pmm.c (Ver 2.0 12/15/2010)	HAL_pmm.c (Ver3.80.14.01)

`uint16_t PMM_setVCoreUp ( uint8_t level){`
 	
`uint32_t PMMRIE_backup, SVSMHCTL_backup, SVSMLCTL_backup;`
 	
`//The code flow for increasing the Vcore has been altered to work around`
`//the erratum FLASH37.`
`//Please refer to the Errata sheet to know if a specific device is affected`
`//DO NOT ALTER THIS FUNCTION`
	`static uint16_t SetVCoreUp(uint8_t level)`
	`{`
	`uint16_t PMMRIE_backup, SVSMHCTL_backup, SVSMLCTL_backup;`
	
	`// The code flow for increasing the Vcore has been altered to work around`
	`// the erratum FLASH37.`
	`// Please refer to the Errata sheet to know if a specific device is affected`
	`// DO NOT ALTER THIS FUNCTION`

`//Open PMM registers for write access`
`HWREG8(PMM_BASE + OFS_PMMCTL0_H) = 0xA5;`
 	
`//Disable dedicated Interrupts`
`//Backup all registers`
`PMMRIE_backup = HWREG16(PMM_BASE + OFS_PMMRIE);`
`HWREG16(PMM_BASE + OFS_PMMRIE) &= ~(SVMHVLRPE | SVSHPE | SVMLVLRPE |`
`SVSLPE | SVMHVLRIE | SVMHIE |`
`SVSMHDLYIE | SVMLVLRIE | SVMLIE |`
`SVSMLDLYIE`
`);`
`SVSMHCTL_backup = HWREG16(PMM_BASE + OFS_SVSMHCTL);`
`SVSMLCTL_backup = HWREG16(PMM_BASE + OFS_SVSMLCTL);`
	`// Open PMM registers for write access`
	`PMMCTL0_H = 0xA5;`
	
	`// Disable dedicated Interrupts`
	`// Backup all registers`
	`PMMRIE_backup = PMMRIE;`
	`PMMRIE &= ~(SVMHVLRPE | SVSHPE | SVMLVLRPE | SVSLPE | SVMHVLRIE |`
	`SVMHIE | SVSMHDLYIE | SVMLVLRIE | SVMLIE | SVSMLDLYIE );`
	
	`SVSMHCTL_backup = SVSMHCTL;`
	`SVSMLCTL_backup = SVSMLCTL;`

`//Clear flags`
`HWREG16(PMM_BASE + OFS_PMMIFG) = 0;`
 	
`//Set SVM highside to new level and check if a VCore increase is possible`
`HWREG16(PMM_BASE + OFS_SVSMHCTL) = SVMHE | SVSHE | (SVSMHRRL0 * level);`
 	
`//Wait until SVM highside is settled`
`while ((HWREG16(PMM_BASE + OFS_PMMIFG) & SVSMHDLYIFG) == 0) ;`
 	
`//Clear flag`
`HWREG16(PMM_BASE + OFS_PMMIFG) &= ~SVSMHDLYIFG;`
	`// Clear flags`
	`PMMIFG = 0;`
	
	`// Set SVM highside to new level and check if a VCore increase is possible`
	`SVSMHCTL = SVMHE | SVSHE | (SVSMHRRL0 * level);`
	
	`// Wait until SVM highside is settled`
	`while ((PMMIFG & SVSMHDLYIFG) == 0);`
	
	`// Clear flag`
	`PMMIFG &= ~SVSMHDLYIFG;`

`//Check if a VCore increase is possible`
`if ((HWREG16(PMM_BASE + OFS_PMMIFG) & SVMHIFG) == SVMHIFG){`
`//-> Vcc is too low for a Vcore increase`
`//recover the previous settings`
`HWREG16(PMM_BASE + OFS_PMMIFG) &= ~SVSMHDLYIFG;`
`HWREG16(PMM_BASE + OFS_SVSMHCTL) = SVSMHCTL_backup;`
 	
`//Wait until SVM highside is settled`
`while ((HWREG16(PMM_BASE + OFS_PMMIFG) & SVSMHDLYIFG) == 0) ;`
 	
`//Clear all Flags`
`HWREG16(PMM_BASE +`
`OFS_PMMIFG) &= ~(SVMHVLRIFG | SVMHIFG | SVSMHDLYIFG |`
`SVMLVLRIFG | SVMLIFG |`
`SVSMLDLYIFG`
`);`
`//Restore PMM interrupt enable register`
`HWREG16(PMM_BASE + OFS_PMMRIE) = PMMRIE_backup;`
`//Lock PMM registers for write access`
	`// Check if a VCore increase is possible`
	`if ((PMMIFG & SVMHIFG) == SVMHIFG) {  // -> Vcc is too low for a Vcore increase`
	`// recover the previous settings`
	
	`PMMIFG &= ~SVSMHDLYIFG;`
	`SVSMHCTL = SVSMHCTL_backup;`
	
	`// Wait until SVM highside is settled`
	`while ((PMMIFG & SVSMHDLYIFG) == 0);`
	
	`// Clear all Flags`
	`PMMIFG &= ~(SVMHVLRIFG | SVMHIFG | SVSMHDLYIFG | SVMLVLRIFG | SVMLIFG | SVSMLDLYIFG);`
	
	`PMMRIE = PMMRIE_backup;  // Restore PMM interrupt enable register`

pmm.c (Ver 2.0 12/15/2010)	HAL_pmm.c (Ver3.80.14.01)

HWREG8(PMM_BASE + OFS_PMMCTL0_H) = 0x00;
//return: voltage not set
return (STATUS_FAIL) ;
}
	PMMCTL0_H = 0x00; // Lock PMM registers for write access
	
	return PMM_STATUS_ERROR; // return: voltage not set
	}

//Set also SVS highside to new level
//Vcc is high enough for a Vcore increase
HWREG16(PMM_BASE + OFS_SVSMHCTL) |= (SVSHRVL0 * level);
 	
//Wait until SVM highside is settled
while ((HWREG16(PMM_BASE + OFS_PMMIFG) & SVSMHDLYIFG) == 0) ;
 	
//Clear flag
HWREG16(PMM_BASE + OFS_PMMIFG) &= ~SVSMHDLYIFG;
 	
//Set VCore to new level
HWREG8(PMM_BASE + OFS_PMMCTL0_L) = PMMCOREV0 * level;
 	
//Set SVM, SVS low side to new level
HWREG16(PMM_BASE + OFS_SVSMLCTL) = SVMLE | (SVSMLRRL0 * level) |
SVSLE | (SVSLRVL0 * level);
	// Set also SVS highside to new level
	// Vcc is high enough for a Vcore increase
	SVSMHCTL |= (SVSHRVL0 * level);
	
	// Wait until SVM highside is settled
	while ((PMMIFG & SVSMHDLYIFG) == 0);
	
	// Clear flag
	PMMIFG &= ~SVSMHDLYIFG;
	
	// Set VCore to new level
	PMMCTL0_L = PMMCOREV0 * level;
	
	// Set SVM, SVS low side to new level
	SVSMLCTL = SVMLE | (SVSMLRRL0 * level) | SVSLE | (SVSLRVL0 * level);

//Wait until SVM, SVS low side is settled
while ((HWREG16(PMM_BASE + OFS_PMMIFG) & SVSMLDLYIFG) == 0) ;
 	
//Clear flag
HWREG16(PMM_BASE + OFS_PMMIFG) &= ~SVSMLDLYIFG;
//SVS, SVM core and high side are now set to protect for the new core level
 	
//Restore Low side settings
//Clear all other bits _except_ level settings
HWREG16(PMM_BASE + OFS_SVSMLCTL) &= (SVSLRVL0 + SVSLRVL1 + SVSMLRRL0 +
SVSMLRRL1 + SVSMLRRL2
);
	// Wait until SVM, SVS low side is settled
	while ((PMMIFG & SVSMLDLYIFG) == 0);
	
	// Clear flag
	PMMIFG &= ~SVSMLDLYIFG;
	// SVS, SVM core and high side are now set to protect for the new core level
	
	// Restore Low side settings
	// Clear all other bits _except_ level settings
	SVSMLCTL &= (SVSLRVL0+SVSLRVL1+SVSMLRRL0+SVSMLRRL1+SVSMLRRL2);

//Clear level settings in the backup register,keep all other bits
SVSMLCTL_backup &=
~(SVSLRVL0 + SVSLRVL1 + SVSMLRRL0 + SVSMLRRL1 + SVSMLRRL2);
 	
//Restore low-side SVS monitor settings
HWREG16(PMM_BASE + OFS_SVSMLCTL) |= SVSMLCTL_backup;
 	
//Restore High side settings
//Clear all other bits except level settings
HWREG16(PMM_BASE + OFS_SVSMHCTL) &= (SVSHRVL0 + SVSHRVL1 +
SVSMHRRL0 + SVSMHRRL1 +
SVSMHRRL2
);
 	
//Clear level settings in the backup register,keep all other bits
SVSMHCTL_backup &=
~(SVSHRVL0 + SVSHRVL1 + SVSMHRRL0 + SVSMHRRL1 + SVSMHRRL2);
 	
//Restore backup
HWREG16(PMM_BASE + OFS_SVSMHCTL) |= SVSMHCTL_backup;
	// Clear level settings in the backup register,keep all other bits
	SVSMLCTL_backup &= ~(SVSLRVL0+SVSLRVL1+SVSMLRRL0+SVSMLRRL1+SVSMLRRL2);
	
	// Restore low-side SVS monitor settings
	SVSMLCTL |= SVSMLCTL_backup;
	
	// Restore High side settings
	// Clear all other bits except level settings
	SVSMHCTL &= (SVSHRVL0+SVSHRVL1+SVSMHRRL0+SVSMHRRL1+SVSMHRRL2);
	
	// Clear level settings in the backup register,keep all other bits
	SVSMHCTL_backup &= ~(SVSHRVL0+SVSHRVL1+SVSMHRRL0+SVSMHRRL1+SVSMHRRL2);
	
	// Restore backup
	SVSMHCTL |= SVSMHCTL_backup;

pmm.c (Ver 2.0 12/15/2010)	HAL_pmm.c (Ver3.80.14.01)

`//Wait until high side, low side settled`
`while (((HWREG16(PMM_BASE + OFS_PMMIFG) & SVSMLDLYIFG) == 0) ||`
`((HWREG16(PMM_BASE + OFS_PMMIFG) & SVSMHDLYIFG) == 0)) ;`
 	
`//Clear all Flags`
`HWREG16(PMM_BASE + OFS_PMMIFG) &= ~(SVMHVLRIFG | SVMHIFG | SVSMHDLYIFG |`
`SVMLVLRIFG | SVMLIFG | SVSMLDLYIFG`
`);`
 	
`//Restore PMM interrupt enable register`
`HWREG16(PMM_BASE + OFS_PMMRIE) = PMMRIE_backup;`
 	
`//Lock PMM registers for write access`
`HWREG8(PMM_BASE + OFS_PMMCTL0_H) = 0x00;`
 	
`return ( STATUS_SUCCESS) ;`
`}`
	`// Wait until high side, low side settled`
	`while (((PMMIFG & SVSMLDLYIFG) == 0) && ((PMMIFG & SVSMHDLYIFG) == 0));`
	
	`// Clear all Flags`
	`PMMIFG &= ~(SVMHVLRIFG | SVMHIFG | SVSMHDLYIFG | SVMLVLRIFG | SVMLIFG | SVSMLDLYIFG);`
	
	`PMMRIE = PMMRIE_backup;  // Restore PMM interrupt enable register`
	
	`PMMCTL0_H = 0x00;  // Lock PMM registers for write access`
	
	`return PMM_STATUS_OK;`
	`}`

`uint16_t PMM_setVCoreDown ( uint8_t level){`
`uint32_t PMMRIE_backup, SVSMHCTL_backup, SVSMLCTL_backup;`
 	
`//The code flow for decreasing the Vcore has been altered to work around`
`//the erratum FLASH37.`
`//Please refer to the Errata sheet to know if a specific device is affected`
`//DO NOT ALTER THIS FUNCTION`
 	
`//Open PMM registers for write access`
`HWREG8(PMM_BASE + OFS_PMMCTL0_H) = 0xA5;`
 	
`//Disable dedicated Interrupts`
`//Backup all registers`
`PMMRIE_backup = HWREG16(PMM_BASE + OFS_PMMRIE);`
`HWREG16(PMM_BASE + OFS_PMMRIE) &= ~(SVMHVLRPE | SVSHPE | SVMLVLRPE |`
`SVSLPE | SVMHVLRIE | SVMHIE |`
`SVSMHDLYIE | SVMLVLRIE | SVMLIE |`
`SVSMLDLYIE`
`);`
`SVSMHCTL_backup = HWREG16(PMM_BASE + OFS_SVSMHCTL);`
`SVSMLCTL_backup = HWREG16(PMM_BASE + OFS_SVSMLCTL);`
	`static uint16_t SetVCoreDown(uint8_t level)`
	`{`
	`uint16_t PMMRIE_backup, SVSMHCTL_backup, SVSMLCTL_backup;`
	
	`// The code flow for decreasing the Vcore has been altered to work around`
	`// the erratum FLASH37.`
	`// Please refer to the Errata sheet to know if a specific device is affected`
	`// DO NOT ALTER THIS FUNCTION`
	
	`// Open PMM registers for write access`
	`PMMCTL0_H = 0xA5;`
	
	`// Disable dedicated Interrupts`
	`// Backup all registers`
	`PMMRIE_backup = PMMRIE;`
	`PMMRIE &= ~(SVMHVLRPE | SVSHPE | SVMLVLRPE | SVSLPE | SVMHVLRIE |`
	`SVMHIE | SVSMHDLYIE | SVMLVLRIE | SVMLIE | SVSMLDLYIE );`
	
	`SVSMHCTL_backup = SVSMHCTL;`
	`SVSMLCTL_backup = SVSMLCTL;`

`//Clear flags`
`HWREG16(PMM_BASE + OFS_PMMIFG) &= ~(SVMHIFG | SVSMHDLYIFG |`
`SVMLIFG | SVSMLDLYIFG`
`);`
 	
`//Set SVM, SVS high & low side to new settings in normal mode`
`HWREG16(PMM_BASE + OFS_SVSMHCTL) = SVMHE | (SVSMHRRL0 * level) |`
`SVSHE | (SVSHRVL0 * level);`
`HWREG16(PMM_BASE + OFS_SVSMLCTL) = SVMLE | (SVSMLRRL0 * level) |`
`SVSLE | (SVSLRVL0 * level);`
	`// Clear flags`
	`PMMIFG &= ~(SVMHIFG | SVSMHDLYIFG | SVMLIFG | SVSMLDLYIFG);`
	
	`// Set SVM, SVS high & low side to new settings in normal mode`
	`SVSMHCTL = SVMHE | (SVSMHRRL0 * level) | SVSHE | (SVSHRVL0 * level);`
	`SVSMLCTL = SVMLE | (SVSMLRRL0 * level) | SVSLE | (SVSLRVL0 * level);`

pmm.c (Ver 2.0 12/15/2010)	HAL_pmm.c (Ver3.80.14.01)

`//Wait until SVM high side and SVM low side is settled`
`while ((HWREG16(PMM_BASE + OFS_PMMIFG) & SVSMHDLYIFG) == 0 ‖`
`(HWREG16(PMM_BASE + OFS_PMMIFG) & SVSMLDLYIFG) == 0) ;`
 	
`//Clear flags`
`HWREG16(PMM_BASE + OFS_PMMIFG) &= ~(SVSMHDLYIFG + SVSMLDLYIFG);`
`//SVS, SVM core and high side are now set to protect for the new core level`
	`// Wait until SVM high side and SVM low side is settled`
	`while ((PMMIFG & SVSMHDLYIFG) == 0 ‖ (PMMIFG & SVSMLDLYIFG) == 0);`
	
	`// Clear flags`
	`PMMIFG &= ~(SVSMHDLYIFG + SVSMLDLYIFG);`
	`// SVS, SVM core and high side are now set to protect for the new core level`

`//Set VCore to new level`
`HWREG8(PMM_BASE + OFS_PMMCTL0_L) = PMMCOREV0 * level;`
 	
`//Restore Low side settings`
`//Clear all other bits _except_ level settings`
`HWREG16(PMM_BASE + OFS_SVSMLCTL) &= (SVSLRVL0 + SVSLRVL1 + SVSMLRRL0 +`
`SVSMLRRL1 + SVSMLRRL2`
`);`
 	
`//Clear level settings in the backup register,keep all other bits`
`SVSMLCTL_backup &=`
`~(SVSLRVL0 + SVSLRVL1 + SVSMLRRL0 + SVSMLRRL1 + SVSMLRRL2);`
 	
`//Restore low-side SVS monitor settings`
`HWREG16(PMM_BASE + OFS_SVSMLCTL) ‖= SVSMLCTL_backup;`
 	
`//Restore High side settings`
`//Clear all other bits except level settings`
`HWREG16(PMM_BASE + OFS_SVSMHCTL) &= (SVSHRVL0 + SVSHRVL1 + SVSMHRRL0 +`
`SVSMHRRL1 + SVSMHRRL2`
`);`
 	
`//Clear level settings in the backup register, keep all other bits`
`SVSMHCTL_backup &=`
`~(SVSHRVL0 + SVSHRVL1 + SVSMHRRL0 + SVSMHRRL1 + SVSMHRRL2);`
	`// Set VCore to new level`
	`PMMCTL0_L = PMMCOREV0 * level;`
	
	`// Restore Low side settings`
	`// Clear all other bits _except_ level settings`
	`SVSMLCTL &= (SVSLRVL0+SVSLRVL1+SVSMLRRL0+SVSMLRRL1+SVSMLRRL2);`
	
	`// Clear level settings in the backup register,keep all other bits`
	`SVSMLCTL_backup &= ~(SVSLRVL0+SVSLRVL1+SVSMLRRL0+SVSMLRRL1+SVSMLRRL2);`
	
	`// Restore low-side SVS monitor settings`
	`SVSMLCTL ‖= SVSMLCTL_backup;`
	
	`// Restore High side settings`
	`// Clear all other bits except level settings`
	`SVSMHCTL &= (SVSHRVL0+SVSHRVL1+SVSMHRRL0+SVSMHRRL1+SVSMHRRL2);`
	
	`// Clear level settings in the backup register, keep all other bits`
	`SVSMHCTL_backup &= ~(SVSHRVL0+SVSHRVL1+SVSMHRRL0+SVSMHRRL1+SVSMHRRL2);`

`//Restore backup`
`HWREG16(PMM_BASE + OFS_SVSMHCTL) ‖= SVSMHCTL_backup;`
 	
`//Wait until high side, low side settled`
`while (((HWREG16(PMM_BASE + OFS_PMMIFG) & SVSMLDLYIFG) == 0) ‖`
`((HWREG16(PMM_BASE + OFS_PMMIFG) & SVSMHDLYIFG) == 0)) ;`
 	
`//Clear all Flags`
`HWREG16(PMM_BASE + OFS_PMMIFG) &= ~(SVMHVLRIFG ‖ SVMHIFG ‖ SVSMHDLYIFG ‖`
`SVMLVLRIFG ‖ SVMLIFG ‖ SVSMLDLYIFG`
`);`
 	
`//Restore PMM interrupt enable register`
`HWREG16(PMM_BASE + OFS_PMMRIE) = PMMRIE_backup;`
`//Lock PMM registers for write access`
`HWREG8(PMM_BASE + OFS_PMMCTL0_H) = 0x00;`
`//Return: OK`
`return ( STATUS_SUCCESS) ;`
`}`
	`// Restore backup`
	`SVSMHCTL ‖= SVSMHCTL_backup;`
	
	`// Wait until high side, low side settled`
	`while (((PMMIFG & SVSMLDLYIFG) == 0) && ((PMMIFG & SVSMHDLYIFG) == 0));`
	
	`// Clear all Flags`
	`PMMIFG &= ~(SVMHVLRIFG ‖ SVMHIFG ‖ SVSMHDLYIFG ‖ SVMLVLRIFG ‖ SVMLIFG ‖ SVSMLDLYIFG);`
	
	`PMMRIE = PMMRIE_backup;  // Restore PMM interrupt enable register`
	
	`PMMCTL0_H = 0x00;  // Lock PMM registers for write access`
	
	`return PMM_STATUS_OK;  // Return: OK`
	`}`

pmm.c (Ver 2.0 12/15/2010)

```c
bool PMM_setVCore ( uint8_t level){
    uint8_t actlevel;
    bool status = STATUS_SUCCESS;
    uint16_t interruptState;

    //Set Mask for Max. level
    level &= PMMCOREV_3;

    //Get actual VCore
    actlevel = (HWREG16(PMM_BASE + OFS_PMMCTL0) & PMMCOREV_3);

    //Disable interrupts because certain peripherals will not
    //work during VCORE change
    interruptState = __get_interrupt_state();
    __disable_interrupt();
    __no_operation();

    //step by step increase or decrease
    while ((level != actlevel) && (status == STATUS_SUCCESS))
    {
        if (level > actlevel){
            status = PMM_setVCoreUp(++actlevel);
        } else   {
            status = PMM_setVCoreDown(--actlevel);
        }
    }

    //Re-enable interrupt state to whatever it was before
    if(interruptState & GIE)
    {
        __enable_interrupt();
    }

    return ( status) ;
}

uint16_t PMM_getInterruptStatus (uint16_t mask)
{
    return ( (HWREG16(PMM_BASE + OFS_PMMIFG)) & mask );
}
```

HAL_pmm.c (Ver3.80.14.01)

```c
uint16_t SetVCore(uint8_t level)
{
  uint16_t actlevel;
  uint16_t status = 0;

  level &= PMMCOREV_3;                        // Set Mask for Max. level



  actlevel = (PMMCTL0 & PMMCOREV_3);          // Get actual VCore









                                              // step by step increase or decrease
  while (((level != actlevel) && (status == 0)) || (level < actlevel)) {

    if (level > actlevel) {
      status = SetVCoreUp(++actlevel);
    }
    else {
      status = SetVCoreDown(--actlevel);
    }
  }




  return status;
}
```