Saturn Memory

PLL and DDR2 Setup Guide

Contents

1. Introduction

This document describes the steps necessary to configure DDR2 on Saturn Platforms.

The DDR2 settings change with:

- Clock Speed
- The specific DDR2 part
- PCB Design

1. Clock speed

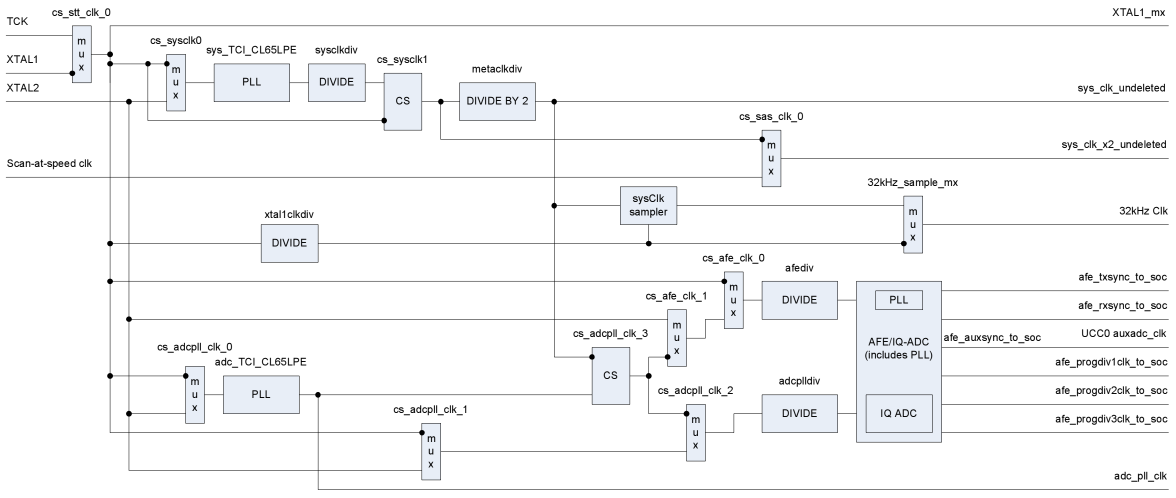
The DDR2 runs at the PLL speed, or a fraction of it as determined by dividers. It is not practical to run the DDR2 directly from the XTAL as the XTAL speeds are too slow.

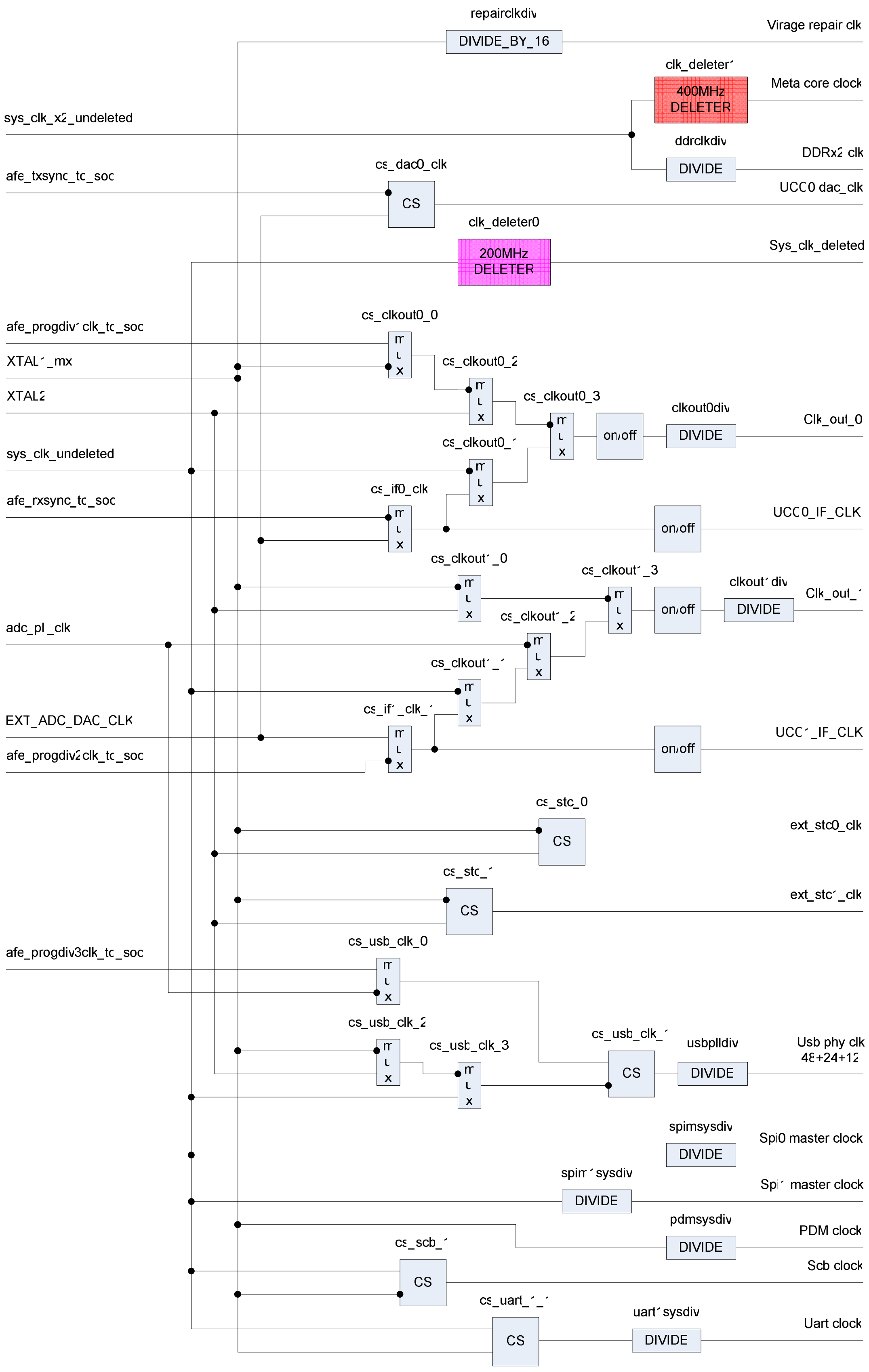
1.1. Relationship between PLL & DDR2 CLK

The DDR2 clock is the result of the PLL being divided by these two registers:

- CR_TOP_SYSCLK_DIV *<- Not normally used to alter DDR2 frequency as it affects sys_clk*
- CR_TOP_DDR_CLKDIV *<- Usually used to set the DDR2 Frequency to a fraction of the PLL*

See the diagram below:




repairclkdiv
DIVIDE_BY_16
Virage repair clk
clk_deleter1
400MHz DELETER
Meta core clock
sys_clk_x2_undeleted
ddrclkdiv
DIVIDE
DDRx2 clk
afe_txsync_tc_soc
cs_dac0_clk
CS
UCC0 dac_clk
clk_deleter0
200MHz DELETER
Sys_clk_deleted
afe_progdiv1clk_tc_soc
cs_clkout0_0
mux
XTAL1_mx
cs_clkout0_2
XTAL2
cs_clkout0_3
on/off
clkout0div
DIVIDE
Clk_out_0
sys_clk_undeleted
cs_clkout0_1
afe_rxsync_tc_soc
cs_if0_clk
on/off
UCC0_IF_CLK
cs_clkout1_0
cs_clkout1_3
clkout1div
Clk_out_1
cs_clkout1_2
adc_pl_clk
cs_clkout1_1
cs_if1_clk_1
EXT_ADC_DAC_CLK
UCC1_IF_CLK
afe_progdiv2clk_tc_soc
cs_stc_0
ext_stc0_clk
cs_stc_1
ext_stc1_clk
afe_progdiv3clk_tc_soc
cs_usb_clk_0
cs_usb_clk_2
cs_usb_clk_3
cs_usb_clk_1
usbplldiv
Usb phy clk 48+24+12
spimsysdiv
Spi0 master clock
spim1sysdiv
Spi1 master clock
pdmsysdiv
PDM clock
cs_scb_1
Scb clock
cs_uart_1_1
uart1sysdiv
Uart clock

1.2. Typical divider settings

Unless there is a requirement to use less power, or the PLL is running faster than the maximum frequency for the DDR2 part to be used, CR_TOP_DDR_CLKDIV is usually set to 0 to not divide.

1.3. Configuring the PLL

The PLL is normally configured in the .img file. Example img files are provided in the package (saturn_apis\reference\build\saturn\meta\es1). However, if you require different clock speeds, you can create a custom .img file using PllGen, which will generate PLL settings given a target frequency, and the XTAL speed. PLLGen can be found in the IMGWorks pagckage (saturn_apis\user\utils\win32\PLLGen).

1.3.1. Worked example: Converting a 400MHz .img file to 380MHz.

Original file:

```
/* PLL Config */
/* 400MHz output from 24.576 Crystal */
/************* WARNING - if this is modified, please also modify the TIMERCONFIG value below */
MWR 0x02015908 0x00000000 * CR_TOP_CLKSWITCH        - Ensure Clk source is XTAL not PLL */
MPAUSE 1000               /*                        - Wait for the switch over to happen*/
MWR 0x02015954 0x16000008 /* CR_TOP_SYSPLL_CTL1     - Put the PLL into reset */
/*set dividers first */
MWR 0x020159AC 0x00000000 /* CR_TOP_DDRCLKDIV       - Set to 1 to divide by 2, or 0 for DDR = Meta Freq */
MWR 0x02015918 0x00000001 /* CR_TOP_META_CLKDIV     - Turn on the clock divide to get 200MHz (UCCs and
Peripherals)*/
MWR 0x02015950 0x09101240 /* CR_TOP_SYSPLL_CTL0     - Part of the Pll config for 400MHz */
MPAUSE 1000               /*                        - Wait for a bit */
MWR 0x02015954 0x06000008 /* CR_TOP_SYSPLL_CTL1     - Take the PLL out of reset */
MWR 0x02015954 0x04000008 /* CR_TOP_SYSPLL_CTL1     - Take the PLL out of bypass */
MPAUSE 2000               /*                        - Wait for a bit more */
MWR 0x02015908 0x00000002 /* CR_TOP_CLKSWITCH       - Switch from XTAL to PLL */
```

Output from PLLGen

```
C:\saturn\win32\pllgen\Debug>pllgen -x D -t 380

===========================================================================
=                                                                         =
=                      Saturn PLL Application                             =
=                (Build date May  23 2011, time 14:53:06)                 =
=                                                                         =
===========================================================================

PllGen - Copyright Imagination Technologies (c) 2011
Input frequency  :          24.576 MHz

Actual frequency :          379.983 MHz

0x02015950 CR_TOP_SYSPLL_CTL0[31..0]: 0x0C801910
0x02015954 CR_TOP_SYSPLL_CTL1[5..0] : 0x0000000C
```

Modified file now running the PLL at 380MHz

```
/* PLL Config */
/* 380MHz output from 24.576 Crystal */
/************* WARNING - if this is modified, please also modify the TIMERCONFIG value below */
MWR 0x02015908 0x00000000 /* CR_TOP_CLKSWITCH      - Ensure Clk source is XTAL not PLL */
MPAUSE 1000               /*                       - Wait for the switch over to happen*/
MWR 0x02015954 0x1600000C /* CR_TOP_SYSPLL_CTL1    - Put the PLL into reset */
/*set dividers first */
MWR 0x020159AC 0x00000000 /* CR_TOP_DDRCLKDIV      - Set to 1 to divide by 2, or 0 for DDR = Meta Freq */
MWR 0x02015918 0x00000001 /* CR_TOP_META_CLKDIV    - Turn on the clock divide to get 190MHz (UCCs and
Peripherals)*/
MWR 0x02015950 0x0C801910 /* CR_TOP_SYSPLL_CTL0    - Part of the Pll config for 380MHz */
MPAUSE 1000               /*                       - Wait for a bit */
MWR 0x02015954 0x0600000C /* CR_TOP_SYSPLL_CTL1    - Take the PLL out of reset */
MWR 0x02015954 0x0400000C /* CR_TOP_SYSPLL_CTL1    - Take the PLL out of bypass */
MPAUSE 2000               /*                       - Wait for a bit more */
MWR 0x02015908 0x00000002 /* CR_TOP_CLKSWITCH      - Switch from XTAL to PLL*/
```

Watch out for:

Note that one of the generated register writes is not the full register size, so when changing the img file, it is important to retain the upper bits.

- CTL1 setting only applies to bits 5:0, so preserve the remaining bits
- CTL1 setting must be applied in three places!

2. Obtaining DDR2 settings

2.1. DDR2 Part specific settings

The part specific settings can be obtained from a spreadsheet maintained by the SOC Hardware team. See:
\\kldata2\ImgWorks\Projdocs\Saturn\Software\Documents\Saturn_ddr_settings_v<ChooseLatestVersion>.xls

Select the tab that corresponds to the particular memory part that you are using, then at the top of the page, fill in the frequency that you will be running the DDR2 at, and the CAS latency. The spreadsheet then calculates the appropriate values to set the registers to. Ignore any marked "BOARD_SPECIFIC", unless you are using a saturn bring up board, in which case they are correct.

2.2. Board specific settings

Some of the DDR2 settings are specific to the physical board layout, taking into account track lengths and termination resistances etc. These are the settings in the spreadsheet that are marked "BOARD_SPECIFIC". If you are not using a saturn bring up board, then you need to obtain these values manually using a scope to verify timing. More details can be provided by the IMGWorks HW team.

3. .img files

Now that you have the correct settings to use, it is important to set them up before any accesses are made to a DDR2 address. Typically, the DDR2 setup is contained within one or more .img files. These files are then passed to the linker during the build. ldlk will then generate appropriate register writes based on those .img files, and place them in:

- a .js file (if you are building an application to run via codescape) or
- an LDR file (if you are building a bootable application).

3.1. .img file order

Whilst most DDR2 configuration registers can be written in any order, as a block they should be written after the PLL has been initialised, and after the clock divider has been set.

Here is an example order:

```
< Setup the PLL ,CR_TOP_DDRCLKDIV and CR_TOP_META_CLKDIV>
<Soft Reset the DDR2 Controller>
<Set DDR2 Configuration registers>
<Take DDR2 Controller out of Soft Reset>
MPAUSE 7000
```

3.2. Example .img file

```
/* Make all executing threads privileged */
MWR 0x04800010 0x00020000 /* Thread 0 */
MWR 0x04801010 0x00020000 /* Thread 1 */

/* Create TBI data structure (used by MeOS) */
TBI INT  THREAD 0
TBI BGND THREAD 0
TBI INT  THREAD 1
TBI BGND THREAD 1


/*****************************************/
/* Configure the PLL and clock dividers */
/*****************************************/

/* ********** */
/* 400.043MHz output from 24.576 Crystal
/* ********** */
************* WARNING - if this is modified, please also modify the TIMERCONFIG value below */
MWR 0x02015908 0x00000000 /* Bypass the PLL */
MPAUSE 1000               /* Wait for the switch over to happen*/
MWR 0x02015954 0x16000008 /* Put the PLL into reset */
/*set dividers first */
MWR 0x020159AC 0x00000000 /* CR_TOP_DDRCLKDIV - Set to 1 to divide by 2, or 0 for DDR = Meta Freq */
MWR 0x02015918 0x00000001 /* Turn on the sys clock divide to get 200MHz (peripherals)*/
MWR 0x02015950 0x09101240
MPAUSE 1000               /* Wait for a bit*/
MWR 0x02015954 0x06000008 /* Take the PLL out of reset */
MWR 0x02015954 0x04000008
MPAUSE 2000               /* Wait for a bit more*/
MWR 0x02015908 0x00000002 /* Unbypass the PLL */



/* Frequency Dependant Settings OPTIMIZED for 400MHz  CL=5 MT47H32M16HR-25EF */


/**********************************/
/* Put the DDR block in to reset */
/**********************************/
MWR 0x02018E7C 0x00000000

MWR 0x02018D08 0x00000000 /* POWERDOWN_EN                */
MWR 0x02018E44 0x00000002 /* BURST8_RDWR                 */
MWR 0x02018D10 0x00000004 /* RDWR_IDLE_GAP               */
MWR 0x02018D54 0x0000001F /* POWERDOWN_TO_X32            */
MWR 0x02018D98 0x00000001 /* REFRESH_TO_X32              */
MWR 0x02018DA0 0x00000000 /* 2T_DELAY                    */
MWR 0x02018DA8 0x00000003 /* FINAL_WAIT_X32              */

/***************************************************************************************************/
/* This block of DDR settings don't exist in the spreadsheet as they are not expected to change. */
/***************************************************************************************************/
MWR 0x02018E3C    0x02      /* CR_REG_DDRC_DLL_CALIB_TO_MIN_X1024             */
MWR 0x02018E40    0x02      /* CR_REG_DDRC_DLL_CALIB_TO_MAX_X1024             */
MWR 0x02018E84    0x97      /* CR_REG_DDRC_GO2CRITICAL_HYSTERESIS             */
MWR 0x02018F6C    0x02      /* CR_PHY_BL                                      */
MWR 0x02018D3C    0x01      /* CR_REG_DDRC_REFRESH_UPDATE_LEVEL               */

/****************************************************************************************************/
/* DDR registers from the settings spreadsheet - likely to change with frequency and board layout */
/****************************************************************************************************/
MWR 0x02018DE0    0x08      /* CR_REG_DDRC_ADDMAP_BANK_B0   */
MWR 0x02018DE4    0x08      /* CR_REG_DDRC_ADDMAP_BANK_B1   */
MWR 0x02018DE8    0x0F      /* CR_REG_DDRC_ADDMAP_BANK_B2      */
MWR 0x02018DEC    0x00      /* CR_REG_DDRC_ADDMAP_COL_B2    */
MWR 0x02018DF0    0x00      /* CR_REG_DDRC_ADDMAP_COL_B3    */
MWR 0x02018DF4    0x00      /* CR_REG_DDRC_ADDMAP_COL_B4_6  */
MWR 0x02018DF8    0x00      /* CR_REG_DDRC_ADDMAP_COL_B7    */
MWR 0x02018DFC    0x00      /* CR_REG_DDRC_ADDMAP_COL_B8    */
MWR 0x02018E00    0x00      /* CR_REG_DDRC_ADDMAP_COL_B9    */
MWR 0x02018E04    0x0F      /* CR_REG_DDRC_ADDMAP_COL_B10   */
MWR 0x02018E08    0x0F      /* CR_REG_DDRC_ADDMAP_COL_B11   */
MWR 0x02018E0C    0x03      /* CR_REG_DDRC_ADDMAP_ROW_B0    */
MWR 0x02018E10    0x03      /* CR_REG_DDRC_ADDMAP_ROW_B1    */
MWR 0x02018E14    0x03      /* CR_REG_DDRC_ADDMAP_ROW_B2_11 */
MWR 0x02018E18    0x03      /* CR_REG_DDRC_ADDMAP_ROW_B12   */
MWR 0x02018E1C    0x0F      /* CR_REG_DDRC_ADDMAP_ROW_B13   */
MWR 0x02018E20    0x0F      /* CR_REG_DDRC_ADDMAP_ROW_B14   */
MWR 0x02018E24    0x0F      /* CR_REG_DDRC_ADDMAP_ROW_B15   */

MWR 0x02018EAC    0x10      /* CR_REG_DDRC_PADS                               */
MWR 0x02018E34    0x05      /* CR_REG_DDRC_RD_ODT_HOLD                        */
MWR 0x02018E38    0x05      /* CR_REG_DDRC_WR_ODT_HOLD                        */
MWR 0x02018E94    0x0       /* CR_REG_DDRC_WR_ODT_DELAY                       */
MWR 0x02018E28   0x1      /* CR_REG_DDRC_RD_ODT_DELAY                        */
MWR 0x02018E2C   0x04     /* CR_REG_DDRC_RANK0_RD_ODT                        */
MWR 0x02018E30   0x03     /* CR_REG_DDRC_RANK0_WR_ODT                        */
MWR 0x02018F7C   0x02     /* CR_REG_LOCAL_RD_ODT                             */
MWR 0x02018F80   0x00     /* CR_REG_LOCAL_WR_ODT                             */
MWR 0x02018F84   0x00     /* CR_REG_LOCAL_IDLE_ODT                           */

MWR 0x02018DB0   0x4F     /* CR_REG_DDRC_PRE_CKE_X1024        */
MWR 0x02018D4C   0x61     /* CR_REG_DDRC_T_RFC_NOM_X32        */
MWR 0x02018D84   0x3      /* CR_REG_DDRC_T_CKE                */
MWR 0x02018D40   0x16     /* CR_REG_DDRC_T_RC                 */
MWR 0x02018D68   0x3      /* CR_REG_DDRC_WRITE_LATENCY        */
MWR 0x02018D58   0x1      /* CR_REG_DDRC_T_FAW                */
MWR 0x02018D5C   0x1B     /* CR_REG_DDRC_T_RAS_MAX            */
MWR 0x02018D48   0x7      /* CR_REG_DDRC_POST_SELFREF_GAP_X32 */
MWR 0x02018D7C   0x3      /* CR_REG_DDRC_RD2PRE               */
MWR 0x02018E88   0x2      /* CR_REG_DDRC_T_MRD                */
MWR 0x02018D60   0x10     /* CR_REG_DDRC_T_RAS_MIN            */
MWR 0x02018D94   0x6      /* CR_REG_DDRC_T_RP                 */
MWR 0x02018D88   0x1      /* CR_REG_DDRC_T_CCD                */
MWR 0x02018F60   0x4      /* CR_PHY_FIRST_WR                  */
MWR 0x02018D50   0xC      /* CR_REG_DDRC_WR2PRE               */
MWR 0x02018DB4   0x1      /* CR_REG_DDRC_POST_CKE_X1024       */
MWR 0x02018D70   0x9      /* CR_REG_DDRC_WR2RD                */
MWR 0x02018D64   0x5      /* CR_REG_DDRC_READ_LATENCY         */
MWR 0x02018F64   0x5      /* CR_PHY_FIRST_RD                  */
MWR 0x02018D44   0x2A     /* CR_REG_DDRC_T_RFC_MIN            */
MWR 0x02018D8C   0x4      /* CR_REG_DDRC_T_RRD                */
MWR 0x02018D6C   0x5      /* CR_REG_DDRC_RD2WR                */
MWR 0x02018D74   0x2      /* CR_REG_DDRC_T_XP                 */
MWR 0x02018D80   0x5      /* CR_REG_DDRC_T_RCD                */

MWR 0x02018DB8 0x00000B52 /* CR_REG_DDRC_MR                   */
MWR 0x02018DBC 0x00002042 /* CR_REG_DDRC_EMR                  */

MWR 0x02018E8C 0x00004000 /* EMR2 Reg                   */
MWR 0x02018E90 0x00006000 /* EMR3 Reg                   */


MWR 0x02018F94 0x00000044 /* CR_PHY_WR_SLAVE_RATIO0 */
MWR 0x02018F98 0x00000038 /* CR_PHY_RD_SLAVE_RATIO0 */
MWR 0x02018F9C 0x0000003A /* CR_PHY_RD_SLAVE_RATIO1 */
MWR 0x02018FA0 0x00000044 /* CR_PHY_WR_SLAVE_RATIO1 */

/************************************/
/* Bring the DDR block out of reset */
/************************************/
MWR 0x02018E7C 0x00000001

/* MC REQ */
MWR 0x02018C10 0x00000000
MWR 0x02018C14 0xFFFFFFFF
MWR 0x02018C18 0x00000000
MWR 0x02018C1C 0x00000000

/* Turn off circular buffers  - DVB-H Link-Layer circular buffering registers (documented in the UCCP320
TRM).  They are on by default for backward compatibility reasons, but will default to off for all except
DVBH- RS/TS Demux from UCCP330 */
MWR 0x0200061C 0x00000000
MWR 0x0200065C 0x00000000
MWR 0x0200066C 0x00000000
MWR 0x0200068C 0x00000000
MWR 0x0200069C 0x00000000
MWR 0x020006AC 0x00000000
MWR 0x020006BC 0x00000000
MWR 0x020006CC 0x00000000
MWR 0x020006DC 0x00000000
MWR 0x020006EC 0x00000000

MPAUSE 7000

/* Timer Clock Configuration - Must be set to (core clock rate - 1) */
MWR 0x04830140 399 /* TIMERCONFIG */
```