U)
```

Description

Error event mask for PMC PPLL Lock Error, This error can be unmasked after the PPLL is locked to alert when the PPLL loses lock.

Define XPM_EVENT_ERROR_MASK_CLKMON

Definition

```
#define XPM_EVENT_ERROR_MASK_CLKMON(0x02000000U)
```

Description

Error event mask for Clock Monitor Errors., Collected from CRP's CLKMON_STATUS register.

Define XPM_EVENT_ERROR_MASK_PMCTO

Definition

```
#define XPM_EVENT_ERROR_MASK_PMCTO(0x04000000U)
```

Description

Error event mask for PMC Interconnect Timeout Errors., Collected from:,Interconnect mission interrupt status register., Interconnect latent status register.,Timeout interrupt status register for SERBs.

Define XPM_EVENT_ERROR_MASK_PMCXMPU

Definition

```
#define XPM_EVENT_ERROR_MASK_PMCXMPU(0x08000000U)
```

Description

Error event mask for PMC XMPU Errors:,Register access error on APB., Read permission violation.,Write permission violation.,Security violation.

Define XPM_EVENT_ERROR_MASK_PMCXPPU

Definition

```
#define XPM_EVENT_ERROR_MASK_PMCXPPU(0x10000000U)
```

Description

Error event mask for PMC XPPU Errors:,Register access error on APB., Master ID not found.,Read permission violation.,Master ID parity error., Master ID access violation.,TrustZone violation.,Aperture parity error.

Define XPM_EVENT_ERROR_MASK_SSIT0

Definition

```
#define XPM_EVENT_ERROR_MASK_SSIT0(0x20000000U)
```

Description

Error event mask for For Master SLR: SSIT Error from Slave SLR1., For Slave SLRs: SSIT Error0 from Master SLR.

Define XPM_EVENT_ERROR_MASK_SSIT1

Definition

```
#define XPM_EVENT_ERROR_MASK_SSIT1(0x40000000U)
```

Description

Error event mask for For Master SLR: SSIT Error from Slave SLR2., For Slave SLRs: SSIT Error1 from Master SLR.

Define XPM_EVENT_ERROR_MASK_SSIT2

Definition

```
#define XPM_EVENT_ERROR_MASK_SSIT2(0x80000000U)
```

Description

Error event mask for For Master SLR: SSIT Error from Slave SLR3., For Slave SLRs: SSIT Error2 from Master SLR.

Error Event Mask for PSM ERR1

Error Events belong to PSM ERR1 Node.

Definitions

Define XPM_EVENT_ERROR_MASK_PS_SW_CR

Definition

```
#define XPM_EVENT_ERROR_MASK_PS_SW_CR(0x00000001U)
```

Description

Error event mask for PS Software can write to trigger register to generate this Correctable Error.

Define XPM_EVENT_ERROR_MASK_PS_SW_NCR

Definition

```
#define XPM_EVENT_ERROR_MASK_PS_SW_NCR(0x00000002U)
```

Description

Error event mask for PS Software can write to trigger register to generate this Non-Correctable Error.

Define XPM_EVENT_ERROR_MASK_PSM_B_CR

Definition

```
#define XPM_EVENT_ERROR_MASK_PSM_B_CR(0x00000004U)
```

Description

Error event mask for PSM Firmware can write to trigger register to generate this Correctable Error.

Define XPM_EVENT_ERROR_MASK_PSM_B_NCR

Definition

```
#define XPM_EVENT_ERROR_MASK_PSM_B_NCR(0x00000008U)
```

Description

Error event mask for PSM Firmware can write to trigger register to generate this Non-Correctable Error.

Define XPM_EVENT_ERROR_MASK_MB_FATAL

Definition

```
#define XPM_EVENT_ERROR_MASK_MB_FATAL(0x00000010U)
```

Description

Error event mask for Or of MB Fatal1, Fatal2, Fatal3 Error.

Define XPM_EVENT_ERROR_MASK_PSM_CR

Definition

```
#define XPM_EVENT_ERROR_MASK_PSM_CR(0x00000020U)
```

Description

Error event mask for PSM Correctable.

Define XPM_EVENT_ERROR_MASK_PSM_NCR

Definition

```
#define XPM_EVENT_ERROR_MASK_PSM_NCR(0x00000040U)
```

Description

Error event mask for PSM Non-Correctable.

Define XPM_EVENT_ERROR_MASK_OCM_ECC

Definition

```
#define XPM_EVENT_ERROR_MASK_OCM_ECC(0x00000080U)
```

Description

Error event mask for Non-Correctable ECC Error during an OCM access.

Define XPM_EVENT_ERROR_MASK_L2_ECC

Definition

```
#define XPM_EVENT_ERROR_MASK_L2_ECC(0x00000100U)
```

Description

Error event mask for Non-Correctable ECC Error during APU L2 Cache access.

Define XPM_EVENT_ERROR_MASK_RPU_ECC

Definition

```
#define XPM_EVENT_ERROR_MASK_RPU_ECC(0x00000200U)
```

Description

Error event mask for ECC Errors during a RPU memory access. Floating-point operation exceptions. RPU REG APB error.

Define XPM_EVENT_ERROR_MASK_RPU_LS

Definition

```
#define XPM_EVENT_ERROR_MASK_RPU_LS(0x00000400U)
```

Description

Error event mask for RPU Lockstep Errors from R5_0. The Lockstep error is not initialized until RPU clock is enabled; therefore, error outcomes are masked by default and are expected to be unmasked after processor clock is enabled and before its reset is released.

Define XPM_EVENT_ERROR_MASK_RPU_CCF

Definition

```
#define XPM_EVENT_ERROR_MASK_RPU_CCF(0x00000800U)
```

Description

Error event mask for RPU Common Cause Failures ORed together. The CCF Error register with the masking capability has to reside in the RPU.

Define XPM_EVENT_ERROR_MASK_GIC_AXI

Definition

```
#define XPM_EVENT_ERROR_MASK_GIC_AXI(0x00001000U)
```

Description

Error event mask for APU GIC AXI Error by the AXI4 master port, such as SLVERR or DECERR.

Define XPM_EVENT_ERROR_MASK_GIC_ECC

Definition

```
#define XPM_EVENT_ERROR_MASK_GIC_ECC(0x00002000U)
```

Description

Error event mask for APU GIC ECC Error, a Non-Correctable ECC error occurred in any ECC-protected RAM.

Define XPM_EVENT_ERROR_MASK_APLL_LOCK

Definition

```
#define XPM_EVENT_ERROR_MASK_APLL_LOCK(0x00004000U)
```

Description

Error event mask for APLL Lock Errors. The error can be unmasked after the PLL is locked to alert when the PLL loses lock.

Define XPM_EVENT_ERROR_MASK_RPLL_LOCK

Definition

```
#define XPM_EVENT_ERROR_MASK_RPLL_LOCK(0x00008000U)
```

Description

Error event mask for RPLL Lock Errors. The error can be unmasked after the PLL is locked to alert when the PLL loses lock.

Define XPM_EVENT_ERROR_MASK_CPM_CR

Definition

```
#define XPM_EVENT_ERROR_MASK_CPM_CR(0x00010000U)
```

Description

Error event mask for CPM Correctable Error.

Define XPM_EVENT_ERROR_MASK_CPM_NCR

Definition

```
#define XPM_EVENT_ERROR_MASK_CPM_NCR(0x00020000U)
```

Description

Error event mask for CPM Non-Correctable Error.

Define XPM_EVENT_ERROR_MASK_LPD_APB

Definition

```
#define XPM_EVENT_ERROR_MASK_LPD_APB(0x00040000U)
```

Description

Error event mask for LPD APB Errors from:,IPI REG,USB2 REG,CRL REG,LPD AFIFM4 REG,LPD IOU REG, LPD IOU SECURE SLCR REG,LPD SLCR REG,LPD SLCR SECURE REG.

Define XPM_EVENT_ERROR_MASK_FPD_APB

Definition

```
#define XPM_EVENT_ERROR_MASK_FPD_APB(0x00080000U)
```

Description

Error event mask for FPD APB Errors from:,FPD AFIFM0 REG,FPD AFIFM2 REG,FPD SLCR REG,FPD SLCR SECURE REG, CRF REG.

Define XPM_EVENT_ERROR_MASK_LPD_PAR

Definition

```
#define XPM_EVENT_ERROR_MASK_LPD_PAR(0x00100000U)
```

Description

Error event mask for Data parity errors from the interfaces connected to the LPD interconnect.

Define XPM_EVENT_ERROR_MASK_FPD_PAR

Definition

```
#define XPM_EVENT_ERROR_MASK_FPD_PAR(0x00200000U)
```

Description

Error event mask for Data parity errors from the interfaces connected to the FPD interconnect.

Define XPM_EVENT_ERROR_MASK_IOU_PAR

Definition

```
#define XPM_EVENT_ERROR_MASK_IOU_PAR(0x00400000U)
```

Description

Error event mask for LPD IO Peripheral Unit Parity Error.

Define XPM_EVENT_ERROR_MASK_PSM_PAR

Definition

```
#define XPM_EVENT_ERROR_MASK_PSM_PAR(0x00800000U)
```

Description

Error event mask for Data parity errors from the interfaces connected to the PSM interconnect.

Define XPM_EVENT_ERROR_MASK_LPD_TO

Definition

```
#define XPM_EVENT_ERROR_MASK_LPD_TO(0x01000000U)
```

Description

Error event mask for LPD Interconnect Timeout errors., Collected from:,Timeout errors at the slaves connected to the LPD interconnect.,Address decode error.,Interconnect mission errors for the slaves connected to the LPD interconnect.

Define XPM_EVENT_ERROR_MASK_FPD_TO

Definition

```
#define XPM_EVENT_ERROR_MASK_FPD_TO(0x02000000U)
```

Description

Error event mask for FPD Interconnect Timeout errors., Collected from:,Coresight debug trace alarms.,Timeout errors at the slaves connected to the FPD interconnect.,Address decode error., Data parity errors on the interfaces connected to the FPD interconnect.

Define XPM_EVENT_ERROR_MASK_PSM_TO

Definition

```
#define XPM_EVENT_ERROR_MASK_PSM_TO(0x04000000U)
```

Description

Error event mask for PSM Interconnect Timeout Errors., Collected from:,Interconnect mission errors for PSM_LOCAL slave or PSM_GLOBAL slave or MDM slave or LPD interconnect or PSM master., Interconnect latent errors for PSM_LOCAL slave or PSM_GLOBAL slave or MDM slave or LPD interconnect or PSM master., Timeout errors at the slaves connected to the PSM interconnect.

Define XPM_EVENT_ERROR_MASK_XRAM_CR

Definition

```
#define XPM_EVENT_ERROR_MASK_XRAM_CR(0x08000000U)
```

Description

Error event mask for XRAM Correctable error. Only applicable in devices that have XRAM.

Define XPM_EVENT_ERROR_MASK_XRAM_NCR

Definition

```
#define XPM_EVENT_ERROR_MASK_XRAM_NCR(0x10000000U)
```

Description

Error event mask for XRAM Non-Correctable error. Only applicable in devices that have XRAM.

Error Event Mask for PSM ERR2

Error Events belong to PSM ERR2 Node.

Definitions

Define XPM_EVENT_ERROR_MASK_LPD_SWDT

Definition

```
#define XPM_EVENT_ERROR_MASK_LPD_SWDT(0x00000001U)
```

Description

Error event mask for Error from Watchdog Timer in the LPD Subsystem.

Define XPM_EVENT_ERROR_MASK_FPD_SWDT

Definition

```
#define XPM_EVENT_ERROR_MASK_FPD_SWDT(0x00000002U)
```

Description

Error event mask for Error from Watchdog Timer in the FPD Subsystem.

Define XPM_EVENT_ERROR_MASK_LPD_XMPU

Definition

```
#define XPM_EVENT_ERROR_MASK_LPD_XMPU(0x00040000U)
```

Description

Error event mask for LPD XMPU Errors:,Register access error on APB., Read permission violation.,Write permission violation.,Security violation.

Define XPM_EVENT_ERROR_MASK_LPD_XPPU

Definition

```
#define XPM_EVENT_ERROR_MASK_LPD_XPPU(0x00080000U)
```

Description

Error event mask for LPD XPPU Errors:,Register access error on APB., Master ID not found.,Read permission violation.,Master ID parity error., Master ID access violation.,TrustZone violation.,Aperture parity error.

Define XPM_EVENT_ERROR_MASK_FPD_XMPU

Definition

```
#define XPM_EVENT_ERROR_MASK_FPD_XMPU(0x00100000U)
```

Description

Error event mask for FPD XMPU Errors:,Register access error on APB., Read permission violation.,Write permission violation.,Security violation.

Error Event Mask for Software error events

Error Events belong to SW ERR Node.

Definitions

Define XPM_EVENT_ERROR_MASK_HB_MON_0

Definition

```
#define XPM_EVENT_ERROR_MASK_HB_MON_0(0x00000001U)
```

Description

Error event mask for Software error events

Health Boot Monitoring errors

Define XPM_EVENT_ERROR_MASK_HB_MON_1

Definition

```
#define XPM_EVENT_ERROR_MASK_HB_MON_1(0x00000002U)
```

Description

Error event mask for Software error events

Health Boot Monitoring errors

Define XPM_EVENT_ERROR_MASK_HB_MON_2

Definition

```
#define XPM_EVENT_ERROR_MASK_HB_MON_2(0x00000004U)
```

Description

Error event mask for Software error events

Health Boot Monitoring errors

Define XPM_EVENT_ERROR_MASK_HB_MON_3

Definition

```
#define XPM_EVENT_ERROR_MASK_HB_MON_3(0x00000008U)
```

Description

Error event mask for Software error events

Health Boot Monitoring errors

Library Parameters in MSS File

The XilPM Library can be integrated with a system using the following snippet in the Microprocessor Software Specification (MSS) file:

The list of flags is as follows:

- rpu0_as_power_management_master
- rpu1_as_power_management_master
- apu_as_power_management_master
- rpu0_as_reset_management_master
- rpu1_as_reset_management_master
- apu_as_reset_management_master
- rpu0_as_overlay_config_master
- rpu1_as_overlay_config_master
- apu_as_overlay_config_master

The flags have two values:

- TRUE (default value): Master(APU/RPU0/RPU1) is enabled as power/reset management master
- FALSE: Master(APU/RPU0/RPU1) is disabled as power/reset management master

***Note*:** Flag rpu1_as_power_management_master and rpu1_as_reset_management_master are only counted if the RPU is in split mode.

Data Structure Index

The following is a list of data structures:

pm_acknowledge

Declaration

```
typedef struct
{
  volatile u8 received,
  u32 node,
  XStatus status,
  u32 opp,
  volatile bool received,
  enum XPmNodeId node
} pm_acknowledge;
```

Table 139: **Structure pm_acknowledge member description**

Member	Description
received	Has acknowledge argument been received?
node	Node argument about which the acknowledge is
status	Acknowledged status
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
{
  volatile u8 received,
  enum XPmSuspendReason reason,
  u32 latency,
  u32 state,
  u32 timeout,
  volatile bool received
} pm_init_suspend;
```

Table 140: **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 141: **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_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)
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,
  u32 event,
  u32 received_event,
  u32 flags,
  volatile u32 oppoint,
  volatile u32 received,
  struct XPm_Ntfier * next,
  enum XPmNodeId node,
  enum XPmNotifyEvent event
} 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)

Table 143: **Structure XPm_Notifier member description** *(cont'd)*

Member	Description
event	Event argument (the event type to receive notifications about)
received_event	Event received from PLM)
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)
event	Event argument (the event type to receive notifications about)

XILINX

Appendix A

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

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright