/\* ============================================================================

\* Copyright (c) Texas Instruments Inc 2002, 2003, 2004, 2005

\*

\* Use of this software is controlled by the terms and conditions found in the

\* license agreement under which this software has been supplied.

\* ===========================================================================

\*/

/\*\* ============================================================================

\* @file Pllc\_example.c

\*

\*

\* @desc Example of PLLC module CSL

\* =============================================================================

\* @n <b> Test Description </b>

\* @n The example shows how to test the PLLC

\* 1. Initalize the PLLC hardware setup data structure with default values

\* 2. Intialize PLLC module

\* 3. Set new values for divider3

\* 4. Set the value using hardware control command

\* 5. Read and verify the new values using hardware query

\* 6. Close the module

\*==============================================================================

\*

\* <b> Test Procedure </b>

\* @verbatim

\* 1. Configure the CCS setup to work with the emulator being used.

\* Refer CCS manual for setup configuration.

\* 2. Load required GEL file depending on the board used.

\* 3. Launch CCS window.

\* 4. Open project pllc\_example.pjt.

\* 5. Build the project and load the .out file of the project.

\* 6. The test can be executed from the main().

\*

\* @endverbatim

\*

\*

\* =============================================================================

\*/

/\* =============================================================================

\* Revision History

\* ===============

\* 12-may-2005 Ganesh K File created.

\*

\* =============================================================================

\*/

**#include** "Pllc\_example.h"

**#include** "C6727Driver.c"

/\* PLLC module handle structure \*/

CSL\_PllcHandle hPllc = NULL;

/\* Variable to count the test failure \*/

Uint32 pllcTestFail = PLLC\_TEST\_PASSED;

/\*

\* =============================================================================

\* @func main

\*

\* @desc

\* This is the main routine for the file.

\*

\* @arg

\* NONE

\*

\* @return

\* NONE

\* =============================================================================

\*/

**void**

**main** (

**void**

)

{

//InitPll();

/\* Perform the test \*/

pllc\_csl\_example\_test ();

/\* Check for errors in test \*/

**if** (pllcTestFail == PLLC\_TEST\_PASSED) {

**printf** ("\nPllc CSL Example Test PASSED\n");

}

**else** {

**printf** ("\nPllc CSL Example Test FAILED\n");

}

}

/\*

\* =============================================================================

\* @func pllc\_csl\_example\_test

\*

\* @desc

\* This is the test routine which write, read and verify divider3 value.

\*

\* @arg

\* NONE

\*

\* @return

\* NONE

\* =============================================================================

\*/

**void**

**pllc\_csl\_example\_test** (

**void**

)

{

CSL\_PllcObj pllcObj;

CSL\_Status status;

CSL\_PllcHwSetup hwSetup = CSL\_PLLC\_HWSETUP\_DEFAULTS;

CSL\_PllcDivCntrl arg;

CSL\_PllcDivCntrl response;

**CSL\_sysInit**();

/\* Initialize the pllc CSL module \*/

status = **CSL\_pllcInit** (NULL);

**if** (status != CSL\_SOK) {

pllcTestFail++;

**printf** ("PLLC: Initialization... Failed.\n");

**printf** ("\tReason: CSL\_pllcInit failed [Status: %d] \n ", status);

**return**;

}

**else** {

**printf** ("PLLC: Module Initialization... Passed.\n");

}

/\* Open the pllc CSL module \*/

hPllc = **CSL\_pllcOpen** (&pllcObj, CSL\_PLLC, NULL, &status);

**if** ((hPllc == NULL) || (status != CSL\_SOK)) {

pllcTestFail++;

**printf** ("PLLC: Opening the instance ... Failed\n");

**return**;

}

**else** {

**printf** ("PLLC: Open valid instance... Passed.\n");

}

/\* Setup PLLC hardware parameters \*/

status = CSL\_pllcHwSetup (hPllc, &hwSetup);

**if** (status != CSL\_SOK) {

pllcTestFail++;

**printf** ("PLLC: Setup of the module ... Failed\n");

pllc\_error\_exit ();

**return**;

}

**else** {

**printf** ("PLLC: Setup of the module ... Passed.\n");

}

/\* Divider 3 Setup structure intialization \*/

arg.divNum = *CSL\_PLLC\_DIV3*;

arg.divControl.divEnable = *CSL\_PLLC\_PLLDIV\_ENABLE*;

arg.divControl.pllDivRatio = PLLC\_DIV3\_RATIO\_NEW\_VAL;

/\* Set the new value \*/

status = **CSL\_pllcHwControl** (hPllc, *CSL\_PLLC\_CMD\_DIV\_CONTROL*, &arg);

**if** (status != CSL\_SOK) {

pllcTestFail++;

**printf** ("PLLC: Hardware Command set Div3 ... Failed\n");

pllc\_error\_exit ();

**return**;

}

**else** {

**printf** ("PLLC: Hardware Command set Div3 ... Passed\n");

}

/\* Get divider 3 status \*/

response.divNum = *CSL\_PLLC\_DIV3*;

status =

**CSL\_pllcGetHwStatus** (hPllc, *CSL\_PLLC\_QUERY\_DIVIDER\_STATE*, &response);

**if** (status != CSL\_SOK) {

pllcTestFail++;

**printf** ("PLLC: Hardware Query Div3 ... Failed\n");

pllc\_error\_exit ();

**return**;

}

**else** {

**printf** ("PLLC: Hardware Query Div3 ... Passed\n");

}

/\* Check for divider enable and divder ratio \*/

**if** ((response.divControl.divEnable != *CSL\_PLLC\_PLLDIV\_ENABLE*) ||

(response.divControl.pllDivRatio != PLLC\_DIV3\_RATIO\_NEW\_VAL)) {

pllcTestFail++;

**printf** ("PLLC: Hardware set Div3 ... Failed\n");

}

**else** {

**printf** ("PLLC: Hardware set Div3 ... Passed\n");

}

/\* Check closing the module \*/

status = **CSL\_pllcClose** (hPllc);

**if** (status != CSL\_SOK) {

pllcTestFail++;

**printf** ("PLLC: Close the instance ... Failed\n");

}

**else** {

**printf** ("PLLC: Close valid instance... Passed.\n");

}

**return**;

}

/\*

\* =============================================================================

\* @func pllc\_error\_exit

\*

\* @desc

\* This calls the module close function in case of error

\*

\* @arg

\* NONE

\*

\* @return

\* NONE

\* =============================================================================

\*/

**void**

**pllc\_error\_exit** (

**void**

)

{

CSL\_Status status;

/\* Check closing the module \*/

status = **CSL\_pllcClose** (hPllc);

**if** (status != CSL\_SOK) {

pllcTestFail++;

**printf** ("PLLC: Close Pllc instance... Failed\n");

}

**else** {

**printf** ("PLLC: Close Pllc instance... Passed\n");

}

}

**#include** <csl\_pllc.h>

**#pragma** CODE\_SECTION (CSL\_pllcHwSetup, ".text:csl\_section:pllc");

CSL\_Status **CSL\_pllcHwSetup** (

CSL\_PllcHandle hPllc,

CSL\_PllcHwSetup \* hwSetup

)

{

CSL\_PllcGoStatus goStat;

CSL\_Status status = CSL\_SOK;

**volatile** Uint32 loopCount;

**if** (hPllc == NULL) {

**return** CSL\_ESYS\_BADHANDLE;

}

**if** (hwSetup == NULL) {

**return** CSL\_ESYS\_INVPARAMS;

}

/\* Set PLLEN to Bypass mode \*/

CSL\_FINS (hPllc->regs->PLLCSR, PLLC\_PLLCSR\_PLLEN,

CSL\_PLLC\_PLLCSR\_PLLEN\_BYPASS);

CSL\_FINS (hPllc->regs->PLLCSR,PLLC\_PLLCSR\_PLLPWRDN,0x00000010u);

/\*

\* wait for 4 clock cycles, and ensure PLLEN mux switches to

\* bypass clock

\*/

/\* DELAY\_200MS arbitrary delay for synchronisation \*/

loopCount = DELAY\_200MS;

**while** (loopCount--) {

**asm** (" NOP");

}

CSL\_FINS (hPllc->regs->PLLCSR, PLLC\_PLLCSR\_OSCPWRDN,

0x00000000u);

hPllc->regs->PLLCSR &= 0x00000040u;

loopCount = DELAY\_200MS;

**while** (loopCount--) {

**asm** (" NOP");

}

/\* Set PLLRST to ASSERTED \*/

CSL\_FINS (hPllc->regs->PLLCSR, PLLC\_PLLCSR\_PLLRST,

CSL\_PLLC\_PLLCSR\_PLLRST\_ASSERTED);

/\* If Pll Mode program PLLDIV0 and PLLM \*/

**if** (hwSetup->pllcMode == CSL\_PLLC\_PLLCSR\_PLLEN\_PLL) {

/\* Program Divider D0 PLLDIV0 \*/

**if** (hwSetup->div0Enable.divEnable == CSL\_PLLC\_PLLDIV0\_D0EN\_ENABLE) {

CSL\_FINS (hPllc->regs->PLLDIV0, PLLC\_PLLDIV0\_D0EN,

CSL\_PLLC\_PLLDIV0\_D0EN\_ENABLE);

CSL\_FINS (hPllc->regs->PLLDIV0, PLLC\_PLLDIV0\_RATIO,

hwSetup->div0Enable.pllDivRatio);

}

/\* Program PLLM \*/

CSL\_FINS (hPllc->regs->PLLM, PLLC\_PLLM\_PLLM, hwSetup->pllM);

/\* arbitrary delay for PLL clock lock \*/

loopCount = DELAY\_200MS;

**while** (loopCount--)

**asm** (" NOP");

}

/\* Wait if any of go set operation is in progress \*/

**do** {

goStat = (CSL\_PllcGoStatus) CSL\_FEXT (hPllc->regs->PLLSTAT,

PLLC\_PLLSTAT\_GOSTAT);

} **while** (goStat == CSL\_PLLC\_PLLSTAT\_GOSTAT\_INPROG);

/\* Program Divider 1 to 3 \*/

**if** (hwSetup->div1Enable.divEnable == CSL\_PLLC\_PLLDIV1\_D1EN\_ENABLE) {

CSL\_FINS (hPllc->regs->PLLDIV1, PLLC\_PLLDIV1\_D1EN,

CSL\_PLLC\_PLLDIV1\_D1EN\_ENABLE);

CSL\_FINS (hPllc->regs->PLLDIV1, PLLC\_PLLDIV0\_RATIO,

hwSetup->div1Enable.pllDivRatio);

}

**if** (hwSetup->div2Enable.divEnable == CSL\_PLLC\_PLLDIV2\_D2EN\_ENABLE) {

CSL\_FINS (hPllc->regs->PLLDIV2, PLLC\_PLLDIV2\_D2EN,

CSL\_PLLC\_PLLDIV2\_D2EN\_ENABLE);

CSL\_FINS (hPllc->regs->PLLDIV2, PLLC\_PLLDIV2\_RATIO,

hwSetup->div2Enable.pllDivRatio);

}

**if** (hwSetup->div3Enable.divEnable == CSL\_PLLC\_PLLDIV0\_D0EN\_ENABLE) {

CSL\_FINS (hPllc->regs->PLLDIV3, PLLC\_PLLDIV3\_D3EN,

CSL\_PLLC\_PLLDIV3\_D3EN\_ENABLE);

CSL\_FINS (hPllc->regs->PLLDIV3, PLLC\_PLLDIV3\_RATIO,

hwSetup->div3Enable.pllDivRatio);

}

/\* set Align sysclk1 with sysclk \*/

CSL\_FINS (hPllc->regs->ALNCTL, PLLC\_ALNCTL\_ALN1,

CSL\_PLLC\_ALNCTL\_ALN1\_ALIGN);

/\* set Align sysclk1 with sysclk \*/

CSL\_FINS (hPllc->regs->ALNCTL, PLLC\_ALNCTL\_ALN2,

CSL\_PLLC\_ALNCTL\_ALN2\_ALIGN);

/\* set Align sysclk1 with sysclk \*/

CSL\_FINS (hPllc->regs->ALNCTL, PLLC\_ALNCTL\_ALN3,

CSL\_PLLC\_ALNCTL\_ALN3\_ALIGN);

/\* Set the GOSET bit in PLLCMD register \*/

CSL\_FINS (hPllc->regs->PLLCMD, PLLC\_PLLCMD\_GOSET,

CSL\_PLLC\_PLLCMD\_GOSET\_SET);

/\* wait till Align \*/

**do** {

goStat = (CSL\_PllcGoStatus) CSL\_FEXT (hPllc->regs->PLLSTAT,

PLLC\_PLLSTAT\_GOSTAT);

} **while** (goStat == CSL\_PLLC\_PLLSTAT\_GOSTAT\_INPROG);

/\* arbitrary delay for synchronisation \*/

loopCount = DELAY\_200MS;

**while** (loopCount--)

**asm** (" NOP");

/\* Set PLLRST = 0 \*/

/\* Set PLLRST to ASSERTED \*/

CSL\_FINS (hPllc->regs->PLLCSR, PLLC\_PLLCSR\_PLLRST,

CSL\_PLLC\_PLLCSR\_PLLRST\_RELEASED);

/\* Wait for PLL to lock \*/

/\* arbitrary delay for synchronisation \*/

loopCount = DELAY\_200MS;

**while** (loopCount--)

**asm** (" NOP");

**if** (hwSetup->pllcMode == CSL\_PLLC\_PLLCSR\_PLLEN\_PLL) {

/\* Set PLLEN = 1 \*/

CSL\_FINS (hPllc->regs->PLLCSR, PLLC\_PLLCSR\_PLLEN,

CSL\_PLLC\_PLLCSR\_PLLEN\_PLL);

}

loopCount = DELAY\_200MS;

**while** (loopCount--)

**asm** (" NOP");

**return** status;

}