→related EMIF-Tool output GEL configuration

```
#include "board_cfg.h"
#include "board_internal.h"

#include "sbl_utils_ddr_config.h"
#include <ti/csl/csl_emif4fAux.h>
#include <ti/csl/cslr_dmm.h>
#include <ti/csl/cslr_ma_mpu_lsm.h>
#include <ti/csl/src/ip/emif4/V2/csl_emif4d5.h>

#define DDR_PHY_CTRL1_VALUE(emif_phy_read_latency, emif_phy_fast_dll_lock,      ¥
   emif_phy_dll_lock_diff, emif_phy_invert_clkout, emif_phy_dis_calib_rst,    ¥
   emif_phy_half_delay_mode, emif_phy_levelling_disabled)                     ¥
   ((emif_phy_read_latency << 0U) | (emif_phy_fast_dll_lock << 9U) |          ¥
   (emif_phy_dll_lock_diff << 10U) | (emif_phy_invert_clkout << 18U) |        ¥
   (emif_phy_dis_calib_rst << 19U) | (emif_phy_half_delay_mode << 21U) |      ¥
   (emif_phy_levelling_disabled))


#define MPU_DEVICE_PRM_REGS                     (0x4ae07d00U)
#define PRM_RSTST_REG                           (0x4U)

/**< IODFT TLGC */
uint32_t ioDftLogicCtrl;

/**< Read Write level ramp window*/
uint32_t readWriteLvlRampWin;

static void ddr_delay(uint32_t ix);

static void emif_ddr3_updateHwLevelOutput(CSL_emifHandle hEmif);

static void ddr_delay(uint32_t ix)
{
   while (ix--) {
      asm("   NOP");
   }
}

int emifConfigureDdr3
(
   CSL_emifHandle hEmif,
   CSL_emifDdrConfig *ddr3Config,
   Uint32 enableHwLeveling
);

/* Set the desired DDR3 configuration -- assumes 66.67 MHz DDR3 clock input */
Board_STATUS Board_DDR3Init()
{
   int retVal = BOARD_SOK;
   Board_IDInfo id;
   Board_STATUS ret;
   ret = Board_getIDInfo(&id);
   if (ret != BOARD_SOK)
   {
      return ret;
   }

   /* Check if version is 1.0 or 1.1 */
   /* Check if DRA chip (AM570x) */
   if ((id.boardName[0] == 'D') &&
      (id.boardName[1] == 'R') &&
      (id.boardName[2] == 'A'))
   {
      SBLUtilsDDR3Config(0);
      return ret;
   }


   CSL_emifObj emifObj1;
   CSL_emifHandle hEmif1 = &emifObj1;
   CSL_emifDdrConfig ddr3Config1;
   CSL_ckgen_cm_core_aonRegs *hCkgenCmCoreAon =
      (CSL_ckgen_cm_core_aonRegs *) CSL_MPU_CKGEN_CM_CORE_AON_REGS;
   CSL_control_core_padRegs *hCtrlCorePad =
```

```
    (CSL_control_core_padRegs *) CSL_MPU_CTRL_MODULE_CORE_CORE_PAD_REGISTERS_REGS;
CSL_control_core_wkupRegs *hCtrlCoreWkup =
    (CSL_control_core_wkupRegs *) CSL_MPU_CTRL_MODULE_WKUP_CORE_REGISTERS_REGS;
CSL_DmmRegs *hDmmCfg =(CSL_DmmRegs *) CSL_MPU_DMM_CONF_REGS_REGS;
CSL_MampuLsmRegs *hMampuLsm = (CSL_MampuLsmRegs *) CSL_MPU_MA_MPU_LSM_REGS;

hEmif1->regs = (CSL_emifRegsOvly)CSL_MPU_EMIF1_CONF_REGS_REGS;

/* DLL override disable =0 ; enable = 1 */
hCkgenCmCoreAon->CM_DLL_CTRL_REG = 0x00000000;

/*
 * CONTROL_DDR3CH1_0 -- channel_1 CMDs
 * -- 40Ohm Ron (011)
 * -- SR=slowest-3 (111) on CMDs
 * -- CLK SR=slow (011)
 * -- No pulls (00)
 */
hCtrlCorePad->CONTROL_DDRCACH1_0 = 0x60606080U;

 /*
 * CTRL_CORE_CONTROL_DDRCH1_0
 * - Impedance = 40ohm / SlewRate = fastest / No pulls
  */
hCtrlCorePad->CONTROL_DDRCH1_0 = 0x40404040U;

/*
 *  CTRL_CORE_CONTROL_DDRCH1_1
 * - Impedance = 40ohm / SlewRate = fastest / No pulls
 */
 hCtrlCorePad->CONTROL_DDRCH1_1 = 0x40404040U;

 /*
 * CTRL_CORE_CONTROL_DDRCH1_2
 * - Impedance = 40ohm / SlewRate = fastest / No pulls
 */
 hCtrlCorePad->CONTROL_DDRCH1_2 = 0x00404000U;

 /*
 * CTRL_CORE_CONTROL_DDRIO_0
 * - DDRCH1_VREF_DQ0/1_INT_EN = 0, reset values for other fields
 */
 hCtrlCorePad->CONTROL_DDRIO_0 = 0x00094A40U;

/*
 * EMIF1_SDRAM_CONFIG_EXT
 * -- cslice_en[2:0]=111 / Local_odt=01 / dyn_pwrdn=1 / dis_reset=0 / rd_lvl_samples=11 (128)
 */
hCtrlCoreWkup->EMIF1_SDRAM_CONFIG_EXT = 0x0001C1A7U;

 ddr3Config1.emifDdrParam.ddrPhyCtrl = hEmif1->regs->DDR_PHY_CONTROL_2;

 ddr3Config1.emifDdrParam.sdramTim1 = 0xD113781C;                    SDRAM_TIM_1
 ddr3Config1.emifDdrParam.sdramTim2 = 0x30B37FE3;                    SDRAM_TIM_2
 ddr3Config1.emifDdrParam.sdramTim3 = 0x409F8AD8;                    SDRAM_TIM_3

 ddr3Config1.emifDdrParam.sdramCfg = 0x61862B32U;                    SDRAM_CONFIG
 ddr3Config1.emifDdrParam.sdramCfg2 = 0x08000000U;
 ddr3Config1.emifDdrParam.sdramRefCtrl = 0x0000144AU;                SDRAM_REF_CTRL
 ddr3Config1.emifDdrParam.zqConfig = 0x5007190BU;
 ddr3Config1.emifDdrParam.sdramPwrMngtCtrl = 0x00000000U;

 ioDftLogicCtrl = hEmif1->regs->IODFT_TEST_LOGIC_GLOBAL_CONTROL;
readWriteLvlRampWin = hEmif1->regs->READ_WRITE_LEVELING_RAMP_WINDOW;

 ddr3Config1.emifDdrPhyParam.ctrlSlaveRatio = 0x80U;                 EMIF_PHY_CTRL_SLAVE_RATIO
 ddr3Config1.emifDdrPhyParam.dqOffset = 0x40U;                       EMIF_PHY_DQ_OFFSET
 ddr3Config1.emifDdrPhyParam.gateLevelInitMode = 0x01U;
 ddr3Config1.emifDdrPhyParam.fifoWeInDelay = 0x0U;
 ddr3Config1.emifDdrPhyParam.ctrlSlaveDelay = 0x0U;
 ddr3Config1.emifDdrPhyParam.readDqsSlaveDelay = 0x20U;
 ddr3Config1.emifDdrPhyParam.writeDqsSlaveDelay = 0x60U;
 ddr3Config1.emifDdrPhyParam.writeDataSlaveDelay = 0x80U;

 ddr3Config1.emifDdrPhyParam.gateLevelRatio = 0x00U;
 ddr3Config1.emifDdrPhyParam.writeLevelInitRatio = 0x00;
 ddr3Config1.emifDdrPhyParam.writeDqsSlaveRatio = 0x60U;
```

```
    ddr3Config1.emifDdrPhyParam.fifoWeSlaveRatio = 0xBBU;
    ddr3Config1.emifDdrPhyParam.useRank0Delays = 0U;

    ddr3Config1.emifDdrPhyParam.gateLevelNumDq0 = 0xFU;
    ddr3Config1.emifDdrPhyParam.writeLevelNumDq0 =
       hEmif1->regs->EXT_PHY_CONTROL_36;

    retVal = emifConfigureDdr3(hEmif1, &ddr3Config1, 1U);

    if(BOARD_SOK == retVal)
    {
       /* MA_LISA_MAP_i */
       hMampuLsm->MAP_0 = 0x80600100U;
       hMampuLsm->MAP_1 = 0x00000000U;
       hMampuLsm->MAP_2 = 0x00000000U;
       hMampuLsm->MAP_3 = 0x00000000U;

       /* DMM_LISA_MAP_i */
       hDmmCfg->LISA_MAP[0U] = 0x80600100U;
       hDmmCfg->LISA_MAP[1U] = 0x00000000U;
       hDmmCfg->LISA_MAP[2U] = 0x00000000U;
       hDmmCfg->LISA_MAP[3U] = 0x00000000U;
    }
    else
    {
       retVal = BOARD_INIT_DDR_FAIL;
    }

    return retVal;
}

int emifConfigureDdr3
(
    CSL_emifHandle hEmif,
    CSL_emifDdrConfig *ddr3Config,
    Uint32 enableHwLeveling
)
{
    int retVal = 0;
    Uint32 regVal = 0U;
     Uint32 emifPhyLevelDisable = 0U;
    Uint32 sdRamRefCtrlInit = 0x0000514CU;                    SDRAM_REF_CTRL_INIT

     /* Fields in DDR_PHY_CTRL_1 */
     /* Bit[21] - calculated using DataMacro/MDLL clock ratio
    * Set to 1 for 532M, so that PHY DLL runs at 266.
    * Set to 0 for 400M, so that PHY DLL runs at 400M.
    * Ensure PHY DLL lower limit of 266M is not violated.
    */
    uint32_t emifPhyHalfDelayMode = 1U;                       EMIF_PHY_HALF_DELAY_MODE
    uint32_t emifPhyDisCalibRst = 0U;    /* Bit[19]    */
    uint32_t emifPhyInvertClkout = 1U;    /* Bit[18]    */   EMIF_PHY_INVERT_CLKOUT
    uint32_t emifPhyDllLockDiff = 0x10U; /* Bit[17:10] */
    uint32_t emifPhyFastDllLock = 0U;   /* Bit[9]     */
    uint32_t emifPhyReadLatency = 0xDU;  /* Bit[4:0], Typically >= (CL + 4) */   EMIF_PHY_READ_LATENCY

     if (0U != (HW_RD_REG32(MPU_DEVICE_PRM_REGS + PRM_RSTST_REG) &
       (PRM_RSTST_GLOBAL_WARM_SW_RST_MASK | PRM_RSTST_EXTERNAL_WARM_RST_MASK)))
    {
       /* Phy reset is required if you are coming back from a warm reset */
       regVal = hEmif->regs->IODFT_TEST_LOGIC_GLOBAL_CONTROL;
       regVal |= 0x400U;
       hEmif->regs->IODFT_TEST_LOGIC_GLOBAL_CONTROL = regVal;
    }

     if(1U == emifPhyInvertClkout)
    {
       regVal = EXT_PHY_CTRL_VALUE((ddr3Config->emifDdrPhyParam.ctrlSlaveRatio + 0x80U));
       hEmif->regs->EXT_PHY_CONTROL_1 = regVal;
       hEmif->regs->EXT_PHY_CONTROL_1_SHADOW = regVal;
    }

     /* PHY settings for DQ offset, DLL override delay, levelling etc. */
    regVal = EXT_PHY_FIFO_WE_SLAVE_CTRL_DELAY(ddr3Config->emifDdrPhyParam.fifoWeInDelay,
          ddr3Config->emifDdrPhyParam.ctrlSlaveDelay);
    hEmif->regs->EXT_PHY_CONTROL_22 = regVal;
    hEmif->regs->EXT_PHY_CONTROL_22_SHADOW = regVal;
```

```
regVal = EXT_PHY_WR_RD_DQS_SLAVE_DELAY(ddr3Config->emifDdrPhyParam.writeDqsSlaveDelay,
      ddr3Config->emifDdrPhyParam.readDqsSlaveDelay);
hEmif->regs->EXT_PHY_CONTROL_23 = regVal;
hEmif->regs->EXT_PHY_CONTROL_23_SHADOW = regVal;
regVal = EXT_PHY_RANK0_DELAY_VALUE(ddr3Config->emifDdrPhyParam.dqOffset,
      ddr3Config->emifDdrPhyParam.gateLevelInitMode,
      ddr3Config->emifDdrPhyParam.useRank0Delays,
      ddr3Config->emifDdrPhyParam.writeDataSlaveDelay);
hEmif->regs->EXT_PHY_CONTROL_24 = regVal;
hEmif->regs->EXT_PHY_CONTROL_24_SHADOW = regVal;
regVal = EXT_PHY_DQ_VALUE(ddr3Config->emifDdrPhyParam.dqOffset);
hEmif->regs->EXT_PHY_CONTROL_25 = regVal;
hEmif->regs->EXT_PHY_CONTROL_25_SHADOW = regVal;

/* Use Init values if HW leveling is enabled */
/* Gate level Init ratios */
regVal = EXT_PHY_GATE_LVL_INIT_VALUE(ddr3Config->emifDdrPhyParam.gateLevelRatio);
hEmif->regs->EXT_PHY_CONTROL_26 = regVal;
hEmif->regs->EXT_PHY_CONTROL_26_SHADOW = regVal;
hEmif->regs->EXT_PHY_CONTROL_27 = regVal;
hEmif->regs->EXT_PHY_CONTROL_27_SHADOW = regVal;
hEmif->regs->EXT_PHY_CONTROL_28 = regVal;
hEmif->regs->EXT_PHY_CONTROL_28_SHADOW = regVal;
hEmif->regs->EXT_PHY_CONTROL_29 = regVal;
hEmif->regs->EXT_PHY_CONTROL_29_SHADOW = regVal;
hEmif->regs->EXT_PHY_CONTROL_30 = regVal;
hEmif->regs->EXT_PHY_CONTROL_30_SHADOW = regVal;

/* WR DQS Init ratios */
regVal = EXT_PHY_WR_LVL_INIT_VALUE(ddr3Config->emifDdrPhyParam.writeLevelInitRatio);
hEmif->regs->EXT_PHY_CONTROL_31 = regVal;
hEmif->regs->EXT_PHY_CONTROL_31_SHADOW = regVal;
hEmif->regs->EXT_PHY_CONTROL_32 = regVal;
hEmif->regs->EXT_PHY_CONTROL_32_SHADOW = regVal;
hEmif->regs->EXT_PHY_CONTROL_33 = regVal;
hEmif->regs->EXT_PHY_CONTROL_33_SHADOW = regVal;
hEmif->regs->EXT_PHY_CONTROL_34 = regVal;
hEmif->regs->EXT_PHY_CONTROL_34_SHADOW = regVal;
hEmif->regs->EXT_PHY_CONTROL_35 = regVal;
hEmif->regs->EXT_PHY_CONTROL_35_SHADOW = regVal;

 regVal = ddr3Config->emifDdrPhyParam.writeLevelNumDq0;
hEmif->regs->EXT_PHY_CONTROL_36 = regVal;
hEmif->regs->EXT_PHY_CONTROL_36_SHADOW = regVal;

 regVal = hEmif->regs->SDRAM_REFRESH_CONTROL_SHADOW;
regVal = (CSL_EMIF4D5_SDRAM_REFRESH_CONTROL_INITREF_DIS_MASK |
      sdRamRefCtrlInit);
regVal = hEmif->regs->SDRAM_REFRESH_CONTROL;
 regVal = (CSL_EMIF4D5_SDRAM_REFRESH_CONTROL_INITREF_DIS_MASK |
      sdRamRefCtrlInit);
hEmif->regs->SDRAM_REFRESH_CONTROL = regVal;

/* Set up the EMIF registers */
hEmif->regs->SDRAM_TIMING_1 = ddr3Config->emifDdrParam.sdramTim1;
hEmif->regs->SDRAM_TIMING_1_SHADOW = ddr3Config->emifDdrParam.sdramTim1;
hEmif->regs->SDRAM_TIMING_2 = ddr3Config->emifDdrParam.sdramTim2;
hEmif->regs->SDRAM_TIMING_2_SHADOW = ddr3Config->emifDdrParam.sdramTim2;
hEmif->regs->SDRAM_TIMING_3 = ddr3Config->emifDdrParam.sdramTim3;
hEmif->regs->SDRAM_TIMING_3_SHADOW = ddr3Config->emifDdrParam.sdramTim3;

hEmif->regs->POWER_MANAGEMENT_CONTROL = ddr3Config->emifDdrParam.sdramPwrMngtCtrl;
hEmif->regs->POWER_MANAGEMENT_CONTROL_SHADOW = ddr3Config->emifDdrParam.sdramPwrMngtCtrl;

hEmif->regs->OCP_CONFIG = 0x0A500000U;

hEmif->regs->IODFT_TEST_LOGIC_GLOBAL_CONTROL = ioDftLogicCtrl;
hEmif->regs->SDRAM_OUTPUT_IMPEDANCE_CALIBRATION_CONFIG = ddr3Config->emifDdrParam.zqConfig;
hEmif->regs->DLL_CALIB_CTRL = 0x00050000U;
hEmif->regs->DLL_CALIB_CTRL_SHADOW = 0x00050000U;

hEmif->regs->READ_WRITE_LEVELING_RAMP_WINDOW = readWriteLvlRampWin;
hEmif->regs->READ_WRITE_LEVELING_RAMP_CONTROL = 0x80000000;
```

hEmif->regs->READ_WRITE_LEVELING_CONTROL = 0U;	DISABLE_READ_LEVELING DISABLE_READ_GATE_LEVELING DISABLE_WRITE_LEVELING

```
regVal = DDR_PHY_CTRL1_VALUE(emifPhyReadLatency, emifPhyFastDllLock,
        emifPhyDllLockDiff, emifPhyInvertClkout, emifPhyDisCalibRst,
        emifPhyHalfDelayMode, emifPhyLevelDisable);

hEmif->regs->DDR_PHY_CONTROL_1 = regVal;
hEmif->regs->DDR_PHY_CONTROL_1_SHADOW = regVal;

/* Backup of the previous value. */
hEmif->regs->DDR_PHY_CONTROL_2 = ddr3Config->emifDdrParam.ddrPhyCtrl;

hEmif->regs->PRIORITY_TO_CLASS_OF_SERVICE_MAPPING = 0U;
hEmif->regs->CONNECTION_ID_TO_CLASS_OF_SERVICE_1_MAPPING = 0U;
hEmif->regs->CONNECTION_ID_TO_CLASS_OF_SERVICE_2_MAPPING = 0U;
hEmif->regs->READ_WRITE_EXECUTION_THRESHOLD = 0x00000305U;
hEmif->regs->COS_CONFIG = 0x00FFFFFU;

/* SDRAM_REF_CTRL_INIT:
 * For DDR3:   value used initially to get 500us delay between
 *           RESET de-assertion to CKE assertion after power-up
 */
hEmif->regs->SDRAM_REFRESH_CONTROL_SHADOW = 0x0000514CU;
hEmif->regs->SDRAM_REFRESH_CONTROL = 0x0000514CU;
hEmif->regs->SDRAM_CONFIG_2 = ddr3Config->emifDdrParam.sdramCfg2;
hEmif->regs->SDRAM_CONFIG = ddr3Config->emifDdrParam.sdramCfg;

ddr_delay(100000);

/* Now update with the correct refresh time */
hEmif->regs->SDRAM_REFRESH_CONTROL_SHADOW = ddr3Config->emifDdrParam.sdramRefCtrl;
hEmif->regs->SDRAM_REFRESH_CONTROL = ddr3Config->emifDdrParam.sdramRefCtrl;

 /* If ECC is enabled. */
 hEmif->regs->ECC_ADDRESS_RANGE_1 = 0U;
 hEmif->regs->ECC_ADDRESS_RANGE_2 = 0U;
 hEmif->regs->ECC_CTRL_REG = (CSL_EMIF4D5_ECC_CTRL_REG_REG_ECC_EN_MASK |
             CSL_EMIF4D5_ECC_CTRL_REG_REG_ECC_ADDR_RGN_PROT_MASK);

/* Launch Full HW levelling. */
regVal = hEmif->regs->EXT_PHY_CONTROL_36;
regVal = (regVal | 0x00000100U);
hEmif->regs->EXT_PHY_CONTROL_36 = regVal;
regVal = hEmif->regs->EXT_PHY_CONTROL_36_SHADOW;
regVal = (regVal | 0x00000100U);
hEmif->regs->EXT_PHY_CONTROL_36_SHADOW = regVal;

/* Disable SDRAM refreshes before levelling */
regVal = hEmif->regs->SDRAM_REFRESH_CONTROL;
regVal = regVal | CSL_EMIF4D5_SDRAM_REFRESH_CONTROL_INITREF_DIS_MASK;
hEmif->regs->SDRAM_REFRESH_CONTROL = regVal;

/* RDWR_LVL_CTRL */
hEmif->regs->READ_WRITE_LEVELING_CONTROL =
   CSL_EMIF4D5_READ_WRITE_LEVELING_CONTROL_RDWRLVLFULL_START_MASK;

/* Some clock cycle delay for refresh to complete. */
ddr_delay(30000U);

/* Wait for the levelling procedure to complete */
while((hEmif->regs->READ_WRITE_LEVELING_CONTROL & 0x80000000) != 0x0U);

/* Enable SDRAM refreshes after levelling */
regVal = hEmif->regs->SDRAM_REFRESH_CONTROL;
regVal = (regVal & ~CSL_EMIF4D5_SDRAM_REFRESH_CONTROL_INITREF_DIS_MASK);
hEmif->regs->SDRAM_REFRESH_CONTROL = regVal;

if((hEmif->regs->STATUS & 0x70) != 0U)
{
   /* Indicates Hardware levelling timeout. */
   retVal = -1;
}
else
{
   emif_ddr3_updateHwLevelOutput(hEmif);

   hEmif->regs->ECC_CTRL_REG = 0U;
}
```

```
    return retVal;
}

static void emif_ddr3_updateHwLevelOutput(CSL_emifHandle hEmif)
{
    /* Following function is needed for whenever CORE can go in and out of
    * INACTIVE/CSWR.
    */
    Uint32 regVal = 0U;

    /*
    ** Updating slave ratios in PHY_STATUSx registers as per HW levelling output
    */
    /* if DISABLE_READ_GATE_LEVELING is set to 0 */
    hEmif->regs->EXT_PHY_CONTROL_2 = hEmif->regs->PHY_STATUS_12;
    hEmif->regs->EXT_PHY_CONTROL_2_SHADOW = hEmif->regs->PHY_STATUS_12;
    hEmif->regs->EXT_PHY_CONTROL_3 = hEmif->regs->PHY_STATUS_13;
    hEmif->regs->EXT_PHY_CONTROL_3_SHADOW = hEmif->regs->PHY_STATUS_13;
    hEmif->regs->EXT_PHY_CONTROL_4 = hEmif->regs->PHY_STATUS_14;
    hEmif->regs->EXT_PHY_CONTROL_4_SHADOW = hEmif->regs->PHY_STATUS_14;
    hEmif->regs->EXT_PHY_CONTROL_5 = hEmif->regs->PHY_STATUS_15;
    hEmif->regs->EXT_PHY_CONTROL_5_SHADOW = hEmif->regs->PHY_STATUS_15;
    hEmif->regs->EXT_PHY_CONTROL_6 = hEmif->regs->PHY_STATUS_16;
    hEmif->regs->EXT_PHY_CONTROL_6_SHADOW = hEmif->regs->PHY_STATUS_16;

    /* if DISABLE_READ_LEVELING is set to 0 */
    hEmif->regs->EXT_PHY_CONTROL_7 = hEmif->regs->PHY_STATUS_7;
    hEmif->regs->EXT_PHY_CONTROL_7_SHADOW = hEmif->regs->PHY_STATUS_7;
    hEmif->regs->EXT_PHY_CONTROL_8 = hEmif->regs->PHY_STATUS_8;
    hEmif->regs->EXT_PHY_CONTROL_8_SHADOW = hEmif->regs->PHY_STATUS_8;
    hEmif->regs->EXT_PHY_CONTROL_9 = hEmif->regs->PHY_STATUS_9;
    hEmif->regs->EXT_PHY_CONTROL_9_SHADOW = hEmif->regs->PHY_STATUS_9;
    hEmif->regs->EXT_PHY_CONTROL_10 = hEmif->regs->PHY_STATUS_10;
    hEmif->regs->EXT_PHY_CONTROL_10_SHADOW = hEmif->regs->PHY_STATUS_10;
    hEmif->regs->EXT_PHY_CONTROL_11 = hEmif->regs->PHY_STATUS_11;
    hEmif->regs->EXT_PHY_CONTROL_11_SHADOW = hEmif->regs->PHY_STATUS_11;

    /* if DISABLE_WRITE_LEVELING is set to 0 */
    hEmif->regs->EXT_PHY_CONTROL_12 = hEmif->regs->PHY_STATUS_17;
    hEmif->regs->EXT_PHY_CONTROL_12_SHADOW = hEmif->regs->PHY_STATUS_17;
    hEmif->regs->EXT_PHY_CONTROL_13 = hEmif->regs->PHY_STATUS_18;
    hEmif->regs->EXT_PHY_CONTROL_13_SHADOW = hEmif->regs->PHY_STATUS_18;
    hEmif->regs->EXT_PHY_CONTROL_14 = hEmif->regs->PHY_STATUS_19;
    hEmif->regs->EXT_PHY_CONTROL_14_SHADOW = hEmif->regs->PHY_STATUS_19;
    hEmif->regs->EXT_PHY_CONTROL_15 = hEmif->regs->PHY_STATUS_20;
    hEmif->regs->EXT_PHY_CONTROL_15_SHADOW = hEmif->regs->PHY_STATUS_20;
    hEmif->regs->EXT_PHY_CONTROL_16 = hEmif->regs->PHY_STATUS_21;
    hEmif->regs->EXT_PHY_CONTROL_16_SHADOW = hEmif->regs->PHY_STATUS_21;

    /* EMIF_PHY_WR_DQS_SLAVE_RATIO */
    hEmif->regs->EXT_PHY_CONTROL_17 = hEmif->regs->PHY_STATUS_22;
    hEmif->regs->EXT_PHY_CONTROL_17_SHADOW = hEmif->regs->PHY_STATUS_22;
    hEmif->regs->EXT_PHY_CONTROL_18 = hEmif->regs->PHY_STATUS_23;
    hEmif->regs->EXT_PHY_CONTROL_18_SHADOW = hEmif->regs->PHY_STATUS_23;
    hEmif->regs->EXT_PHY_CONTROL_19 = hEmif->regs->PHY_STATUS_24;
    hEmif->regs->EXT_PHY_CONTROL_19_SHADOW = hEmif->regs->PHY_STATUS_24;
    hEmif->regs->EXT_PHY_CONTROL_20 = hEmif->regs->PHY_STATUS_25;
    hEmif->regs->EXT_PHY_CONTROL_20_SHADOW = hEmif->regs->PHY_STATUS_25;
    hEmif->regs->EXT_PHY_CONTROL_21 = hEmif->regs->PHY_STATUS_26;
    hEmif->regs->EXT_PHY_CONTROL_21_SHADOW = hEmif->regs->PHY_STATUS_26;

    regVal = hEmif->regs->DDR_PHY_CONTROL_1;
    regVal = (regVal | 0x0E000000U);
    hEmif->regs->DDR_PHY_CONTROL_1 = regVal;

    regVal = hEmif->regs->DDR_PHY_CONTROL_1_SHADOW;
    regVal = (regVal | 0x0E000000U);
    hEmif->regs->DDR_PHY_CONTROL_1_SHADOW = regVal;

    hEmif->regs->READ_WRITE_LEVELING_RAMP_CONTROL = 0U;
}
```