DDR Controller Configuration on LS2085/LS2080 Bringing up

This document introduces DDR controller configuration to support new type DDR4 SDRAMs during LS2085/LS2080 target board bringing up. These sections are described in the following, DDR controller memory mapped registers related with new SDRAM setting up on LS2085/LS2080 processors; using “DDR Memory Controller Configuration” tool provided in QCVS tool to calculate the basic DDR controller configuration parameters according to the new DIMM datasheet or SPD provided by the manufacture; use DDR validation(DDRv) tool to validate and optimize the DDR configuration by gradually refining an initial DDR configuration up to an optimal configuration; modify CodeWarrior initialization file and u-boot source code with DDR controller optimized configuration parameter.

1. DDR Controller Memory Mapped registers on LS2085/LS2080

The following section introduces DDR controller memory mapped registers which need to be initialized when the target board boots up.

DDR memory controller 1:
0x01080130: DDR SDRAM clock control (DDR_SDRAM_CLK_CNTL)
0x01080000:Chip select 0 memory bounds (CS0_BNDS)
0x01080080: Chip select a configuration (CS0_CONFIG)
0x010800c0: Chip select a configuration 2 (CS0_CONFIG_2)
0x01080008: Chip select 1 memory bounds (CS1_BNDS)
0x01080084: Chip select a configuration (CS1_CONFIG)
0x010800c4: Chip select a configuration 2(CS1_CONFIG_2)
0x01080010: Chip select 2 memory bounds (CS2_BNDS)
0x01080088: Chip select a configuration (CS2_CONFIG)
0x010800c8: Chip select a configuration 2(CS2_CONFIG_2)
0x01080018: Chip select 3 memory bounds (CS3_BNDS)
0x0108008c: Chip select a configuration (CS3_CONFIG)
0x010800cc: Chip select a configuration 2(CS3_CONFIG_2)
0x01080100: DDR SDRAM timing configuration 3 (TIMING_CFG_3)
0x01080104: DDR SDRAM timing configuration 0 (TIMING_CFG_0)
0x01080108: DDR SDRAM timing configuration 1 (TIMING_CFG_1)
0x0108010c: DDR SDRAM timing configuration 2 (TIMING_CFG_2)
0x01080160: DDR SDRAM timing configuration 4 (TIMING_CFG_4)
0x01080164: DDR SDRAM timing configuration 5 (TIMING_CFG_5)
0x01080168: DDR SDRAM timing configuration 6 (TIMING_CFG_6)
0x0108016c: DDR SDRAM timing configuration 7 (TIMING_CFG_7)
0x01080250: DDR SDRAM timing configuration 8 (TIMING_CFG_8)
0x01080118: DDR SDRAM mode configuration (DDR_SDRAM_MODE)
0x0108011c: DDR SDRAM mode configuration 2 (DDR_SDRAM_MODE_2)
0x01080200: DDR SDRAM mode configuration 3 (DDR_SDRAM_MODE_3)
0x01080204: DDR SDRAM mode configuration 4 (DDR_SDRAM_MODE_4)
0x01080208: DDR SDRAM mode configuration 5 (DDR_SDRAM_MODE_5)
0x0108020c: DDR SDRAM mode configuration 6 (DDR_SDRAM_MODE_6)
0x01080210: DDR SDRAM mode configuration 7 (DDR_SDRAM_MODE_7)
0x01080214: DDR SDRAM mode configuration 8 (DDR_SDRAM_MODE_8)
0x01080220: DDR SDRAM mode configuration 9 (DDR_SDRAM_MODE_9)
0x01080224: DDR SDRAM mode configuration 10 (DDR_SDRAM_MODE_10)
0x01080228: DDR SDRAM mode configuration 11 (DDR_SDRAM_MODE_11)
0x0108022c: DDR SDRAM mode configuration 12 (DDR_SDRAM_MODE_12)
0x01080230: DDR SDRAM mode configuration 13 (DDR_SDRAM_MODE_13)
0x01080234: DDR SDRAM mode configuration 14 (DDR_SDRAM_MODE_14)
0x01080238: DDR SDRAM mode configuration 15 (DDR_SDRAM_MODE_15)
0x0108023c: DDR SDRAM mode configuration 16 (DDR_SDRAM_MODE_16)

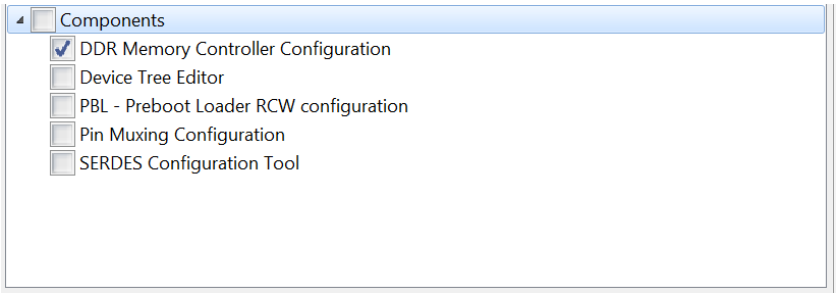
0x01080120: DDR SDRAM mode control (DDR_SDRAM_MD_CNTL)
0x01080124: DDR SDRAM interval configuration (DDR_SDRAM_INTERVAL)
0x01080174: DDR write leveling control (DDR_WRLVL_CNTL)
0x01080190: DDR write leveling control 2 (DDR_WRLVL_CNTL_2)
0x01080194: DDR write leveling control 3 (DDR_WRLVL_CNTL_3)
0x01080120: DDR SDRAM mode control (DDR_SDRAM_MD_CNTL)
0x01080170: DDR ZQ calibration control (DDR_ZQ_CNTL)
0x01080400: DQ mapping register 0 (DDR_DQ_MAP0)
0x01080404: DQ mapping register 1 (DDR_DQ_MAP1)
0x01080408: DQ mapping register 2 (DDR_DQ_MAP2)
0x0108040c: DQ mapping register 3 (DDR_DQ_MAP3)
0x01080260: DDR SDRAM control configuration 3 (DDR_SDRAM_CFG_3)
0x01080128: DDR SDRAM data initialization (DDR_DATA_INIT)
0x0108017c: DDR Self Refresh Counter (DDR_SR_CNTR)
0x01080180: DDR Control Words 1 (DDR_SDRAM_RCW_1)
0x01080184: DDR Control Words 2 (DDR_SDRAM_RCW_2)
0x010801a0: DDR Control Words 3 (DDR_SDRAM_RCW_3)
0x010801a4: DDR Control Words 4 (DDR_SDRAM_RCW_4)
0x010801a8: DDR Control Words 5 (DDR_SDRAM_RCW_5)
0x010801ac: DDR Control Words 6 (DDR_SDRAM_RCW_6)
0x01080b28: DDR Control Driver 1 (DDRCDR_1)
0x01080114: DDR SDRAM control configuration 2 (DDR_SDRAM_CFG_2)
0x01080148: DDR training initialization address (DDR_INIT_ADDR)
0x0108014c: DDR training initialization extended address (DDR_INIT_EXT_ADD
RESS)
0x01080b2c: DDR Control Driver 2(DDRCDR_2)
0x01080e44: Memory error disable (ERR_DISABLE)
0x01080e48: Memory error interrupt enable (ERR_INT_EN)
0x01080114: DDR SDRAM control configuration 2 (DDR_SDRAM_CFG_2)
0x01080110: DDR SDRAM control configuration (DDR_SDRAM_CFG)

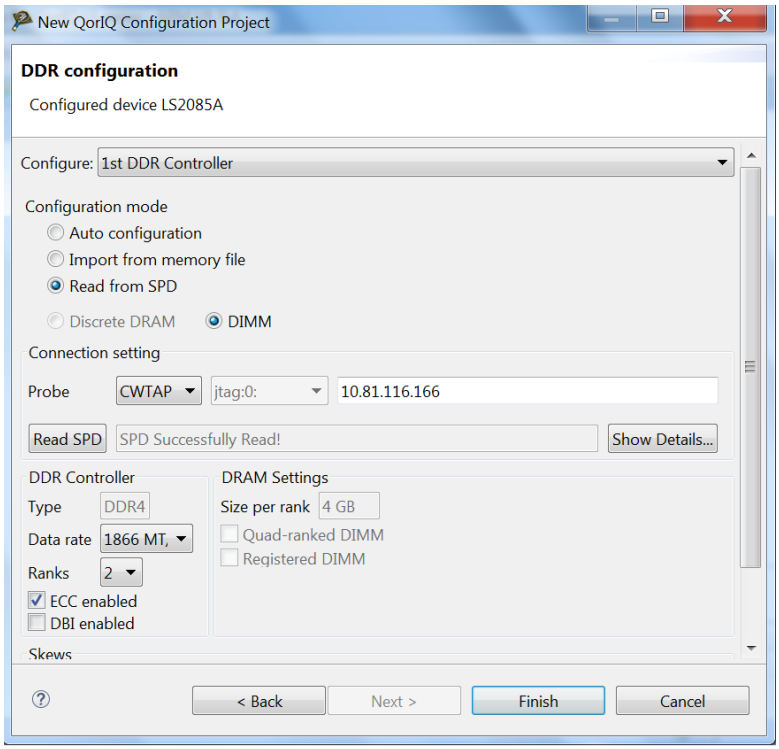
2. Generate Basic DDR Controller Configuration Parameters with QCVS Tool

In this document, reading SPD method is used to create a QCVS project. If there is no DDR SPD on target board, the user needs to configure a QCVS project with automatic way, and configure Properties parameters according the DDR data sheet.

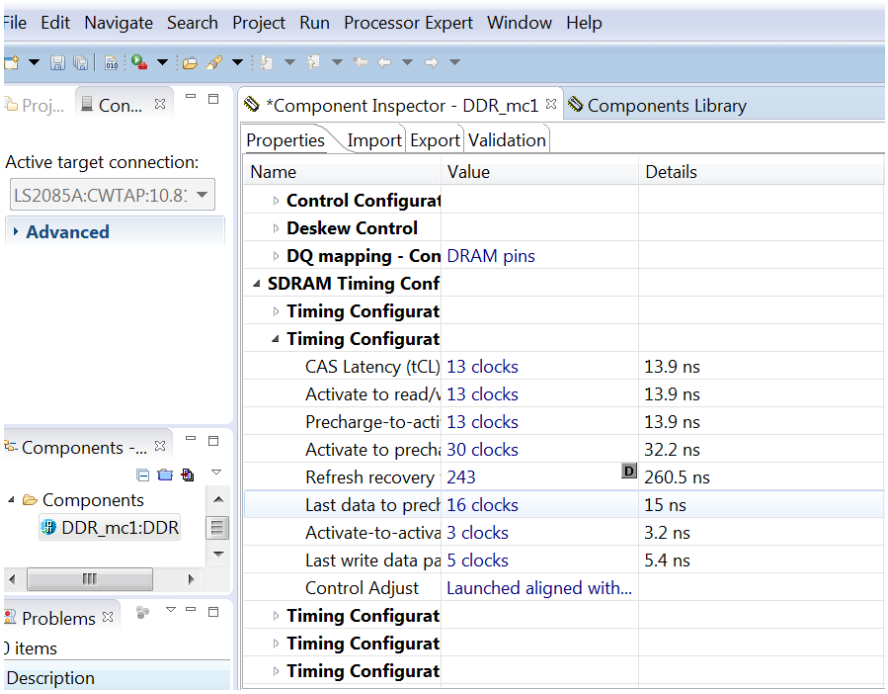
The following figure describes using “Reading from SPD” method to get DDR parameters from the target board on LS2085.

The user need to configure CodeWarrior TAP connection information to connect to the target board, then click “Read SPD” to wait QCVS to read DDR parameters from the target board.
It is needed to configure DDR data rate the same with the target board setting to prepare the setting for validation with DDRv Tool later.





After create the QCVS tool, SDRAM timing configuration is configured as the following based on the parameters read from SPD.



So far, the user could generate the basic DDR controller configuration based on SPD data from Project->Generate Processor Expert Code, the generated files are located “Generated_Code” folder.

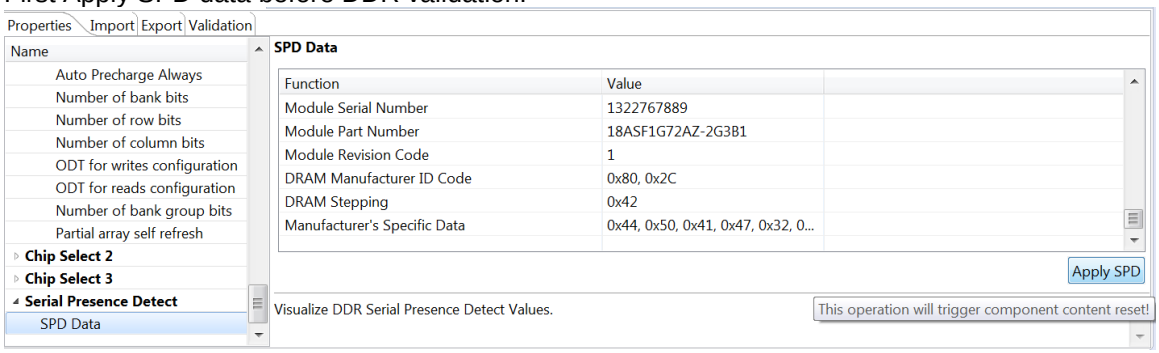
3. Validation and Optimize DDR Controller Configuration with DDRv Tool

DDRv tool is a software component installed on top of QCVS tool, it requires the specific license. The DDRv tool performs validating with a step by step approach, at each step obtaining an optimal DDR configuration with representing input for the next step. DDRv Tool initializes the target board with the basic parameters generated by QCVS in the previous step. At the end of the validation, a tuned up DDR configuration is obtained.

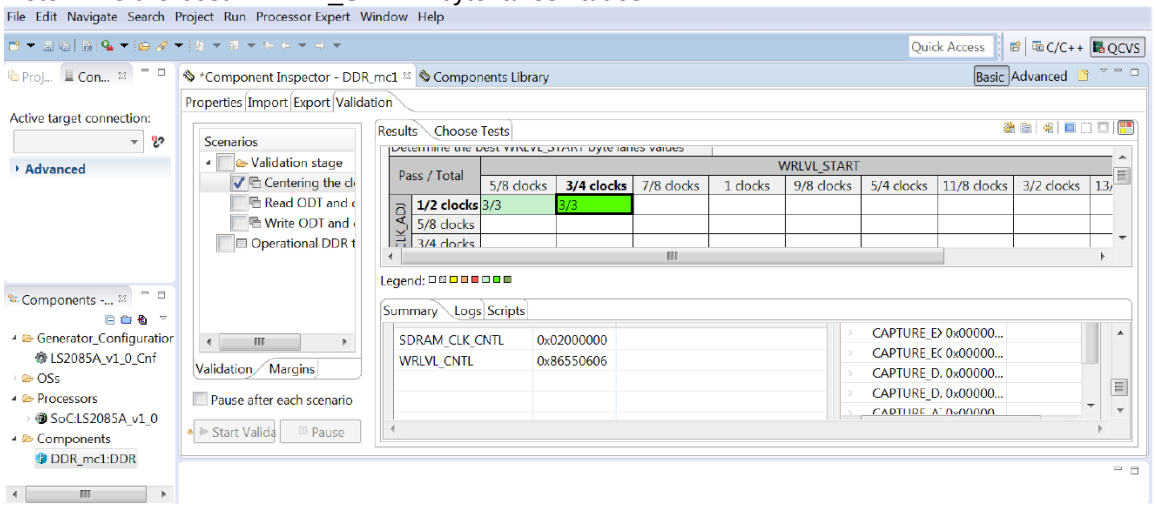
As usual, the user needs to perform validation such as centering the clock, Read ODT and driver and Write ODT and driver, then get updated DDR controller configuration registers.

Centering the clock scenario varies the CLK_ADJ and WRLVL_START DDR controller parameters.

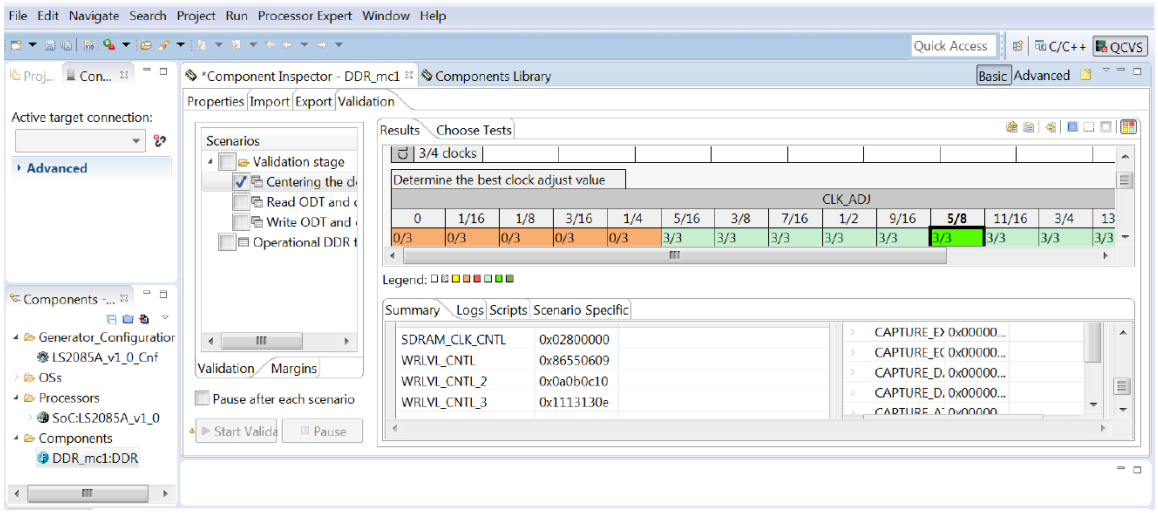
First Apply SPD data before DDR validation:



Determine the best WRLVL_START byte lanes values



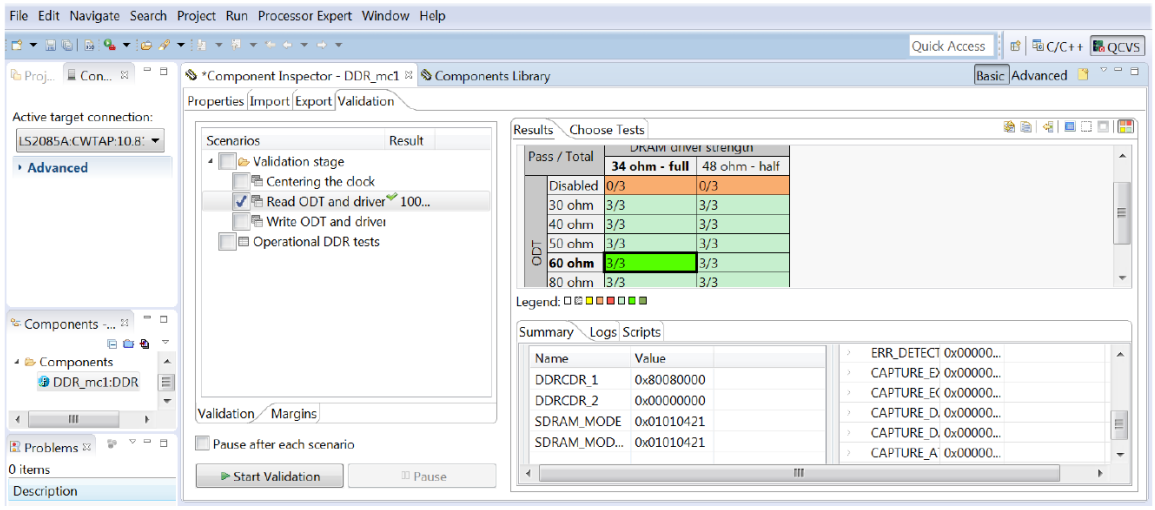
Determine the best clock adjust value.



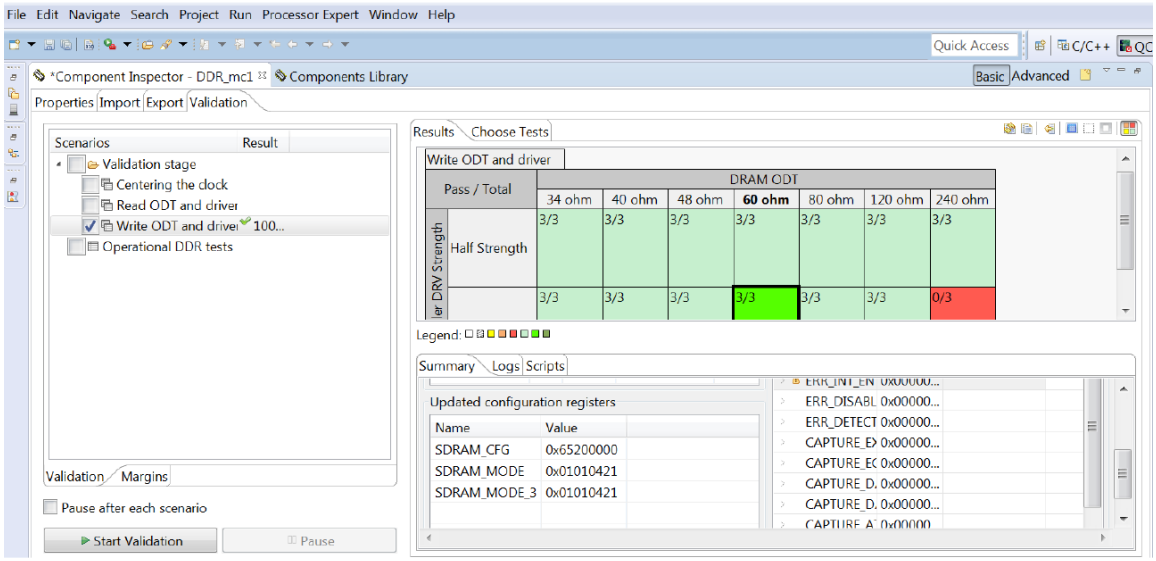
Determine WRLVL margin per byte lane

	Pass / Total	Lane 0	Lane 1	Lane 2	Lane 3	Lane 4
L_START	1/8 clocks	0/3	0/3	0/3	0/3	0/3
	1/4 clocks	0/3	0/3	0/3	0/3	0/3
	3/8 clocks	0/3	0/3	0/3	0/3	0/3
	1/2 clocks	0/3	0/3	0/3	0/3	0/3
	5/8 clocks	0/3	0/3	0/3	0/3	0/3
	3/4 clocks	3/3	3/3	0/3	0/3	0/3
	7/8 clocks	3/3	3/3	0/3	0/3	0/3
	1 clocks	3/3	3/3	3/3	0/3	0/3
	9/8 clocks	3/3	3/3	3/3	3/3	0/3
	5/4 clocks	3/3	3/3	3/3	3/3	0/3
	11/8 clocks	3/3	3/3	1/3	3/3	0/3
	3/2 clocks	3/3	3/3	3/3	3/3	3/3
	13/8 clocks	3/3	3/3	3/3	3/3	3/3
	7/4 clocks	0/3	0/3	3/3	3/3	3/3
	15/8 clocks	0/3	0/3	3/3	3/3	3/3
	2 clocks	0/3	0/3	0/3	3/3	3/3

Read ODT and driver



Write ODT and driver



The validation progress is displayed using colors and numbers. Each cell is colored based on the validation result.
Red: Indicates that validation failed for the cell due to target connection issues or unknown causes.
Yellow: Indicates that validation failed for the cell because it could not pass some tests
Orange: Indicates that validation failed for the cell due to incorrect DDR configuration.
Light green: Indicates that the cell passed all tests.
Bright green: Indicates that the cell has optimal DDR configuration.

Get the optimal DDR configuration registers as the following.

SDRAM_CLK_CNTL	0x02000000
WRLVL_CNTL	0x86550606
WRLVL_CNTL_2	0x0a0b0c10
WRLVL_CNTL_3	0x1113130e
DDRCDR_1	0x80080000
DDRCDR_2	0x00000000
SDRAM_CFG	0xe5200000
SDRAM_MODE	0x01010421
SDRAM_MODE_3	0x01010421

Regenerate DDR configuration script from Project->Generate Processor Expert code, DDR controller configuration the script ddrCtrl_1.py will be used in CodeWarrior initialization file and u-boot.

4. Porting CodeWarrior initialization file with the custom DDR configuration parameters

After follow create an ARMv8 bare board project following the new project wizards, please open the initialization file panel from Run->Debug Configuration->GDB Hardware Debugging-><project>->Debugger->Configure Target Connection, duplicate LS2085ARDB target connection configuration and edit it to pen Target Init File, and modify DDR Initialization section as the following.

*CCSR_LE_M(0x0000000001080130 + id * 0x10000, 0x02000000)*
*CCSR_LE_M(0x0000000001080000 + id * 0x10000, 0x000000ff)*
*CCSR_LE_M(0x0000000001080080 + id * 0x10000, 0x80040322)*

```
    CCSR_LE_M(0x00000000010800c0 + id * 0x10000, 0x00000000)
    CCSR_LE_M(0x0000000001080008 + id * 0x10000, 0x000001ff)
    CCSR_LE_M(0x0000000001080084 + id * 0x10000, 0x80000322)
    CCSR_LE_M(0x00000000010800c4 + id * 0x10000, 0x00000000)
    CCSR_LE_M(0x0000000001080010 + id * 0x10000, 0x0200023F)
    CCSR_LE_M(0x0000000001080088 + id * 0x10000, 0x00010202)
    CCSR_LE_M(0x00000000010800c8 + id * 0x10000, 0x00000000)
    CCSR_LE_M(0x0000000001080018 + id * 0x10000, 0x0240027F)
    CCSR_LE_M(0x000000000108008c + id * 0x10000, 0x00010202)
    CCSR_LE_M(0x00000000010800cc + id * 0x10000, 0x00000000)

elif (DDR_CLK < DDR2133_TH):
    # 1866
    CCSR_LE_M(0x0000000001080100 + id * 0x10000, 0x010e1100)
    CCSR_LE_M(0x0000000001080104 + id * 0x10000, 0x80660018)
    CCSR_LE_M(0x0000000001080108 + id * 0x10000, 0xded8b035)
    CCSR_LE_M(0x000000000108010c + id * 0x10000, 0x00513154)
    CCSR_LE_M(0x0000000001080160 + id * 0x10000, 0x00220001)
    CCSR_LE_M(0x0000000001080164 + id * 0x10000, 0x04401400)
    CCSR_LE_M(0x0000000001080168 + id * 0x10000, 0x00000000)
    CCSR_LE_M(0x000000000108016c + id * 0x10000, 0x15500000)
    CCSR_LE_M(0x0000000001080250 + id * 0x10000, 0x03335900)
    CCSR_LE_M(0x0000000001080118 + id * 0x10000, 0x01010421)
    CCSR_LE_M(0x000000000108011c + id * 0x10000, 0x00080000)
    CCSR_LE_M(0x0000000001080200 + id * 0x10000, 0x01010421)
    CCSR_LE_M(0x0000000001080204 + id * 0x10000, 0x00080000)
    CCSR_LE_M(0x0000000001080208 + id * 0x10000, 0x00000000)
    CCSR_LE_M(0x000000000108020c + id * 0x10000, 0x00000000)
    CCSR_LE_M(0x0000000001080210 + id * 0x10000, 0x00000000)
    CCSR_LE_M(0x0000000001080214 + id * 0x10000, 0x00000000)
    CCSR_LE_M(0x0000000001080220 + id * 0x10000, 0x00000500)
    CCSR_LE_M(0x0000000001080224 + id * 0x10000, 0x04000000)
    CCSR_LE_M(0x0000000001080228 + id * 0x10000, 0x00000400)
    CCSR_LE_M(0x000000000108022c + id * 0x10000, 0x04000000)
    CCSR_LE_M(0x0000000001080230 + id * 0x10000, 0x00000000)
    CCSR_LE_M(0x0000000001080234 + id * 0x10000, 0x00000000)
    CCSR_LE_M(0x0000000001080238 + id * 0x10000, 0x00000000)
    CCSR_LE_M(0x000000000108023c + id * 0x10000, 0x00000000)
    CCSR_LE_M(0x0000000001080120 + id * 0x10000, 0x00000000)
    CCSR_LE_M(0x0000000001080124 + id * 0x10000, 0x1c6d071b)
    CCSR_LE_M(0x0000000001080174 + id * 0x10000, 0x86550606)
    CCSR_LE_M(0x0000000001080190 + id * 0x10000, 0x0a0b0c10)
    CCSR_LE_M(0x0000000001080194 + id * 0x10000, 0x1112130e)

    CCSR_LE_M(0x0000000001080120 + id * 0x10000, 0x862004c0)
    CCSR_LE_M(0x0000000001080120 + id * 0x10000, 0x862004f0)
    CCSR_LE_M(0x0000000001080120 + id * 0x10000, 0x86200470)
    CCSR_LE_M(0x0000000001080120 + id * 0x10000, 0x962004c0)
    CCSR_LE_M(0x0000000001080120 + id * 0x10000, 0x962004f0)
    CCSR_LE_M(0x0000000001080120 + id * 0x10000, 0x96200470)


    CCSR_LE_M(0x0000000001080254 + id * 0x10000, 0x00000000)
    CCSR_LE_M(0x0000000001080170 + id * 0x10000, 0x8a090705)
    CCSR_LE_M(0x0000000001080400 + id * 0x10000, 0x32c57554)
    CCSR_LE_M(0x0000000001080404 + id * 0x10000, 0xd4bb0bd4)
```

```
CCSR_LE_M(0x0000000001080408 + id * 0x10000, 0x2ec2f554)
CCSR_LE_M(0x000000000108040c + id * 0x10000, 0xd95d4001)
CCSR_LE_M(0x0000000001080260 + id * 0x10000, 0x00000000)

CCSR_LE_M(0x0000000001080128 + id * 0x10000, 0xdeadbeef)
CCSR_LE_M(0x000000000108017c + id * 0x10000, 0x00000000)
CCSR_LE_M(0x0000000001080180 + id * 0x10000, 0x00000000)
CCSR_LE_M(0x0000000001080184 + id * 0x10000, 0x00000000)
CCSR_LE_M(0x00000000010801a0 + id * 0x10000, 0x00000000)
CCSR_LE_M(0x00000000010801a4 + id * 0x10000, 0x00000000)
CCSR_LE_M(0x00000000010801a8 + id * 0x10000, 0x00000000)
CCSR_LE_M(0x00000000010801ac + id * 0x10000, 0x00000000)
CCSR_LE_M(0x0000000001080b28 + id * 0x10000, 0x80080000)
CCSR_LE_M(0x0000000001080114 + id * 0x10000, 0x00401151)
CCSR_LE_M(0x0000000001080148 + id * 0x10000, 0x00000000)
CCSR_LE_M(0x000000000108014c + id * 0x10000, 0x00000000)
CCSR_LE_M(0x0000000001080b2c + id * 0x10000, 0x00000000)
CCSR_LE_M(0x0000000001080e44 + id * 0x10000, 0x00000000)
CCSR_LE_M(0x0000000001080e48 + id * 0x10000, 0x0000001d)
CCSR_LE_M(0x0000000001080f70 + id * 0x10000, 0x00000400)
CCSR_LE_M(0x0000000001080114 + id * 0x10000, 0x00401151)
CCSR_LE_M(0x0000000001080110 + id * 0x10000, 0xe5200000)
CCSR_LE_M(0x0000000001080110 + id * 0x10000, 0xe5200000)

CCSR_LE_M(0x0000000001080f70 + id * 0x10000, 0x00000000)
CCSR_LE_M(0x0000000001080f04 + id * 0x10000, 0x00000400)
```

5. Porting U-BOOT Source with the custom DDR configuration parameters

In u-boot source of LS2085/LS2080, reading from DDR SPD is used, in the file board/freescale/ls208xardb/ddr.c, clk_adjust, wrlvl_start, wrlvl_ctl parameters are got from board_specific_parameters table defined in board/freescale/ls208xardb/ddr.h.

```
ddr_freq = get_ddr_freq(ctrl_num) / 1000000;

while (pbsp->datarate_mhz_high) {
    if (pbsp->n_ranks == pdimm[slot].n_ranks &&
       (pdimm[slot].rank_density >> 30) >= pbsp->rank_gb) {
        if (ddr_freq <= pbsp->datarate_mhz_high) {
            popts->clk_adjust = pbsp->clk_adjust;
            popts->wrlvl_start = pbsp->wrlvl_start;
            popts->wrlvl_ctl_2 = pbsp->wrlvl_ctl_2;
            popts->wrlvl_ctl_3 = pbsp->wrlvl_ctl_3;
            goto found;
        }
        pbsp_highest = pbsp;
    }
```

```
            pbsp++;

        }
```

Modify board_specific_parameters table rdimm0 according to the optimized parameters WRLVL_CNTL, WRLVL_CNTL_2, WRLVL_CNTL_3 validated by DDRv Tool.

```
static const struct board_specific_parameters rdimm0[] = {
        /*
         * memory controller 0
         *   num|  hi| rank|  clk| wrlvl |   wrlvl   |  wrlvl
         * ranks| mhz| GB  |adjst| start |   ctl2    |  ctl3
         */
        {2,  1350, 0, 4,     6, 0x0708090B, 0x0C0D0E09,},
        {2,  1666, 0, 4,     7, 0x08090A0C, 0x0D0F100B,},
        {2,  1900, 0, 4,     6, 0x0a0b0c10, 0x1113130e,},
        {2,  2200, 0, 4,     8, 0x090A0C0F, 0x1012130C,},
        {}
};
```