

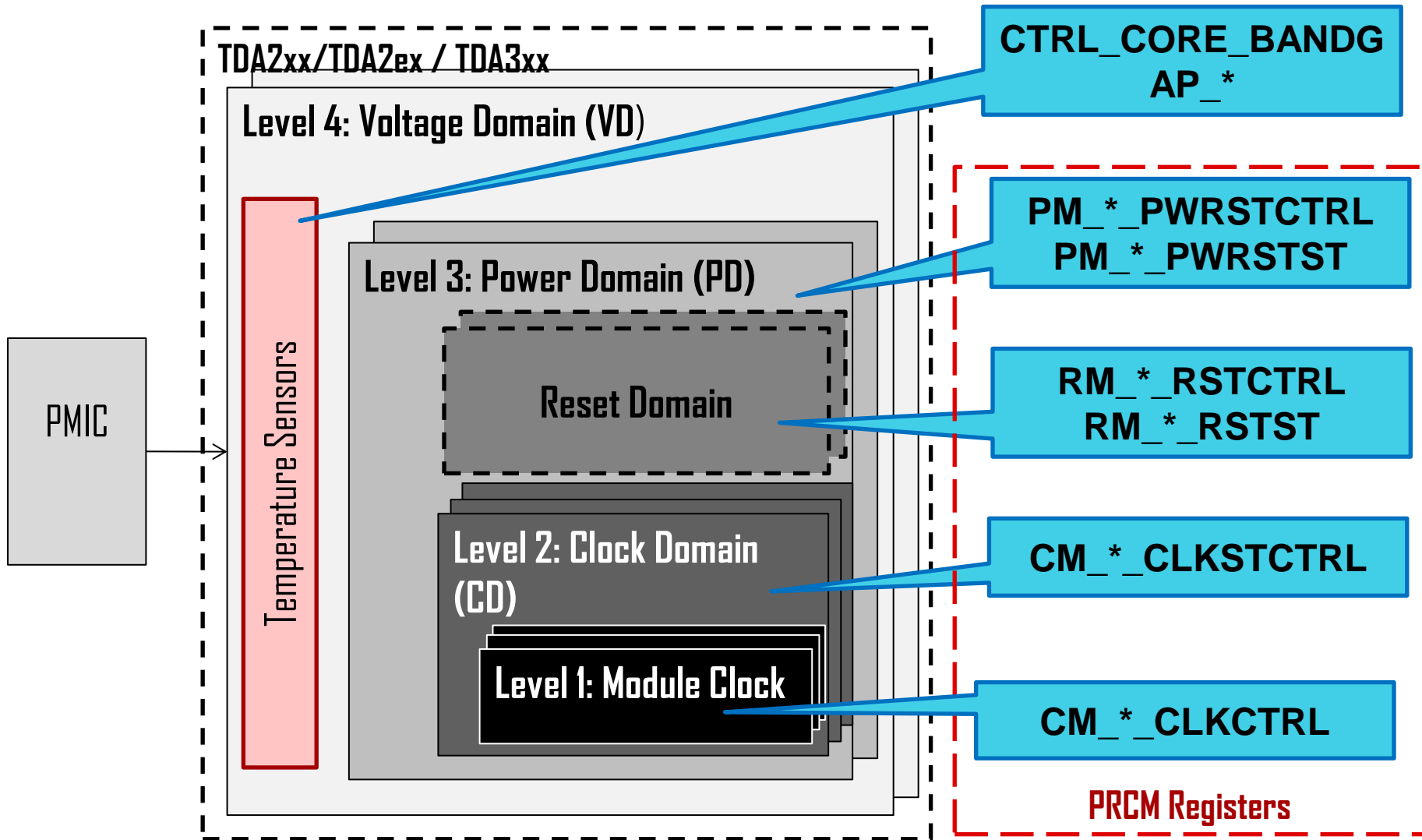
# **Power Management Software Overview (TDA2xx/TDA2ex/TDA3xx)**

# Agenda

- PRCM Hardware Overview
  - Power, Clock Domains, Module Level
- How to keep Power consumption in check?
  - Initialize the system:
    - Set power state for different Modules.
    - Set the clock rate for CPUs.
  - Dynamic Power Management
  - Software Thermal Management
- Power Management (PM) Software Stack Overview
  - PMHAL
  - PMLIB

# PRCM Hardware Overview

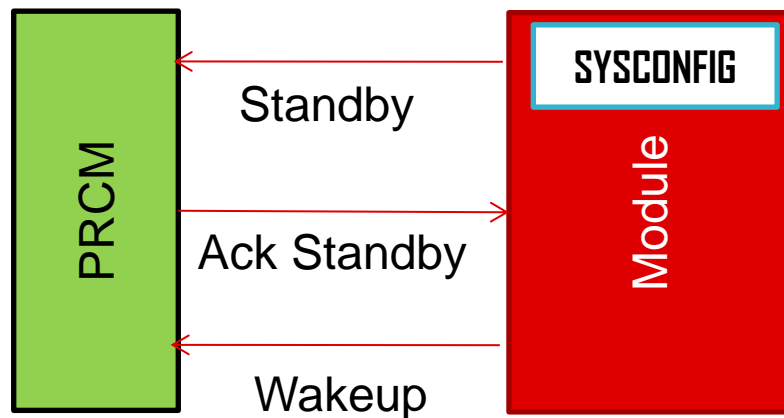
# PRCM Hardware Overview



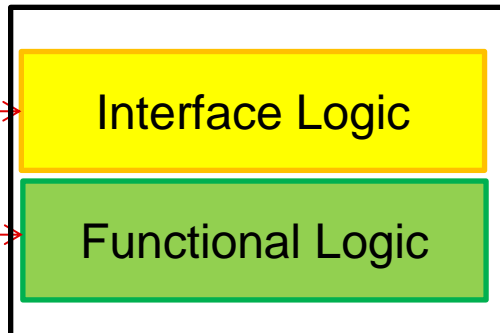
# Module PM

## Master Standby

- Valid for Initiators to the Interconnect.
- When Master does not want clocks configure IP level SYSCONFIG MIDDLEMODE or STANDBYMODE.
- PRCM reflects status in CLKCTRL[x].STBYST

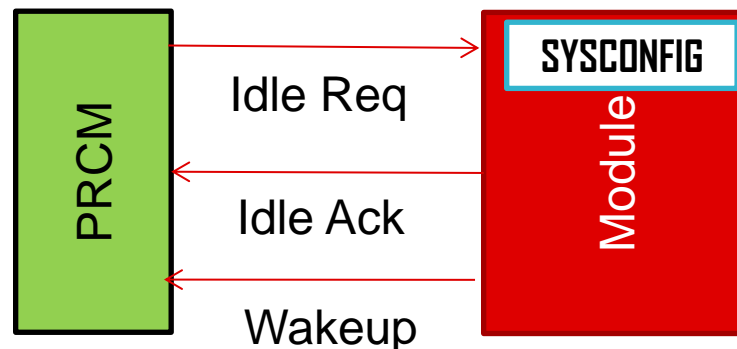


Interface clock  
(ICLK) →  
Functional clock  
(FCLK) →



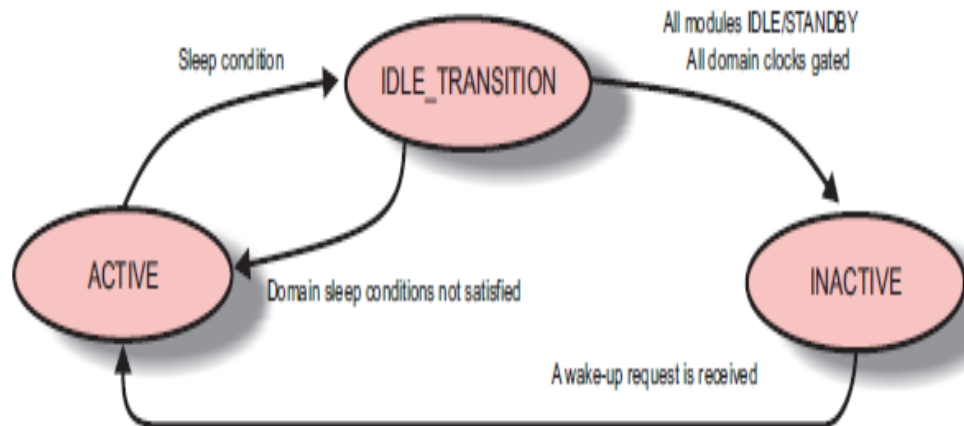
## Slave Idle

- Valid for modules which respond to requests.
- Configure PRCM register CLKCTRL.MODULEMODE.
- Configure IP level SYSCONFIG SIDLEMODE or IDLEMODE.
- PRCM reflects status in CLKCTRL[x].IDLEST



# Clock Domain (CD) PM

- Clock domain allows control of the dynamic/active power consumption of the device.
- Device has multiple Clock Domains. Each Clock domain may have one or more modules.



## Rel Condition For INACTIVE

AND All master modules in the clock domain are in STANDBY state.

No wake-up request is asserted by any module of the clock domain.

No static domain dependency from any other domain is active.

The SW\_SLEEP/HW\_AUTO clock transition mode is set for the clock domain (CLKTRCTRL = 0x1 / 0x3).

## Rel Condition For ACTIVE

OR The SW\_WKUP clock transition mode for the clock domain is set (CLKTRCTRL = 0x2).

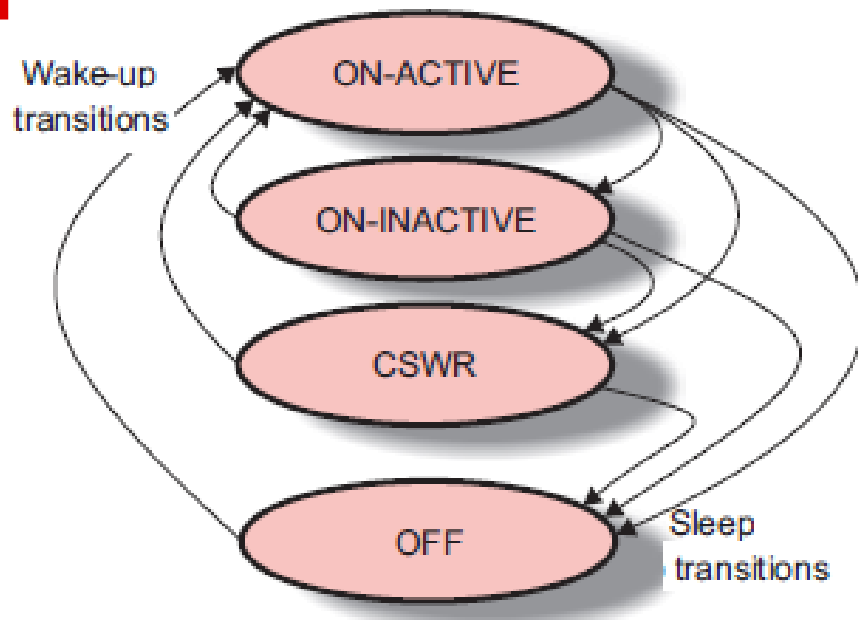
At least one wake-up request is asserted by one of the modules of the clock domain

At least one static dependency from another clock domain is active.

Clock Activity State can be read from  
CM\_<CD>\_CLKSTCTRL.CLKACTIVITY\_\*\_F/ICLK

# Power Domain (PD) PM

- Power Domain allows for control of leakage power consumption of the device.
- If no clock domains are on the PD can go to ON-INACTIVE, RETENTION or OFF state.
- If any one clock domain is active then the power domain would remain on.



PD State	Logic State	Memory State	CD State
ON-ACTIVE	ON	ON	ACTIVE
ON-INACTIVE	ON	PWRSTCTRL.<MEM>_ONSTATE	IDLE
CSWR	ON	PWRSTCTRL.<MEM>_RETSTATE	IDLE
OFF	OFF	OFF	IDLE

Context Maintained

Context Lost

# TDA2xx/2ex vs TDA3xx (PRCM)

- TDA2xx has 5 Voltage Domains (VD\_CORE, VD\_MPU, VD\_DSPEVE, VD\_GPU, VD\_IVA); TDA3xx has 2 voltage domains (VD\_CORE, VD\_DSPEVE)
- TDA3xx does not support Adaptive Body Bias (ABB)
- TDA2xx has 5 temperature sensors (VD\_CORE, VD\_MPU, VD\_DSPEVE, VD\_GPU, VD\_IVA); TDA3xx has 1 temperature sensor (VD\_CORE)
- TDA2xx has different clock tree structure than TDA3xx due to DPLL changes.



# How to keep Power consumption in check?

# System Initialization

# Initializing the system

- Ensure modules not getting used are turned off.

Module Name	Reset Power State	SBL Desired Action
MPU C0 & C1	ON	Force Off C1 when not used
IPU, DSP1 & 2	OFF	Initialize core when valid application image is present. Power Off if not.
EVE1 /EVE2	ON (Clock Gated)	Initialize core when valid application image is present. Power Off if not.
MMC1, IEEE1500_2_OCP	ON	Disable Module if not used

- Modules like MMC2, MLB\_SS, SATA, OCP2SCP1, OCP2SCP3, USB\_OTG\_SS1, USB\_OTG\_SS2, USB\_OTG\_SS3, USB\_OTG\_SS4, PCIESS1, PCIESS2 etc are disabled by default..

# Initializing the system

- **System Configuration:** (Set the Power and Clock State for different modules)
  - Program the module to any of the 3 states:
    - **DISABLED** – Lowest Power Configuration.
    - **AUTO CLOCK GATE (AUTO\_CG)** – Clocks disabled when module not used.
    - **ALWAYS ENABLED** – Highest Power Configuration
- Takes care of Power Domain, Clock Domain, Module level (optional clocks, sys-config) and Static dependency configuration.
- Additionally takes care of reset configurations.
- Example: `starterware \examples\pm\systemconfig\main tda2xx/tda3xx.c`
- **Note:** This API does not take care of dependencies between enabling modules.

# System Configuration API

```
pmErrCode_t PMLIBSysConfigSetPowerState(  
    const pmlibSysConfigPowerStateParams_t *inputTable,  
    uint32_t                                numConfig,  
    uint32_t                                timeout,  
    pmlibSysConfigErrReturn_t               *resultReturn);
```

Module Name	Power State
Module 1	Always Enabled
Module 2	Disabled
Module 3	Auto CG

# Initializing the system

- **Clock Rate** (Setting the clock rate for different CPUs/Peripherals)

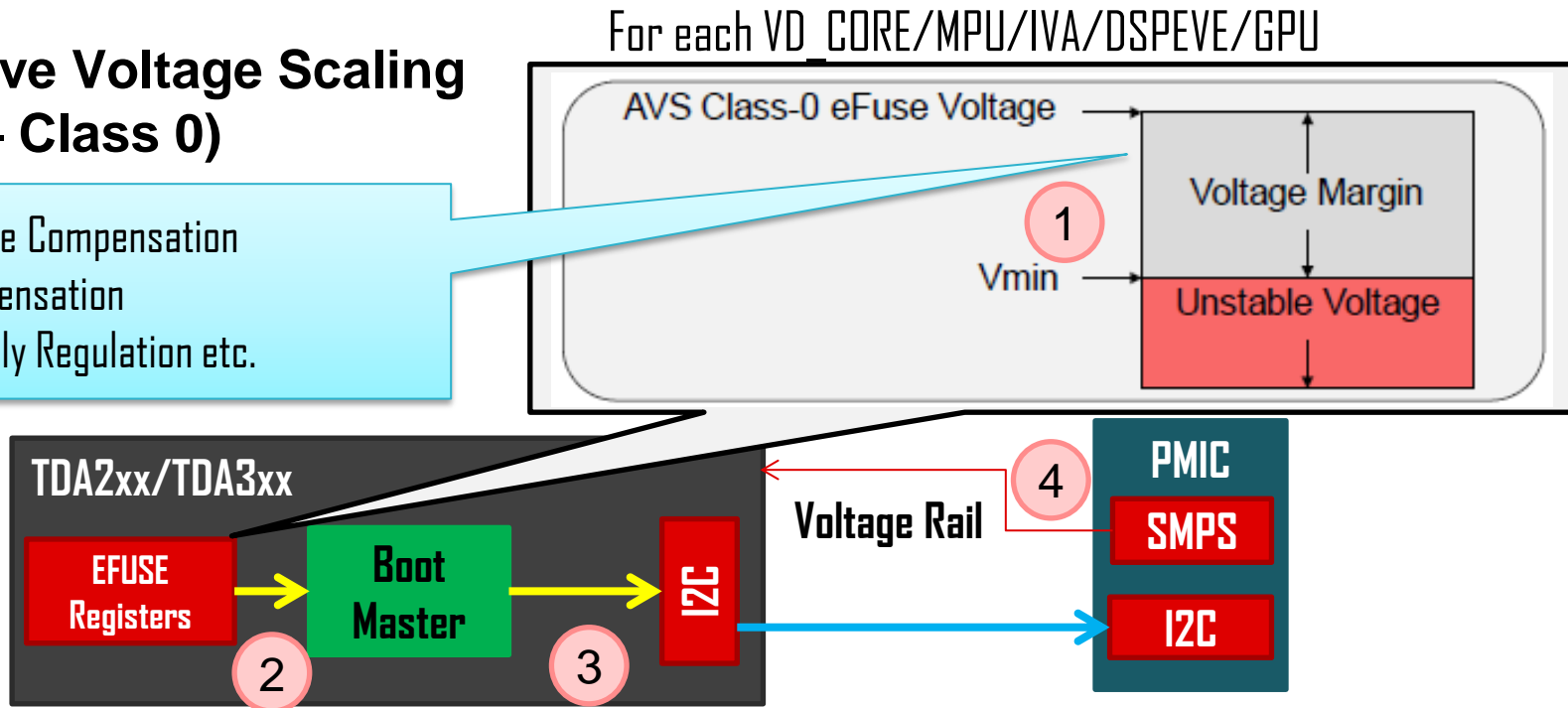
```
pmErrCode_t PMLIBClkRateSet(pmhalPrcmModuleId_t modId,  
                             pmhalPrcmClockId_t  clkId,  
                             uint32_t             clkRate);  
  
pmErrCode_t PMLIBClkRateGet(pmhalPrcmModuleId_t modId,  
                             pmhalPrcmClockId_t  clkId,  
                             uint32_t             *clkRate);
```

- Takes care of required OPP change for the given frequency.
- Internal database maintained to find the corresponding DPLL configurations for the given frequency.
- “Generic Clk ID” support provided to allow the user to not have to remember the clock name for each and every module.

# Set Optimal Voltage for Lower Power Dissipation

- **Adaptive Voltage Scaling (AVS – Class 0)**

Temperature Compensation  
Aging Compensation  
Power Supply Regulation etc.

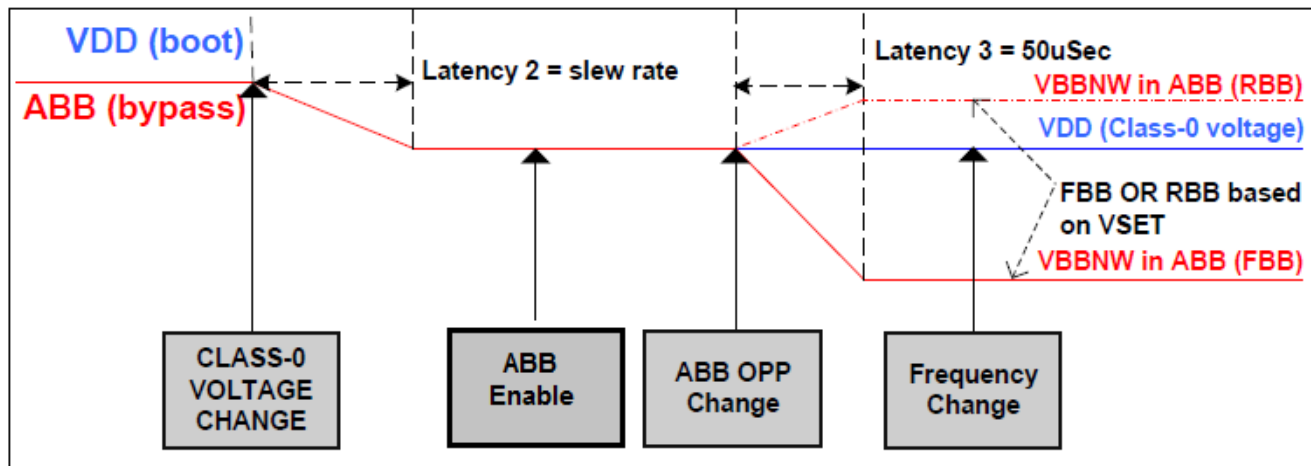
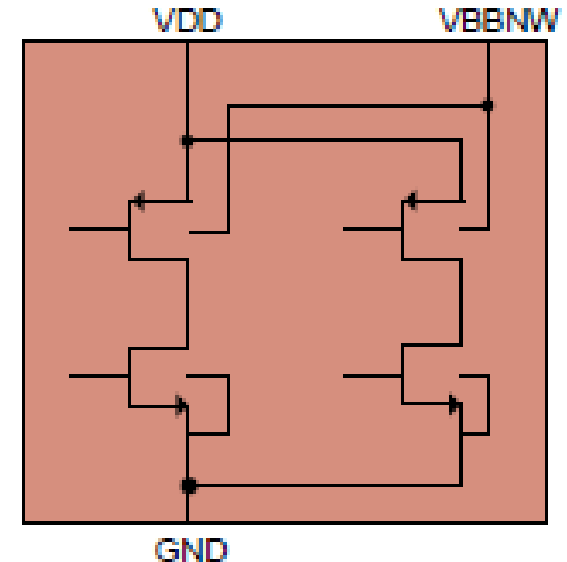


- AVS should be executed before other domains are taken out of reset and before their DPLLs are locked. (in SBL)
- Reduce the risk of Hot devices entering into a thermal condition.
- Ensure reliability and to guarantee that the lifetime POHs are achieved.

# Increase performance and reduce leakage

- **Adaptive Body Bias (ABB)**
- Apply a voltage to the NWEELL of the PMOS transistors to change the Threshold Voltage.
- Configure at Boot time.

Reverse Body Bias (RBB)	Forward Body Bias (FBB)
$V_{BBNW} > V_{DD}$	$V_{BBNW} < V_{DD}$
For Strong Samples	For Weak Samples
Increase $V_{th}$	Decrease $V_{th}$
Reduce Leakage	Increase Performance



Valid only for  
TDA2xx



# APIs to Set AVS & ABB at the right OPP

```
pmErrCode_t      retVal = PM_SUCCESS;
pmhalVmOppId_t   oppId;
const pmhalPmicOperations_t *pmicOps;

/* Enable I2C1 for PMIC Communication
 * Force Wake-up clock domain l4per*/
PMHALCMSSetCdClockMode(
    PMHAL_PRCM_CD_L4PER,
    PMHAL_PRCM_CD_CLKTRNMODES_SW_WAKEUP,
    PM_TIMEOUT_INFINITE);

PMHALModuleModeSet(PMHAL_PRCM_MOD_I2C1,
    PMHAL_PRCM_MODULE_MODE_ENABLED,
    PM_TIMEOUT_INFINITE);

/* Get the pmic ops and register with
the pmic interface. */
pmicOps = PMHALTps65917GetPMICOps();
retVal = PMHALPmicRegister(pmicOps);
```

## PMHAL:

- starterware\_include\pm\pmhal\pmhal\_vm.h
- starterware\_include\pm\pmhal\pmhal\_pmic.h

```
if (PM_SUCCESS == retVal)
{
    retVal = PMHALVMSetOpp(
        PMHAL_PRCM_VD_MPU, oppId,
        PM_TIMEOUT_INFINITE);

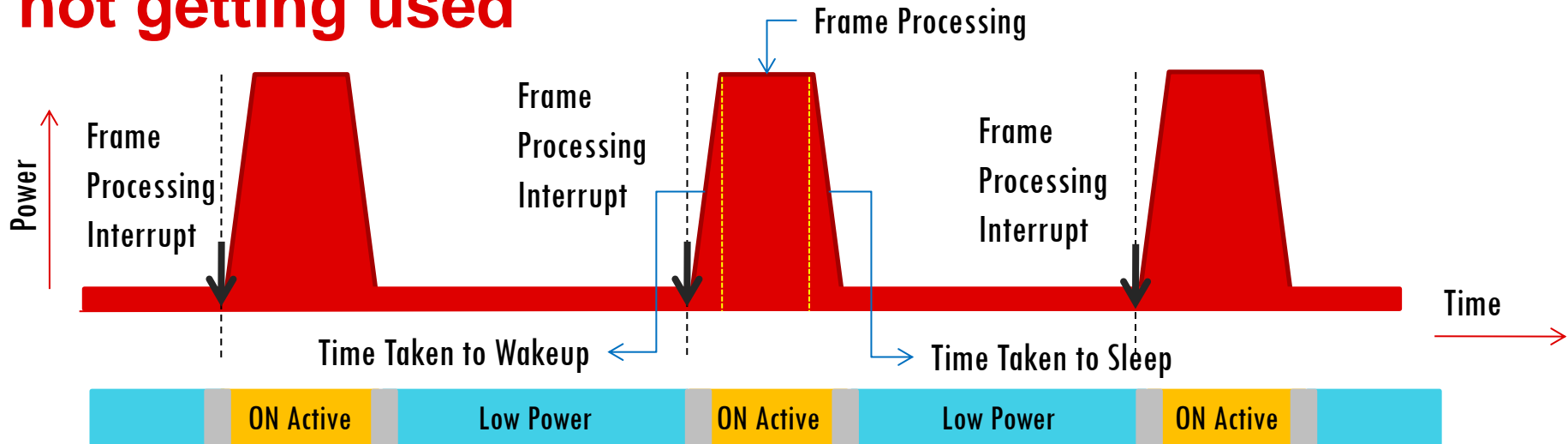
    /* VD_CORE can only support OPP_NOM
    */
    retVal |= PMHALVMSetOpp(
        PMHAL_PRCM_VD_CORE,
        PMHAL_VM_OPP_NOM,
        PM_TIMEOUT_INFINITE);

    /* Set the voltage for
    * PMHAL_PRCM_VD_IVAHD,
    * PMHAL_PRCM_VD_DSPEVE
    * and PMHAL_PRCM_VD_GPU.
    */
    for (vdId = PMHAL_PRCM_VD_IVAHD;
        vdId < PMHAL_PRCM_VD_RTC;
        vdId++)
    {
        retVal |= PMHALVMSetOpp(vdId,
            oppId,
            PM_TIMEOUT_INFINITE);
    }
}
```



# Dynamic Power Management

# Reduce power consumption when a CPU Core is not getting used



- Context of the CPU is maintained.
- Configure Interrupts which would act as wakeup events.
- Define the lowest power state of the CPU when not processing.
  - **MPU** : Closed Switch Retention - starterware \examples\pm\cpuidle\main a15host.c
  - **IPU** : Auto Clock Gate - starterware \examples\pm\cpuidle\main m4.c
  - **DSP** : Auto Clock Gate - starterware \examples\pm\cpuidle\main c66x.c
  - **EVE** : Auto Clock Gate - starterware \examples\pm\arp32 cpuidle\main arp32.c

# MPU CPU Idle Sequence

```
MPU_WUGEN_1_DisableAll();
```

```
/* Flushing the DCache is required to
   ensure CPU0 does not get pipeline
   * stalled when the cache is enabled
   later and the cache invalidate is
   * performed. */
```

```
CP15DCacheCleanFlush();
```

```
PMLIBCpu1ForcedOff();
```

**CPU1 Forced  
OFF in SBL**

## 2 One Time CPU 0 Initialization (Beginning of Application)

```
/* 1st param indicates the type of the
   ramp 0 - Slow Ramp up, 1 - Fast
   Ramp up, 2nd param The value set in
   this field determines the slow
   ramp-up time */
```

```
pmhalMpuLprmHgRampParams_t hgRampParam
= {1, 0};
```

**Programs MPU\_PRCM**

```
/* Enable Hg/FastRampup in Retention*/
```

```
PMHALMpuLprmSetHgRampParams(  
                                &hgRampParam);
```

```
PMHALMpuLprmSetMercuryRetention();
```

```
pmlibSysConfigPowerStateParams_t
```

```
inputTable = {
```

```
PMHAL_PRCM_MOD_MPU,
```

```
PMLIB_SYS_CONFIG_AUTO_CG};
```

```
status = PMLIBSysConfigSetPowerState(  
&inputTable, 1, PM_TIMEOUT_NOWAIT,  
NULL);
```

## 3 Call @ Run Time between Frames (SYSBIOS Idle Task)

```
/* Configuring enabled Interrupts to  
be wakeup capable */
```

```
MPU_WUGEN_0_Interrupt_Lookup();
```

```
PMLIBCpuIdle(  
PMHAL_PRCM_PD_STATE_RETENTION);
```

PMLIB: starterware\\_include\pm\pmlib\pmlib\_cpuidle.h  
PMHAL: starterware\\_include\pm\pmhal\pmhal\_mpu\_lprm.h  
WUGEN: starterware\\_include\armv7a\tda2xx\mpu\_wugen.h

# DSP CPU Idle Sequence

One Time DSP Initialization

1

```
pmlibSysConfigPowerStateParams_t inputTable = {  
    PMHAL_PRCM_MOD_DSP1, PMLIB_SYS_CONFIG_AUTO_CG};
```

```
status = PMLIBSysConfigSetPowerState(  
    &inputTable, 1, PM_TIMEOUT_NOWAIT, NULL);
```

```
/* C66x CorePac has an additional field to enable power down mode*/  
PMLIBSetCorepacPowerDown((uint32_t) 1U);
```

2

```
/* Configuring enabled Interrupts to be wakeup capable */  
DSP_WUGEN_IRQ_Interrupt_Lookup();    Call @ Run Time between Frames (SYSBIOS Idle Task)  
/* Idle Instruction and sysconfig configuration. Parameter is dummy  
*/  
status = PMLIBCpuIdle(PMHAL_PRCM_PD_STATE_ON_INACTIVE);
```

Dummy Parameter for  
DSP/EVE/IPU

PMLIB: starterware\_include\pm\pmlib\pmlib\_cpuidle.h  
WUGEN: starterware\_include\c66x\dsp\_wugen.h

# IPU CPU Idle Sequence

```
/* Set IPU to Auto clock Gate*/
pmlibSysConfigPowerStateParams_t inputTable = {
    PMHAL_PRCM_MOD_IPU1, PMLIB_SYS_CONFIG_AUTO_CG};

status = PMLIBSysConfigSetPowerState(
    &inputTable, 1, PM_TIMEOUT_NOWAIT, NULL);
```

One Time IPU Initialization

1

```
#ifdef TDA3XX_FAMILY_BUILD
/*This is required as the force override bit CTRL_CORE_SEC_IPU_WAKEUP
 * does not set the right values for the PRCM registers and when the
 * override is lifted then cores are left in a bad power and reset state.
 */
PMHALResetRelease(PMHAL_PRCM_RG_IPU1_CPU0_RST, PM_TIMEOUT_NOWAIT);
PMHALResetRelease(PMHAL_PRCM_RG_IPU1_RST, PM_TIMEOUT_NOWAIT);
retVal += (int32_t) PMHALModuleModeSet(PMHAL_PRCM_MOD_IPU1,
                                         PMHAL_PRCM_MODULE_MODE_AUTO,
                                         PM_TIMEOUT_NOWAIT);

#endif
```

Special Care about for TDA3xx

2

```
PMHALResetAssert(PMHAL_PRCM_RG_IPU1_CPU1_RST);
```

3

Only when CPU1 not Used

```
IPU_WUGEN_Interrupt_Lookup();
retVal = (int32_t) PMLIBCpuIdle(PMHAL_PRCM_PD_STATE_RETENTION);
```

4

Call @ Run Time (SYSBIOS Idle Task)

Dummy Parameter  
for DSP/EVE/IPU

PMHAL: starterware\include\pm\pmhal\pmhal\_rm.h  
WUGEN: starterware\include\armv7m\ipu\_wugen.h



# EVE CPU Idle Sequence

```
pmlibSysConfigPowerStateParams_t inputTable = {  
    PMHAL_PRCM_MOD_EVE1, PMLIB_SYS_CONFIG_AUTO_CG};
```

One Time EVE Initialization

1

```
status = PMLIBSysConfigSetPowerState(  
    &inputTable, 1, PM_TIMEOUT_NOWAIT, NULL);
```

```
ARP32_WUGEN_IRQ_Interrupt_Lookup();
```

Call @ Run Time between Frames (SYSBIOS Idle Task)

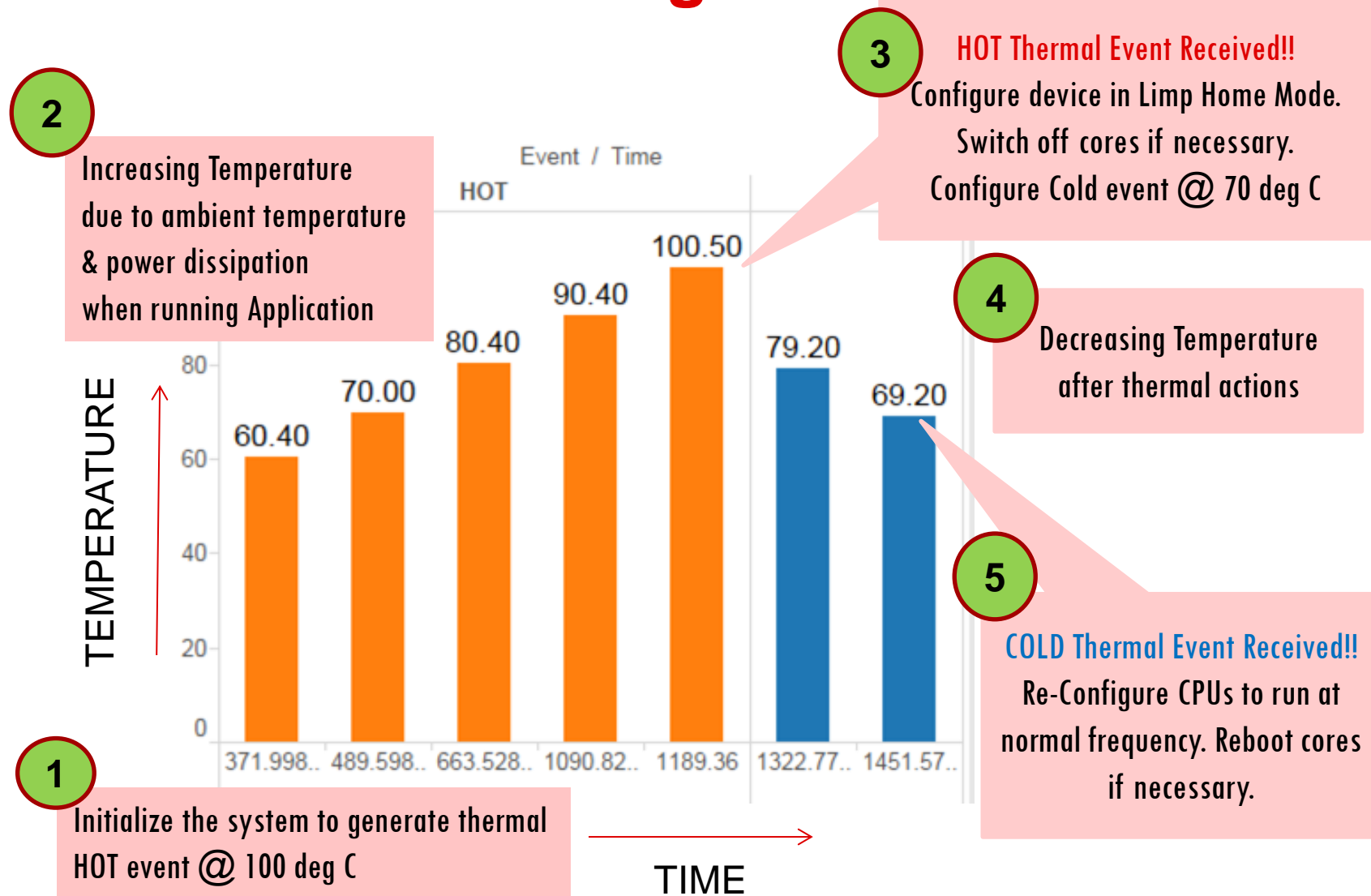
```
/* Program Force Standby for the EDMA TCs */  
HW_WR_REG32(SOC_EVE_EDMA_TC0_BASE + EDMA_TC_SYSCONFIG, 0x0);  
HW_WR_REG32(SOC_EVE_EDMA_TC1_BASE + EDMA_TC_SYSCONFIG, 0x0);  
  
status = PMLIBCpuIdle(PMHAL_PRCM_PD_STATE_ON_ACTIVE);  
  
/* Program Smart Standby for the EDMA TCs after coming out of Idle*/  
HW_WR_REG32(SOC_EVE_EDMA_TC0_BASE + EDMA_TC_SYSCONFIG, 0x28);  
HW_WR_REG32(SOC_EVE_EDMA_TC1_BASE + EDMA_TC_SYSCONFIG, 0x28);
```

2

# Software Thermal Management

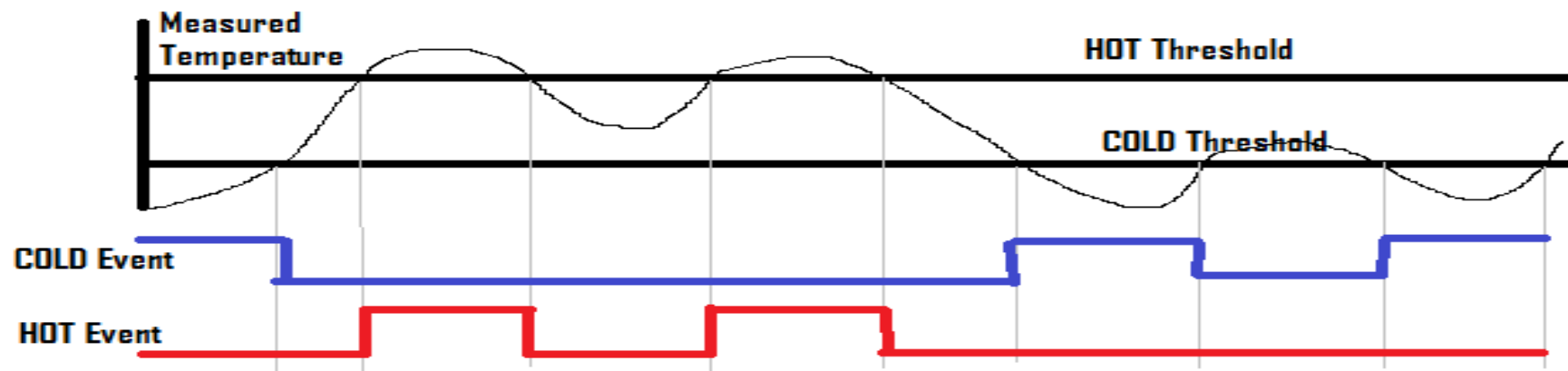
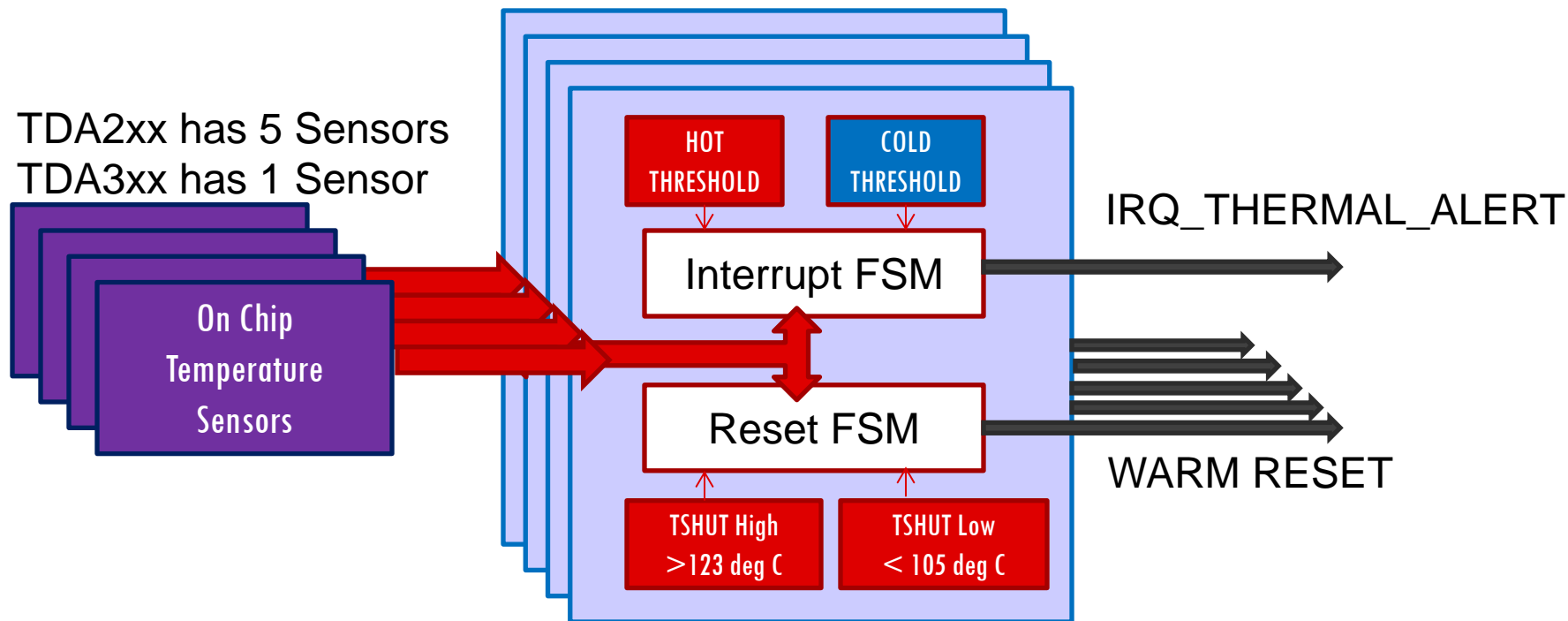


# Software Thermal Management



# Alert regarding a thermal event

TDA2xx has 5 Sensors  
TDA3xx has 1 Sensor



# Alert Regarding a Thermal Event

1

## One Time Thermal Event Initialization

```
/* Registering TimerIsr */
Intc_IntRegister(IRQ_NUM,
    (IntrFuncPtr) TemperatureSensorIsr,
    NULL);

/* temp in milli deg C */
HOT_EVT_TEMP_THRESH = 100000;
/* 100 deg C */

PMHALBgapSetHotThreshold(voltId,
    HOT_EVT_TEMP_THRESH);
```

## Configure HOT/Cold Threshold Based on Thermal Actions

```
COLD_EVT_TEMP_THRESH = 70000;
/* 70 deg C */
```

```
PMHALBgapSetColdThreshold(voltId,
    COLD_EVT_TEMP_THRESH);
```

```
/* temp in milli deg C */
HOT_EVT_TEMP_THRESH = 110000;
/* 110 deg C */
```

```
PMHALBgapSetHotThreshold(voltId,
    HOT_EVT_TEMP_THRESH);
```

**HOT Event!!**

TemperatureSensorIsr:

2

Disable  
Temperature  
IRQ

3

Alter Hot and  
Cold  
Temperature  
Threshold

4

Clear Pending  
Temperature  
IRQ

5

Enable  
Temperature  
IRQ

6

Take necessary  
thermal Action

Example: `starterware \examples\pm\junction temp sensor\main tda2xx.c`

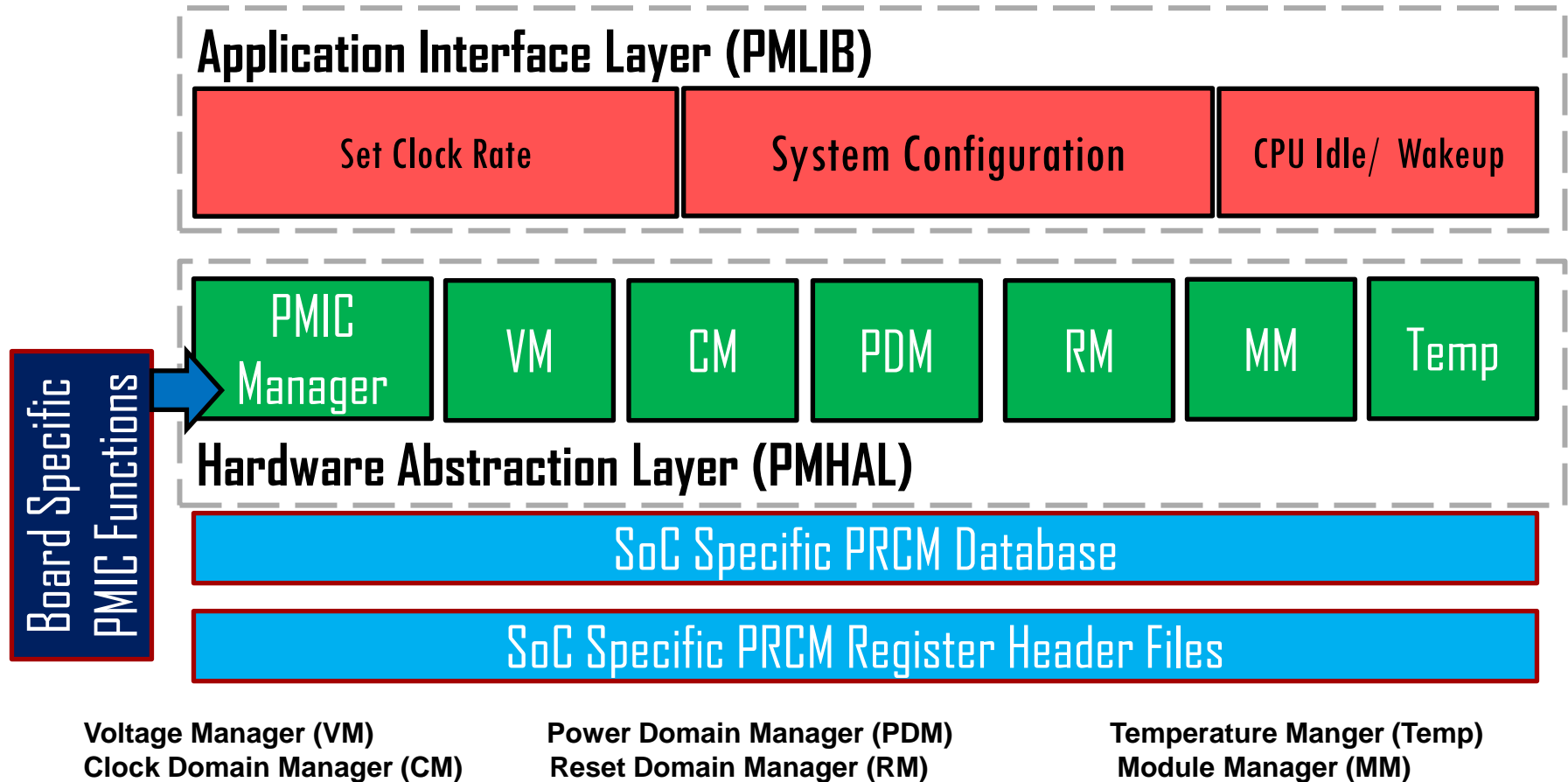
PMHAL: `starterware \include\pm\pmhal\pmhal_bgap.h`



TEXAS INSTRUMENTS

# PM Software Stack

# Power Management Software Stack



# References

- ADAS PM Application Note:  
<https://cdds.ext.ti.com/ematrix/common/emxNavigator.jsp?objectId=28670.42872.8315.8463>
- PRCM Hardware Details: TDA2xx/TDA2ex/TDA3xx TRM
- VisionSDK\_DevelopmentGuide.pdf Section 7 for PM Vision SDK integration details.
- For any further questions please contact your TI representative.

# Thank you