# S5PV210 时钟体系

S5PV210 中包含 3 大类时钟 domain，分别是主系统时钟 domain (简称 MSYS，下面

将

使用简称来进行相关讲解)、显示相关的时钟 domain (DSYS)、外围设备的时钟 domain

(PSYS)。

1） MSYS：用来给 cortex a8 处理器，dram 控制器，3D，IRAM，IROM，中断控制器等提

供时钟；

2) DSYS：用来给显示相关的部件提供时钟，包括 FIMC, FIMD, JPEG, and multimedia IPs；

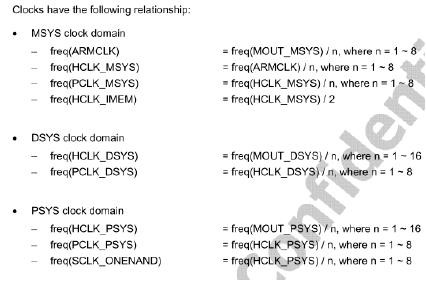
3) PSYS:用来给外围设备提供时钟，如 i2s, spi,i2c,uart 等

实验板外接的晶振频率(简称 Fin)为 24MHz，通过时钟控制逻辑 PLL 可以提高系统

时钟。S5PV210 共有 4 个倍频器，即PLL，包括APLL(供 MSYS 使用),MPLL(供DSYS

使用),EPLL(供 PSYS使用),VPLL(供 video 相关的时钟使用)。3大类时钟domain中，可以

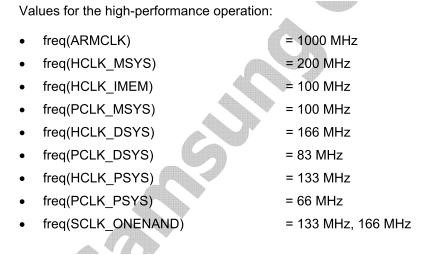
使用不同的分频，使其给不同部件输出所需要的时钟，各类时钟的关系如下图：



S5PV210 时钟分类图

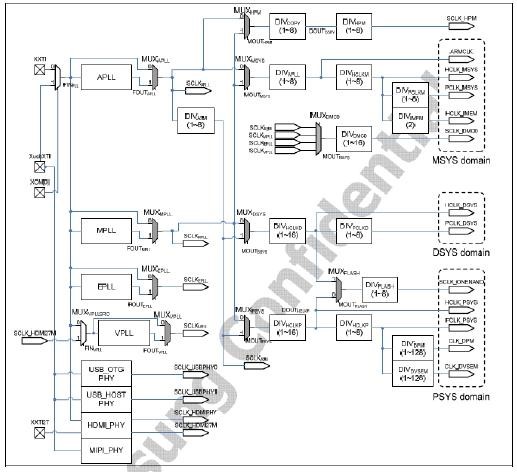
如何确定各类时钟的值，芯片手册上给出了参考值，我们按照参考值设置即可，各类时钟的

参考值如下图：



S5PV210 时钟设置参考值图

具体如何设置上述各种各样的时钟，可参考下图(芯片手册 P361)：



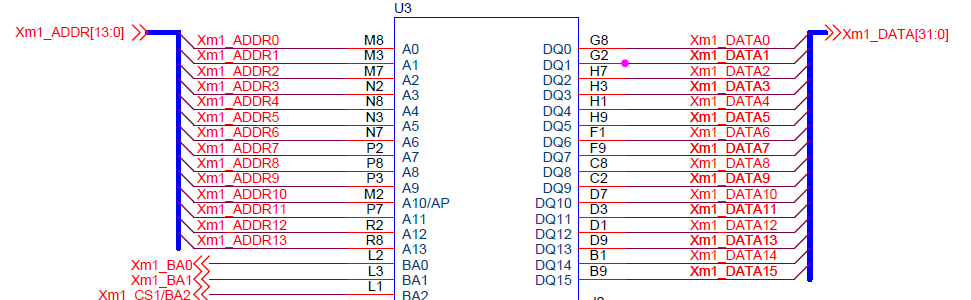
S5PV210 时钟设置参考图

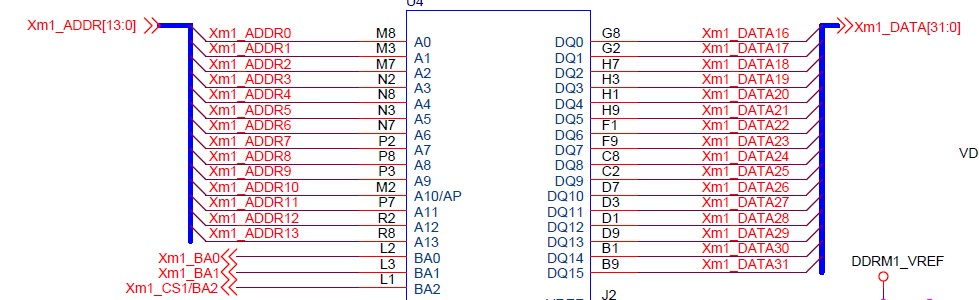
# 内存初始化

SDRAM基础：

通过s3c2440的内存原理以及时序来理解s5pv210 SDRAM原理、时序。

首先看一下核心板内存如何连接的





 可以看一下两个内存芯片接的地址总线均为Xm1\_ADDR[13:0]，数据总线Xm1\_DATA[15:0]、Xm1\_DATA[31:16]，两个内存芯片是并联的，当地址总线Xm1\_ADDR[13:0]寻址时，

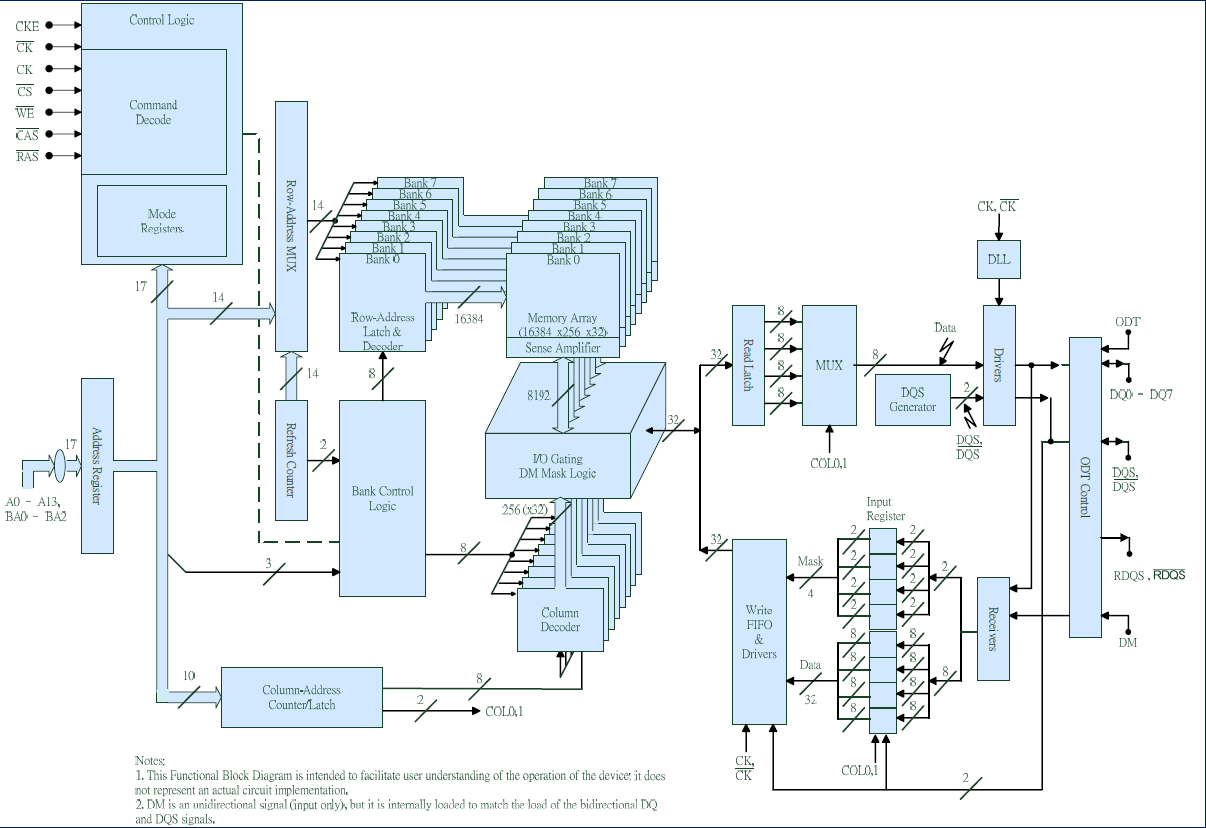
可以同时在两个内存芯片上各获取16位数据来组成一个32位数据，并由32位数据总线输出。

在看下面这幅图：下图为每个内存芯片内部框图：Block Diagram (128Mb x 8)，从图中可以看出：有8bank：bank0—bank1，BA0-BA2用来选择bank的，正好可选8个。

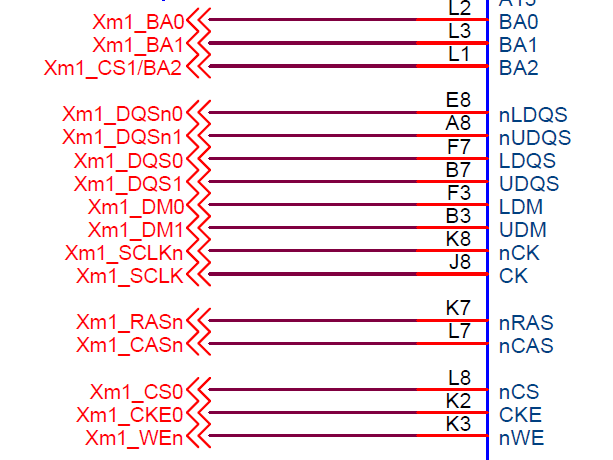
每个bank的row寻址为14位，列寻址的位10位(行列寻址是共用14位地址总线的，通过不同的时序来加以区分，下面会详细介绍时序)，所以每个bank的大小为2的24次方bit，为16MB，

一共8个bank，总共128MB。从核心板图上可以看出整个SDRAM，由4对16位内存芯片组成，每个内存芯片128MB，其中有两个内存芯片并联成32位内存，共256MB内存连接在DMC0，

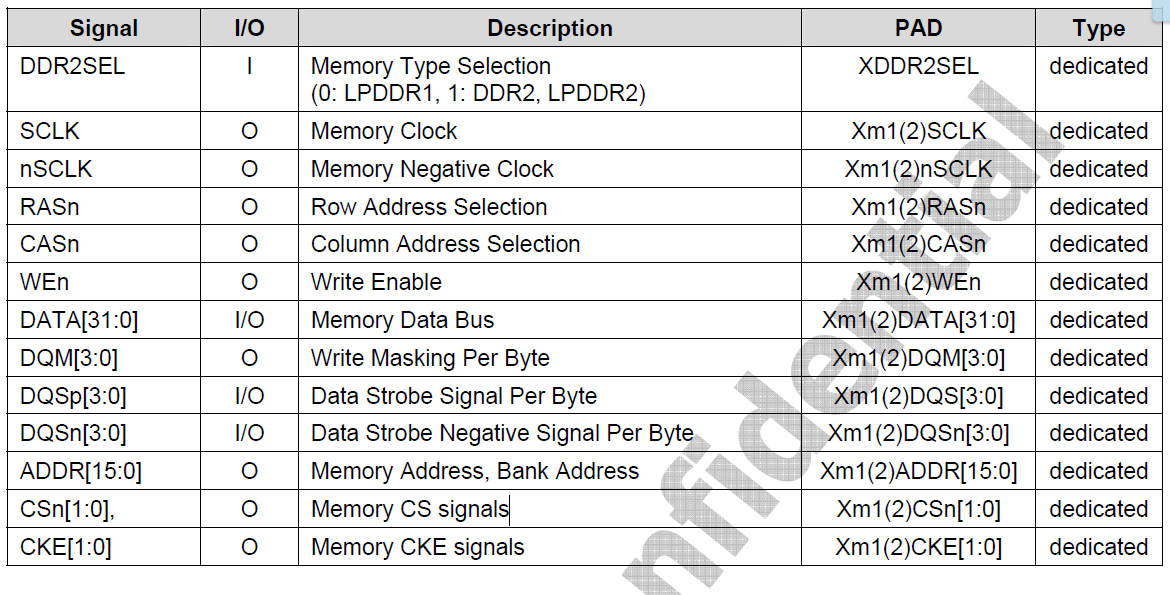
另两个内存芯片也并联成32位内存，共256MB连接在DMC1。



 2：内存芯片各引脚说明



除去上面介绍的Xm1\_ADDR[13:0]做行列地址总线、Xm1\_DATA[31:0]作为地址总线以外还有以下接线：



Xm1\_BA[2:0]：bank address，bank选择引脚；

Xm1\_SCLK：内存时钟；

Xm1\_SCLKn：负时钟

Xm1\_RASn：行地址选择；

Xm1\_CASn：列地址选择；

CS：片选；

CKE：命令有效时钟；

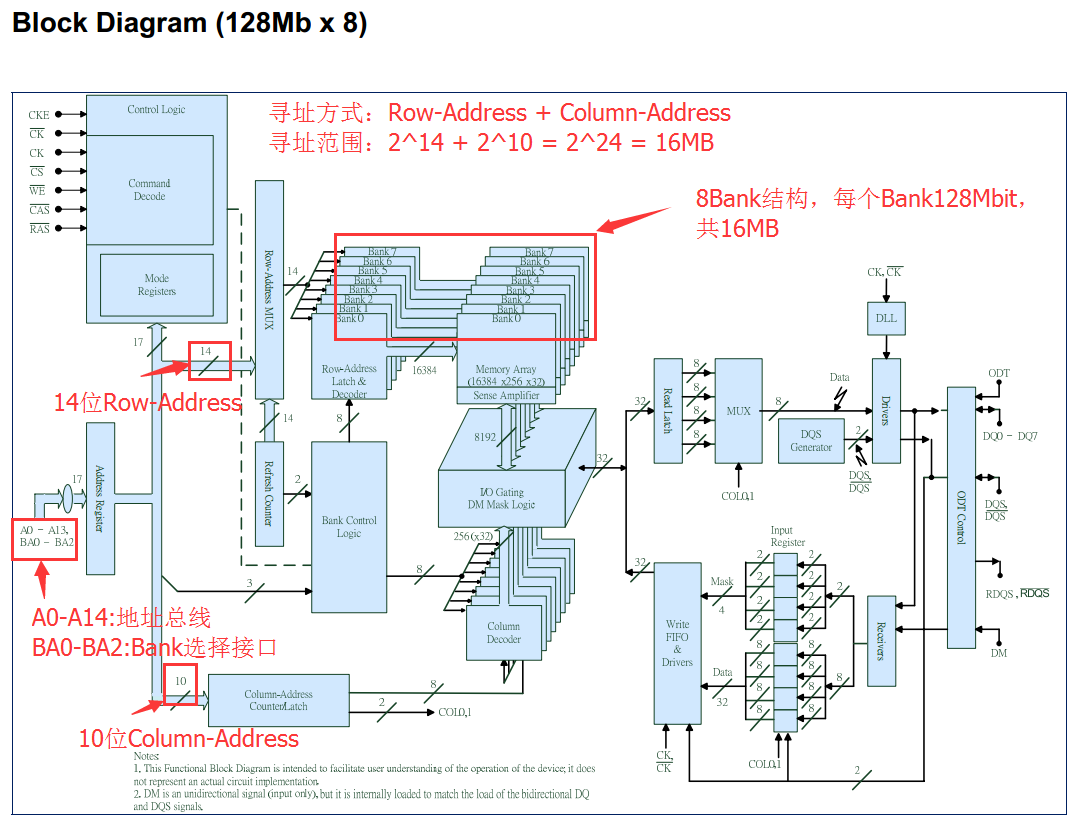
WE：写信号；

DQS：数据选通信号；

DQSn：数据选通负信号

内存详情

## https://img-blog.csdn.net/20160925022420563内存寻址详解



采用128Mb \* 8 结构，每个Bank可寻址的大小是16MB，BA0-BA2用来选择8个Bank，寻址的方式：Row-Address+Column-Address。

故单内存芯片可寻址的大小为：16MB \* 8 = 128MB

两片单芯片内存并联组合成32位数据总线，可与SoC进行总线通信，可寻址的内存大小便为：128MB+128MB= 256MB，在九鼎的开发板上采用了DRAM0+DRAM1分配内存地址，所以DRAM0和DRAM1都是256MB内存大小，合起来就是512MB内存大小。

所以开发板上DRAM0和DRAM1可寻址的有效范围分别为：

DRAM0：0x20000000 - 0x2FFFFFFF (256MB)

DRAM1：0x40000000 - 0x4FFFFFFF (256MB)

其他地址为非法地址，比如：0x30000000

## 4、SDRAM初始化详解

SDRAM初始化共涉及到两个文件：s5pv21.h 和 sdram\_init.s

三星在官方的数据手册里（section 05 memory->DRAM CONTROLLER->1.2.1.3章节中）讲到DDR2的初始化过程，共27步，如下

1. 提供稳压电源给内存控制器和内存芯片，内存控制器必须保持CLE在低电平，此时就会提供稳压电源。注：当CKE引脚为低电平时，XDDR2SEL应该处于高电平

2. 依照时钟频率正确配置PhyControl0.ctrl\_start\_point和PhyControl0.ctrl\_incbit-fields的值。配置的PhyControl0.ctrl\_dll\_on值为'1'以打开PHY

DLL。

3. DQS Cleaning：依照时钟频率和内存的tAC参数正确设置PhyControl1.ctrl\_shiftc and PhyControl1.ctrl\_offsetcbit-fields位的值。

4. 配置PhyControl0.ctrl\_start位的值为'1'

5. 配置ConControl，与此同时，auto refresh自动刷新计数器应该关闭

6. 配置MemControl，与此同时，所有的power down（休眠模式）应该闭关

7. 配置MemConfig0寄存器。如果有两组内存芯片(比如有8片DDR，这8片DDR是分别挂在Memory Port1和Memory Port2上)，再配置MemConfig1寄存器。

8. 配置PrechConfig和PwrdnConfig寄存器

9. 依照内存的tAC参数配置TimingAref, TimingRow, TimingData和TimingPower寄存器

10. 如果需要QoS标准，配置QosControl0~15和QosConfig0~15r寄存器

11. 等待PhyStatus0.ctrl\_locked位变为'1'。检查是否PHY DLL是否已锁

12. PHY DLL补偿在内存操作时由PVT(Process, Voltage and Temperature，处理器、电压和温度)变化引起的延迟量。但是，PHY DLL不能因某些可靠的内存操作而切断，除非是工作在低频率下。如果关闭PHY DLL，依照PhyStatus0.ctrl\_lock\_value[9:2]位的值正确配置PhyControl0.ctrl\_force位的值来弥补延迟量(fix

delay amount)。清除PhyControl0.ctrl\_dll\_on位的值来关闭PHY DLL。

13. 上电后，确定最小值为200us的稳定时钟是否发出

14. 使用DirectCmd寄存器发出一个NOP命令，保证CKE引脚为高电平

15. 等最小400ns

16. 使用DirectCmd寄存器发出一个PALL命令

17. 使用DirectCmd寄存器发出一个EMRS2命令，program操作参数

18. 使用DirectCmd寄存器发出一个EMRS3命令，program操作参数

19. 使用DirectCmd寄存器发出一个EMRS命令来使能内存DLLs

20. 使用DirectCmd寄存器发出一个MRS命令，重启内存DLL

21. 使用DirectCmd寄存器发出一个PALL命令

22. 使用DirectCmd寄存器发出两个Auto

Refresh(自动刷新)命令

23. 使用DirectCmd寄存器发出一个MRS命令，program操作参数，不要重启内存DLL

24. 等待最小200时钟周期

25. 使用DirectCmd寄存器发出一个EMRS命令给程序的运行参数。如果OCD校正(Off-Chip

Driver，片外驱动调校)没有使用，改善一个EMRS命令去设置OCD校准的默认值。在此之后，发送一个EMRS指令去退出OCD校准模式，继续program操作参数

26. 如果有两组DDR芯片，重复14-25步配置chip1的内存，刚刚配置的是chip0，也就是第一组内存芯片

27. 配置ConControlto来打开自动刷新计数器

28. 如果需要power down(休眠)模式，配置MemControl寄存器.

S5pv21.h ：定义了SDRAM中所有的寄存器地址

sdram\_init.s：定义了SDRAM中初始化的过程，此过程完全参考三星原厂要求的这27个步骤

在初始化过程中涉及到的寄存器分别在：

section 02 system->GENERAL PURPOSE INPUT/OUTPUT->2.2章节

section 05 memory->DRAM CONTROLLER->1.4章节

注意：在源码sdram\_init.s中，朱老师修改了DMC0\_MEMCONTROL和DMC0\_MEMCONFIG\_0两个头文件的值，分别为：

// MemControl BL=4, 1Chip, DDR2 Type, dynamic self refresh, force precharge, dynamic power down off

#define DMC0\_MEMCONTROL 0x00202400

// MemConfig0 256MB config, 8 banks,Mapping Method[12:15]0:linear, 1:linterleaved, 2:Mixed

#define DMC0\_MEMCONFIG\_0 0x20F01323 1234

没修改前的值：

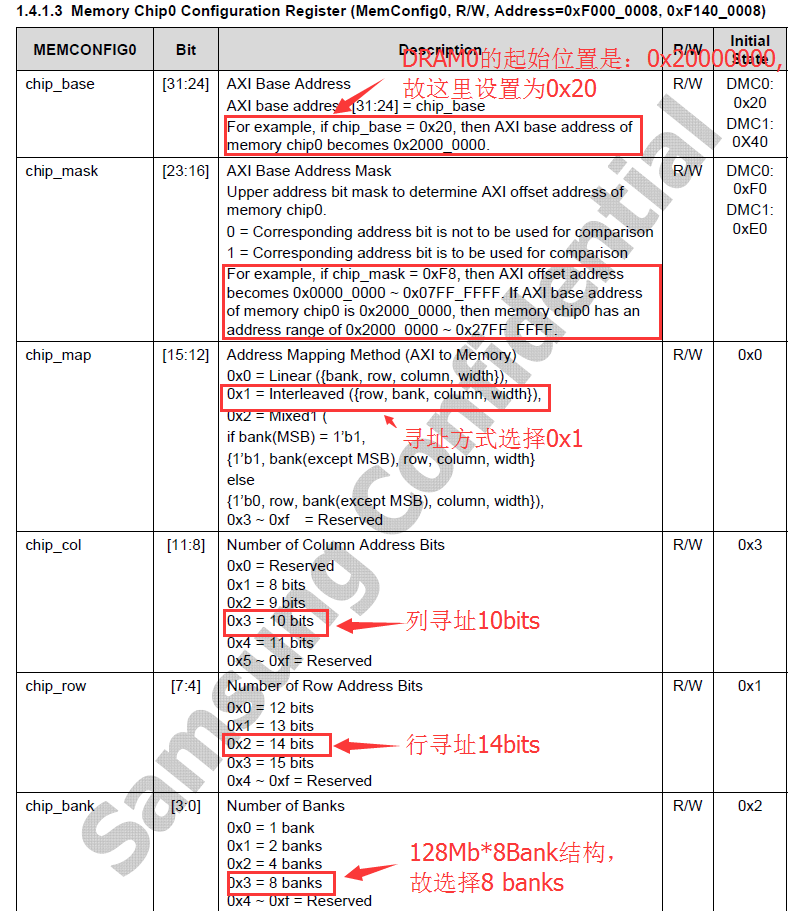
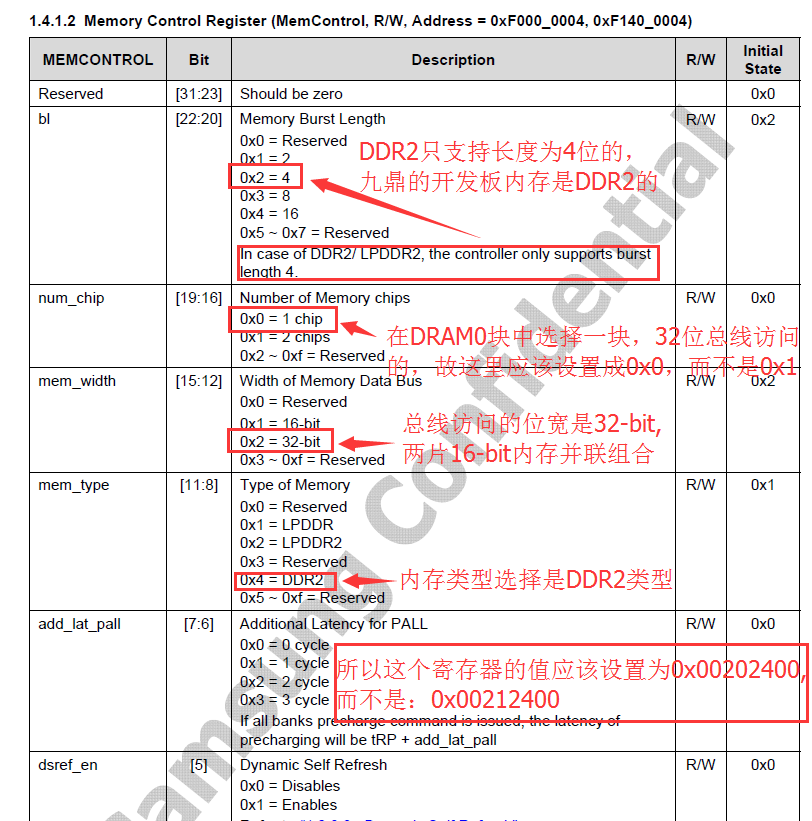
// MemControl BL=4, 1Chip, DDR2 Type, dynamic self refresh, force precharge, dynamic power down off

#define DMC0\_MEMCONTROL 0x00212400

// MemConfig0 512MB config, 8 banks,Mapping Method[12:15]0:linear, 1:linterleaved, 2:Mixed

#define DMC0\_MEMCONFIG\_0 0x20E01323 1234

修改的原因：在官方的数据手册中DMC\_MEMCONTROL 寄存器中应该如下图设置



在MemConfig0寄存器中的chip\_mask选项中，因为反码的原因，我们选择F0，二进制位0b11110000，代表的意义是前面4个1，后面4个0，反码之后变为0F，二进制位0b00001111，代表的意义是前面4个0，后面4个1。因为我们的DRAM0起始地址为0x2000\_0000，故芯片掩码过后是0x2FFF\_FFFF，共256M。如果设置成E0，则二进制位11100000，反码后为00011111，掩码过后为0x3FFF\_FFFF，共512M。与开发板设置的大小不符，开发板上选择了2片16位数据总线的128MB内存芯片组合成32位数据总线的256MB内存芯片，所以不能设置成E0，而应该设置成F0.

其他的设置都没什么变化，因为没用到DRAM0中的chip1,所以DRAM0中的DMC0\_MEMCONFIG\_1等寄存器设置采用默认值或者修改没什么影响。

只要DRAM0中的chip0按照三星官方数据手册中的那27步设置就可以了。

在链接规则文件中link.ldr，把其实地址设置为0x20000000, 代码如下

SECTIONS

{

. = 0x20000000;

.text : {

start.o

sdram\_init.o

\* (.text)

}

.data : {

\* (.data)

}

bss\_start = .;

.bss : {

\* (.bss)

}

bss\_end = .;

}

# 附录

初始化代码

Start.s:

#define WTCON 0xE2700000

#define SVC\_STACK 0xD0037D80

.global \_start // 把\_start链接属性改为外部，这样其他文件就可以看见\_start了

\_start:

//关看门狗（向WTCON的bit5写入0即可）

ldr r0,= WTCON

ldr r1,= (0<<5)

str r1,[r0]

//设置SVC栈，实现汇编与C的相互调用

ldr sp,= SVC\_STACK

//开关icache

mrc p15,0,r0,c1,c0,0 //读出cp15的c1到r0中

//bic r0, r0, #(1<<12) //bit12 清0 关icache

orr r0, r0, #(1<<12) //bit12 置1 开icache

mcr p15,0,r0,c1,c0,0 //将r0写入cp15中的c1中

//初始化ddr

bl sdram\_asm\_init //bl跳转，mov pc,=lr回来

//重定位

// adr指令用于加载\_start当前运行地址(即dnw下载时选择的地址(0xd0020010))

adr r0, \_start // adr加载时就叫短加载

// ldr指令用于加载\_start的链接地址:0x20000000

ldr r1, =\_start // ldr加载时如果目标寄存器是pc就叫长跳转，如果目标寄存器是r1等就叫长加载

// bss段的起始地址

ldr r2, =bss\_start // 就是我们重定位代码的结束地址，重定位只需重定位代码段和数据段即可

cmp r0, r1 // 比较\_start的运行时地址和链接地址是否相等

beq clean\_bss // 如果相等说明不需要重定位，所以跳过copy\_loop，直接到clean\_bss

// 如果不相等说明需要重定位，那么直接执行下面的copy\_loop进行重定位

// 重定位完成后继续执行clean\_bss。

// 用汇编来实现的一个while循环

copy\_loop:

ldr r3, [r0], #4 // 源 后面的#4就是：r0的四个字节赋值给r3，r0地址加4

str r3, [r1], #4 // 目的 这两句代码就完成了4个字节内容的拷贝

cmp r1, r2 // r1和r2都是用ldr加载的，都是链接地址，所以r1不断+4总能等于r2

bne copy\_loop

// 清bss段，其实就是在链接地址处把bss段全部清零

clean\_bss:

ldr r0, =bss\_start

ldr r1, =bss\_end

cmp r0, r1 // 如果r0等于r1，说明bss段为空，直接下去

beq run\_on\_dram // 清除bss完之后的地址

mov r2, #0 // r2 = 0

clear\_loop:

str r2, [r0], #4 // 先将r2中的值放入r0所指向的内存地址（r0中的值作为内存地址），

cmp r0, r1 // 然后r0 = r0 + 4(即将bss清0)

bne clear\_loop

run\_on\_dram:

// 长跳转到led\_blink开始第二阶段

ldr pc, =czg\_led // ldr指令实现长跳转，b指令是短跳转

//最后这里跳转，是跳转到重定位的代码所对应的czg\_led这个函数那里去执行的

link.lds:

SECTIONS

{. = 0x20000000; /\*指定链接地址为0x20000000\*/.text : { /\*代码段\*/start.o /\*指定链接的顺序为：start.o->sdram\_init.o->其他的一些文件\*/\* (.text) /\*这里表示其他的一些.o文件\*/}

.data : { /\*数据段\*/\* (.data) /\*这里表示所有的数据段的文件\*/}bss\_start = .; /\*把当前的地址赋值给bss\_start\*/.bss : { /\*bss段\*/\* (.bss) /\*所有bss段的文件\*/}bss\_end = .;

/\*把当前的地址赋值给bss\_end\*/

Makefile:

led.bin: start.o led.o sdram\_init.oarm-linux-ld -Tlink.lds -o led.elf $^#代码段运行地址为0x0，将所有依赖文件链接为led.elfarm-linux-objcopy -O binary led.elf led.bin #将led.elf复制一份为led.bin文件arm-linux-objdump -D led.elf > led\_elf.dis#将led.elf文件转换为.dis反汇编文件gcc mkv210\_image.c -o mkx210./mkx210 led.bin 210.bin

%.o : %.Sarm-linux-gcc -o $@ $< -c -nostdlib

%.o : %.carm-linux-gcc -o $@ $< -c -nostdlib

clean:rm \*.o \*.elf \*.bin \*.dis mkx210 -f

sdram\_init..S:

#include "s5pv210.h"

#if 1

#define DMC0\_MEMCONTROL 0x00202400 // MemControl BL=4, 1Chip, DDR2 Type, dynamic self refresh, force precharge, dynamic power down off

#define DMC0\_MEMCONFIG\_0 0x20F01323 // MemConfig0 256MB config, 8 banks,Mapping Method[12:15]0:linear, 1:linterleaved, 2:Mixed

#define DMC0\_MEMCONFIG\_1 0x30F00312 // MemConfig1 默认值

#define DMC0\_TIMINGA\_REF 0x00000618 // TimingAref 7.8us\*133MHz=1038(0x40E), 100MHz=780(0x30C), 20MHz=156(0x9C), 10MHz=78(0x4E)

#define DMC0\_TIMING\_ROW 0x28233287 // TimingRow for @200MHz

#define DMC0\_TIMING\_DATA 0x23240304 // TimingData CL=3

#define DMC0\_TIMING\_PWR 0x09C80232 // TimingPower

#define DMC1\_MEMCONTROL 0x00202400 // MemControl BL=4, 2 chip, DDR2 type, dynamic self refresh, force precharge, dynamic power down off

#define DMC1\_MEMCONFIG\_0 0x40F01323 // MemConfig0 512MB config, 8 banks,Mapping Method[12:15]0:linear, 1:linterleaved, 2:Mixed

#define DMC1\_MEMCONFIG\_1 0x60E00312 // MemConfig1

#define DMC1\_TIMINGA\_REF 0x00000618 // TimingAref 7.8us\*133MHz=1038(0x40E), 100MHz=780(0x30C), 20MHz=156(0x9C), 10MHz=78(0x4

#define DMC1\_TIMING\_ROW 0x28233289 // TimingRow for @200MHz

#define DMC1\_TIMING\_DATA 0x23240304 // TimingData CL=3

#define DMC1\_TIMING\_PWR 0x08280232 // TimingPower

#endif

#if 0

#define DMC0\_MEMCONTROL 0x00212400 // MemControl BL=4, 1Chip, DDR2 Type, dynamic self refresh, force precharge, dynamic power down off

#define DMC0\_MEMCONFIG\_0 0x20E01323 // MemConfig0 512MB config, 8 banks,Mapping Method[12:15]0:linear, 1:linterleaved, 2:Mixed

#define DMC0\_MEMCONFIG\_1 0x40F01323 // MemConfig1

#define DMC0\_TIMINGA\_REF 0x00000618 // TimingAref 7.8us\*133MHz=1038(0x40E), 100MHz=780(0x30C), 20MHz=156(0x9C), 10MHz=78(0x4E)

#define DMC0\_TIMING\_ROW 0x28233287 // TimingRow for @200MHz

#define DMC0\_TIMING\_DATA 0x23240304 // TimingData CL=3

#define DMC0\_TIMING\_PWR 0x09C80232 // TimingPower

#define DMC1\_MEMCONTROL 0x00202400 // MemControl BL=4, 2 chip, DDR2 type, dynamic self refresh, force precharge, dynamic power down off

#define DMC1\_MEMCONFIG\_0 0x40C01323 // MemConfig0 512MB config, 8 banks,Mapping Method[12:15]0:linear, 1:linterleaved, 2:Mixed

#define DMC1\_MEMCONFIG\_1 0x00E01323 // MemConfig1

#define DMC1\_TIMINGA\_REF 0x00000618 // TimingAref 7.8us\*133MHz=1038(0x40E), 100MHz=780(0x30C), 20MHz=156(0x9C), 10MHz=78(0x4

#define DMC1\_TIMING\_ROW 0x28233289 // TimingRow for @200MHz

#define DMC1\_TIMING\_DATA 0x23240304 // TimingData CL=3

#define DMC1\_TIMING\_PWR 0x08280232 // TimingPower

#endif

.global sdram\_asm\_init

sdram\_asm\_init:

ldr r0, =0xf1e00000

ldr r1, =0x0

str r1, [r0, #0x0]

/\* DMC0 Drive Strength (Setting 2X) \*/

ldr r0, =ELFIN\_GPIO\_BASE

ldr r1, =0x0000AAAA

str r1, [r0, #MP1\_0DRV\_SR\_OFFSET] // 寄存器中对应0b10，就是2X

ldr r1, =0x0000AAAA

str r1, [r0, #MP1\_1DRV\_SR\_OFFSET]

ldr r1, =0x0000AAAA

str r1, [r0, #MP1\_2DRV\_SR\_OFFSET]

ldr r1, =0x0000AAAA

str r1, [r0, #MP1\_3DRV\_SR\_OFFSET]

ldr r1, =0x0000AAAA

str r1, [r0, #MP1\_4DRV\_SR\_OFFSET]

ldr r1, =0x0000AAAA

str r1, [r0, #MP1\_5DRV\_SR\_OFFSET]

ldr r1, =0x0000AAAA

str r1, [r0, #MP1\_6DRV\_SR\_OFFSET]

ldr r1, =0x0000AAAA

str r1, [r0, #MP1\_7DRV\_SR\_OFFSET]

ldr r1, =0x00002AAA

str r1, [r0, #MP1\_8DRV\_SR\_OFFSET]

/\* DMC1 Drive Strength (Setting 2X) \*/

ldr r0, =ELFIN\_GPIO\_BASE

ldr r1, =0x0000AAAA

str r1, [r0, #MP2\_0DRV\_SR\_OFFSET]

ldr r1, =0x0000AAAA

str r1, [r0, #MP2\_1DRV\_SR\_OFFSET]

ldr r1, =0x0000AAAA

str r1, [r0, #MP2\_2DRV\_SR\_OFFSET]

ldr r1, =0x0000AAAA

str r1, [r0, #MP2\_3DRV\_SR\_OFFSET]

ldr r1, =0x0000AAAA

str r1, [r0, #MP2\_4DRV\_SR\_OFFSET]

ldr r1, =0x0000AAAA

str r1, [r0, #MP2\_5DRV\_SR\_OFFSET]

ldr r1, =0x0000AAAA

str r1, [r0, #MP2\_6DRV\_SR\_OFFSET]

ldr r1, =0x0000AAAA

str r1, [r0, #MP2\_7DRV\_SR\_OFFSET]

ldr r1, =0x00002AAA

str r1, [r0, #MP2\_8DRV\_SR\_OFFSET]

/\* DMC0 initialization at single Type\*/

ldr r0, =APB\_DMC\_0\_BASE

ldr r1, =0x00101000 @PhyControl0 DLL parameter setting, manual 0x00101000

str r1, [r0, #DMC\_PHYCONTROL0]

ldr r1, =0x00000086 @PhyControl1 DLL parameter setting, LPDDR/LPDDR2 Case

str r1, [r0, #DMC\_PHYCONTROL1]

ldr r1, =0x00101002 @PhyControl0 DLL on

str r1, [r0, #DMC\_PHYCONTROL0]

ldr r1, =0x00101003 @PhyControl0 DLL start

str r1, [r0, #DMC\_PHYCONTROL0]

find\_lock\_val:

ldr r1, [r0, #DMC\_PHYSTATUS] @Load Phystatus register value

and r2, r1, #0x7

cmp r2, #0x7 @Loop until DLL is locked

bne find\_lock\_val

and r1, #0x3fc0

mov r2, r1, LSL #18

orr r2, r2, #0x100000

orr r2 ,r2, #0x1000

orr r1, r2, #0x3 @Force Value locking

str r1, [r0, #DMC\_PHYCONTROL0]

#if 0 /\* Memory margin test 10.01.05 \*/

orr r1, r2, #0x1 @DLL off

str r1, [r0, #DMC\_PHYCONTROL0]

#endif

/\* setting DDR2 \*/

ldr r1, =0x0FFF2010 @ConControl auto refresh off

str r1, [r0, #DMC\_CONCONTROL]

ldr r1, =DMC0\_MEMCONTROL @MemControl BL=4, 1 chip, DDR2 type, dynamic self refresh, force precharge, dynamic power down off

str r1, [r0, #DMC\_MEMCONTROL]

ldr r1, =DMC0\_MEMCONFIG\_0 @MemConfig0 256MB config, 8 banks,Mapping Method[12:15]0:linear, 1:linterleaved, 2:Mixed

str r1, [r0, #DMC\_MEMCONFIG0]

ldr r1, =DMC0\_MEMCONFIG\_1 @MemConfig1

str r1, [r0, #DMC\_MEMCONFIG1]

ldr r1, =0xFF000000 @PrechConfig

str r1, [r0, #DMC\_PRECHCONFIG]

ldr r1, =DMC0\_TIMINGA\_REF @TimingAref 7.8us\*133MHz=1038(0x40E), 100MHz=780(0x30C), 20MHz=156(0x9C), 10MHz=78(0x4E)

str r1, [r0, #DMC\_TIMINGAREF]

ldr r1, =DMC0\_TIMING\_ROW @TimingRow for @200MHz

str r1, [r0, #DMC\_TIMINGROW]

ldr r1, =DMC0\_TIMING\_DATA @TimingData CL=3

str r1, [r0, #DMC\_TIMINGDATA]

ldr r1, =DMC0\_TIMING\_PWR @TimingPower

str r1, [r0, #DMC\_TIMINGPOWER]

ldr r1, =0x07000000 @DirectCmd chip0 Deselect

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x01000000 @DirectCmd chip0 PALL

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00020000 @DirectCmd chip0 EMRS2

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00030000 @DirectCmd chip0 EMRS3

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00010400 @DirectCmd chip0 EMRS1 (MEM DLL on, DQS# disable)

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00000542 @DirectCmd chip0 MRS (MEM DLL reset) CL=4, BL=4

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x01000000 @DirectCmd chip0 PALL

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x05000000 @DirectCmd chip0 REFA

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x05000000 @DirectCmd chip0 REFA

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00000442 @DirectCmd chip0 MRS (MEM DLL unreset)

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00010780 @DirectCmd chip0 EMRS1 (OCD default)

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00010400 @DirectCmd chip0 EMRS1 (OCD exit)

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x07100000 @DirectCmd chip1 Deselect

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x01100000 @DirectCmd chip1 PALL

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00120000 @DirectCmd chip1 EMRS2

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00130000 @DirectCmd chip1 EMRS3

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00110400 @DirectCmd chip1 EMRS1 (MEM DLL on, DQS# disable)

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00100542 @DirectCmd chip1 MRS (MEM DLL reset) CL=4, BL=4

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x01100000 @DirectCmd chip1 PALL

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x05100000 @DirectCmd chip1 REFA

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x05100000 @DirectCmd chip1 REFA

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00100442 @DirectCmd chip1 MRS (MEM DLL unreset)

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00110780 @DirectCmd chip1 EMRS1 (OCD default)

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00110400 @DirectCmd chip1 EMRS1 (OCD exit)

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x0FF02030 @ConControl auto refresh on

str r1, [r0, #DMC\_CONCONTROL]

ldr r1, =0xFFFF00FF @PwrdnConfig

str r1, [r0, #DMC\_PWRDNCONFIG]

ldr r1, =0x00202400 @MemControl BL=4, 2 chip, DDR2 type, dynamic self refresh, force precharge, dynamic power down off

str r1, [r0, #DMC\_MEMCONTROL]

// 上面是DRAM0初始化步骤

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// 下面是DRAM1初始化步骤，两者没有联系，是并列的。

/\* DMC1 initialization \*/

ldr r0, =APB\_DMC\_1\_BASE

ldr r1, =0x00101000 @Phycontrol0 DLL parameter setting

str r1, [r0, #DMC\_PHYCONTROL0]

ldr r1, =0x00000086 @Phycontrol1 DLL parameter setting

str r1, [r0, #DMC\_PHYCONTROL1]

ldr r1, =0x00101002 @PhyControl0 DLL on

str r1, [r0, #DMC\_PHYCONTROL0]

ldr r1, =0x00101003 @PhyControl0 DLL start

str r1, [r0, #DMC\_PHYCONTROL0]

find\_lock\_val1:

ldr r1, [r0, #DMC\_PHYSTATUS] @Load Phystatus register value

and r2, r1, #0x7

cmp r2, #0x7 @Loop until DLL is locked

bne find\_lock\_val1

and r1, #0x3fc0

mov r2, r1, LSL #18

orr r2, r2, #0x100000

orr r2, r2, #0x1000

orr r1, r2, #0x3 @Force Value locking

str r1, [r0, #DMC\_PHYCONTROL0]

#if 0 /\* Memory margin test 10.01.05 \*/

orr r1, r2, #0x1 @DLL off

str r1, [r0, #DMC\_PHYCONTROL0]

#endif

/\* settinf fot DDR2 \*/

ldr r0, =APB\_DMC\_1\_BASE

ldr r1, =0x0FFF2010 @auto refresh off

str r1, [r0, #DMC\_CONCONTROL]

ldr r1, =DMC1\_MEMCONTROL @MemControl BL=4, 2 chip, DDR2 type, dynamic self refresh, force precharge, dynamic power down off

str r1, [r0, #DMC\_MEMCONTROL]

ldr r1, =DMC1\_MEMCONFIG\_0 @MemConfig0 512MB config, 8 banks,Mapping Method[12:15]0:linear, 1:linterleaved, 2:Mixed

str r1, [r0, #DMC\_MEMCONFIG0]

ldr r1, =DMC1\_MEMCONFIG\_1 @MemConfig1

str r1, [r0, #DMC\_MEMCONFIG1]

ldr r1, =0xFF000000

str r1, [r0, #DMC\_PRECHCONFIG]

ldr r1, =DMC1\_TIMINGA\_REF @TimingAref 7.8us\*133MHz=1038(0x40E), 100MHz=780(0x30C), 20MHz=156(0x9C), 10MHz=78(0x4

str r1, [r0, #DMC\_TIMINGAREF]

ldr r1, =DMC1\_TIMING\_ROW @TimingRow for @200MHz

str r1, [r0, #DMC\_TIMINGROW]

ldr r1, =DMC1\_TIMING\_DATA @TimingData CL=3

str r1, [r0, #DMC\_TIMINGDATA]

ldr r1, =DMC1\_TIMING\_PWR @TimingPower

str r1, [r0, #DMC\_TIMINGPOWER]

ldr r1, =0x07000000 @DirectCmd chip0 Deselect

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x01000000 @DirectCmd chip0 PALL

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00020000 @DirectCmd chip0 EMRS2

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00030000 @DirectCmd chip0 EMRS3

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00010400 @DirectCmd chip0 EMRS1 (MEM DLL on, DQS# disable)

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00000542 @DirectCmd chip0 MRS (MEM DLL reset) CL=4, BL=4

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x01000000 @DirectCmd chip0 PALL

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x05000000 @DirectCmd chip0 REFA

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x05000000 @DirectCmd chip0 REFA

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00000442 @DirectCmd chip0 MRS (MEM DLL unreset)

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00010780 @DirectCmd chip0 EMRS1 (OCD default)

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00010400 @DirectCmd chip0 EMRS1 (OCD exit)

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x07100000 @DirectCmd chip1 Deselect

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x01100000 @DirectCmd chip1 PALL

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00120000 @DirectCmd chip1 EMRS2

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00130000 @DirectCmd chip1 EMRS3

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00110440 @DirectCmd chip1 EMRS1 (MEM DLL on, DQS# disable)

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00100542 @DirectCmd chip1 MRS (MEM DLL reset) CL=4, BL=4

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x01100000 @DirectCmd chip1 PALL

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x05100000 @DirectCmd chip1 REFA

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x05100000 @DirectCmd chip1 REFA

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00100442 @DirectCmd chip1 MRS (MEM DLL unreset)

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00110780 @DirectCmd chip1 EMRS1 (OCD default)

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x00110400 @DirectCmd chip1 EMRS1 (OCD exit)

str r1, [r0, #DMC\_DIRECTCMD]

ldr r1, =0x0FF02030 @ConControl auto refresh on

str r1, [r0, #DMC\_CONCONTROL]

ldr r1, =0xFFFF00FF @PwrdnConfig

str r1, [r0, #DMC\_PWRDNCONFIG]

ldr r1, =DMC1\_MEMCONTROL @MemControl BL=4, 2 chip, DDR2 type, dynamic self refresh, force precharge, dynamic power down off

str r1, [r0, #DMC\_MEMCONTROL]

// 函数返回

mov pc, lr

s5pv210.h(硬件设备头文件)：

/\* S5PC110 device base addresses \*/

#define ELFIN\_DMA\_BASE 0xE0900000

#define ELFIN\_LCD\_BASE 0xF8000000

#define ELFIN\_USB\_HOST\_BASE 0xEC200000

#define ELFIN\_I2C\_BASE 0xE1800000

#define ELFIN\_I2S\_BASE 0xE2100000

#define ELFIN\_ADC\_BASE 0xE1700000

#define ELFIN\_SPI\_BASE 0xE1300000

#define ELFIN\_HSMMC\_0\_BASE 0xEB000000

#define ELFIN\_HSMMC\_1\_BASE 0xEB100000

#define ELFIN\_HSMMC\_2\_BASE 0xEB200000

#define ELFIN\_HSMMC\_3\_BASE 0xEB300000

#define ELFIN\_CLOCK\_POWER\_BASE 0xE0100000

#define IO\_RET\_REL ((1 << 31) | (1 << 29) | (1 << 28))

/\* Clock & Power Controller for s5pc110\*/

#define APLL\_LOCK\_OFFSET 0x00

#define MPLL\_LOCK\_OFFSET 0x08

#define EPLL\_LOCK\_OFFSET 0x10

#define VPLL\_LOCK\_OFFSET 0x20

#define APLL\_CON0\_OFFSET 0x100

#define APLL\_CON1\_OFFSET 0x104

#define MPLL\_CON\_OFFSET 0x108

#define EPLL\_CON\_OFFSET 0x110

#define VPLL\_CON\_OFFSET 0x120

#define CLK\_SRC0\_OFFSET 0x200

#define CLK\_SRC1\_OFFSET 0x204

#define CLK\_SRC2\_OFFSET 0x208

#define CLK\_SRC3\_OFFSET 0x20c

#define CLK\_SRC4\_OFFSET 0x210

#define CLK\_SRC5\_OFFSET 0x214

#define CLK\_SRC6\_OFFSET 0x218

#define CLK\_SRC\_MASK0\_OFFSET 0x280

#define CLK\_SRC\_MASK1\_OFFSET 0x284

#define CLK\_DIV0\_OFFSET 0x300

#define CLK\_DIV1\_OFFSET 0x304

#define CLK\_DIV2\_OFFSET 0x308

#define CLK\_DIV3\_OFFSET 0x30c

#define CLK\_DIV4\_OFFSET 0x310

#define CLK\_DIV5\_OFFSET 0x314

#define CLK\_DIV6\_OFFSET 0x318

#define CLK\_DIV7\_OFFSET 0x31c

#define CLK\_GATE\_IP0\_OFFSET 0x460

#define CLK\_GATE\_IP1\_OFFSET 0x464

#define CLK\_GATE\_IP2\_OFFSET 0x468

#define CLK\_GATE\_IP3\_OFFSET 0x46c

#define CLK\_GATE\_IP4\_OFFSET 0x470

#define CLK\_GATE\_BLOCK\_OFFSET 0x480

#define CLK\_OUT\_OFFSET 0x500

#define CLK\_DIV\_STAT0\_OFFSET 0x1000

#define CLK\_DIV\_STAT1\_OFFSET 0x1004

#define CLK\_MUX\_STAT0\_OFFSET 0x1100

#define CLK\_MUX\_STAT1\_OFFSET 0x1104

#define SW\_RST\_OFFSET 0x2000

#define ONEDRAM\_CFG\_OFFSET 0x6208

#define OSC\_CON\_OFFSET 0x8000

#define RST\_STAT\_OFFSET 0xa000

#define PWR\_CFG\_OFFSET 0xc000

#define EINT\_WAKEUP\_MASK\_OFFSET0xc004

#define WAKEUP\_MASK\_OFFSET 0xc008

#define NORMAL\_CFG\_OFFSET 0xc010

#define IDLE\_CFG\_OFFSET 0xc020

#define STOP\_CFG\_OFFSET 0xc030

#define STOP\_MEM\_CFG\_OFFSET 0xc034

#define SLEEP\_CFG\_OFFSET 0xc040

#define OSC\_FREQ\_OFFSET 0xc100

#define OSC\_STABLE\_OFFSET 0xc104

#define PWR\_STABLE\_OFFSET 0xc108

#define MTC\_STABLE\_OFFSET 0xc110

#define CLAMP\_STABLE\_OFFSET 0xc114

#define WAKEUP\_STAT\_OFFSET 0xc200

#define BLK\_PWR\_STAT\_OFFSET 0xc204

#define BODY\_BIAS\_CON\_OFFSET 0xc300

#define ION\_SKEW\_CON\_OFFSET 0xc310

#define ION\_SKEW\_MON\_OFFSET 0xc314

#define IOFF\_SKEW\_CON\_OFFSET 0xc320

#define IOFF\_SKEW\_MON\_OFFSET 0xc324

#define OTHERS\_OFFSET 0xe000

#define OM\_STAT\_OFFSET 0xe100

#define MIE\_CONTROL\_OFFSET 0xe800

#define HDMI\_CONTROL\_OFFSET 0xe804

#define USB\_PHY\_CONTROL\_OFFSET 0xe80c

#define DAC\_CONTROL\_OFFSET 0xe810

#define MIPI\_DPHY\_CONTROL\_OFFSET 0xe814

#define ADC\_CONTROL\_OFFSET 0xe818

#define PS\_HOLD\_CONTROL\_OFFSET 0xe81c

#define INFORM0\_OFFSET 0xf000

#define INFORM1\_OFFSET 0xf004

#define INFORM2\_OFFSET 0xf008

#define INFORM3\_OFFSET 0xf00c

#define INFORM4\_OFFSET 0xf010

#define INFORM5\_OFFSET 0xf014

#define INFORM6\_OFFSET 0xf018

#define INFORM7\_OFFSET 0xf01c

#define INF\_REG0\_OFFSET 0x00

#define INF\_REG1\_OFFSET 0x04

#define INF\_REG2\_OFFSET 0x08

#define INF\_REG3\_OFFSET 0x0c

#define INF\_REG4\_OFFSET 0x10

#define INF\_REG5\_OFFSET 0x14

#define INF\_REG6\_OFFSET 0x18

#define INF\_REG7\_OFFSET 0x1c

/\*

\* GPIO

\*/

#define ELFIN\_GPIO\_BASE 0xE0200000

#define GPA0CON\_OFFSET 0x000

#define GPA0DAT\_OFFSET 0x004

#define GPA0PUD\_OFFSET 0x008

#define GPA0DRV\_SR\_OFFSET 0x00C

#define GPA0CONPDN\_OFFSET 0x010

#define GPA0PUDPDN\_OFFSET 0x014

#define GPA1CON\_OFFSET 0x020

#define GPA1DAT\_OFFSET 0x024

#define GPA1PUD\_OFFSET 0x028

#define GPA1DRV\_SR\_OFFSET 0x02C

#define GPA1CONPDN\_OFFSET 0x030

#define GPA1PUDPDN\_OFFSET 0x034

#define GPBCON\_OFFSET 0x040

#define GPBDAT\_OFFSET 0x044

#define GPBPUD\_OFFSET 0x048

#define GPBDRV\_SR\_OFFSET 0x04C

#define GPBCONPDN\_OFFSET 0x050

#define GPBPUDPDN\_OFFSET 0x054

#define GPC0CON\_OFFSET 0x060

#define GPC0DAT\_OFFSET 0x064

#define GPC0PUD\_OFFSET 0x068

#define GPC0DRV\_SR\_OFFSET 0x06C

#define GPC0CONPDN\_OFFSET 0x070

#define GPC0PUDPDN\_OFFSET 0x074

#define GPC1CON\_OFFSET 0x080

#define GPC1DAT\_OFFSET 0x084

#define GPC1PUD\_OFFSET 0x088

#define GPC1DRV\_SR\_OFFSET 0x08C

#define GPC1CONPDN\_OFFSET 0x090

#define GPC1PUDPDN\_OFFSET 0x094

#define GPD0CON\_OFFSET 0x0A0

#define GPD0DAT\_OFFSET 0x0A4

#define GPD0PUD\_OFFSET 0x0A8

#define GPD0DRV\_SR\_OFFSET 0x0AC

#define GPD0CONPDN\_OFFSET 0x0B0

#define GPD0PUDPDN\_OFFSET 0x0B4

#define GPD1CON\_OFFSET 0x0C0

#define GPD1DAT\_OFFSET 0x0C4

#define GPD1PUD\_OFFSET 0x0C8

#define GPD1DRV\_SR\_OFFSET 0x0CC

#define GPD1CONPDN\_OFFSET 0x0D0

#define GPD1PUDPDN\_OFFSET 0x0D4

#define GPE0CON\_OFFSET 0x0E0

#define GPE0DAT\_OFFSET 0x0E4

#define GPE0PUD\_OFFSET 0x0E8

#define GPE0DRV\_SR\_OFFSET 0x0EC

#define GPE0CONPDN\_OFFSET 0x0F0

#define GPE0PUDPDN\_OFFSET 0x0F4

#define GPE1CON\_OFFSET 0x100

#define GPE1DAT\_OFFSET 0x104

#define GPE1PUD\_OFFSET 0x108

#define GPE1DRV\_SR\_OFFSET 0x10C

#define GPE1CONPDN\_OFFSET 0x110

#define GPE1PUDPDN\_OFFSET 0x114

#define GPF0CON\_OFFSET 0x120

#define GPF0DAT\_OFFSET 0x124

#define GPF0PUD\_OFFSET 0x128

#define GPF0DRV\_SR\_OFFSET 0x12C

#define GPF0CONPDN\_OFFSET 0x130

#define GPF0PUDPDN\_OFFSET 0x134

#define GPF1CON\_OFFSET 0x140

#define GPF1DAT\_OFFSET 0x144

#define GPF1PUD\_OFFSET 0x148

#define GPF1DRV\_SR\_OFFSET 0x14C

#define GPF1CONPDN\_OFFSET 0x150

#define GPF1PUDPDN\_OFFSET 0x154

#define GPF2CON\_OFFSET 0x160

#define GPF2DAT\_OFFSET 0x164

#define GPF2PUD\_OFFSET 0x168

#define GPF2DRV\_SR\_OFFSET 0x16C

#define GPF2CONPDN\_OFFSET 0x170

#define GPF2PUDPDN\_OFFSET 0x174

#define GPF3CON\_OFFSET 0x180

#define GPF3DAT\_OFFSET 0x184

#define GPF3PUD\_OFFSET 0x188

#define GPF3DRV\_SR\_OFFSET 0x18C

#define GPF3CONPDN\_OFFSET 0x190

#define GPF3PUDPDN\_OFFSET 0x194

#define GPG0CON\_OFFSET 0x1A0

#define GPG0DAT\_OFFSET 0x1A4

#define GPG0PUD\_OFFSET 0x1A8

#define GPG0DRV\_SR\_OFFSET 0x1AC

#define GPG0CONPDN\_OFFSET 0x1B0

#define GPG0PUDPDN\_OFFSET 0x1B4

#define GPG1CON\_OFFSET 0x1C0

#define GPG1DAT\_OFFSET 0x1C4

#define GPG1PUD\_OFFSET 0x1C8

#define GPG1DRV\_SR\_OFFSET 0x1CC

#define GPG1CONPDN\_OFFSET 0x1D0

#define GPG1PUDPDN\_OFFSET 0x1D4

#define GPG2CON\_OFFSET 0x1E0

#define GPG2DAT\_OFFSET 0x1E4

#define GPG2PUD\_OFFSET 0x1E8

#define GPG2DRV\_SR\_OFFSET 0x1EC

#define GPG2CONPDN\_OFFSET 0x1F0

#define GPG2PUDPDN\_OFFSET 0x1F4

#define GPG3CON\_OFFSET 0x200

#define GPG3DAT\_OFFSET 0x204

#define GPG3PUD\_OFFSET 0x208

#define GPG3DRV\_SR\_OFFSET 0x20C

#define GPG3CONPDN\_OFFSET 0x210

#define GPG3PUDPDN\_OFFSET 0x214

#define MP1\_0DRV\_SR\_OFFSET 0x3CC

#define MP1\_1DRV\_SR\_OFFSET 0x3EC

#define MP1\_2DRV\_SR\_OFFSET 0x40C

#define MP1\_3DRV\_SR\_OFFSET 0x42C

#define MP1\_4DRV\_SR\_OFFSET 0x44C

#define MP1\_5DRV\_SR\_OFFSET 0x46C

#define MP1\_6DRV\_SR\_OFFSET 0x48C

#define MP1\_7DRV\_SR\_OFFSET 0x4AC

#define MP1\_8DRV\_SR\_OFFSET 0x4CC

#define MP2\_0DRV\_SR\_OFFSET 0x4EC

#define MP2\_1DRV\_SR\_OFFSET 0x50C

#define MP2\_2DRV\_SR\_OFFSET 0x52C

#define MP2\_3DRV\_SR\_OFFSET 0x54C

#define MP2\_4DRV\_SR\_OFFSET 0x56C

#define MP2\_5DRV\_SR\_OFFSET 0x58C

#define MP2\_6DRV\_SR\_OFFSET 0x5AC

#define MP2\_7DRV\_SR\_OFFSET 0x5CC

#define MP2\_8DRV\_SR\_OFFSET 0x5EC

/\* GPH0 \*/

#define GPH0CON\_OFFSET 0xc00

#define GPH0DAT\_OFFSET 0xc04

#define GPH0PUD\_OFFSET 0xc08

#define GPH0DRV\_OFFSET 0xc0c

/\* GPH1 \*/

#define GPH1CON\_OFFSET 0xc20

#define GPH1DAT\_OFFSET 0xc24

#define GPH1PUD\_OFFSET 0xc28

#define GPH1DRV\_OFFSET 0xc2c

/\* GPH2 \*/

#define GPH2CON\_OFFSET 0xc40

#define GPH2DAT\_OFFSET 0xc44

#define GPH2PUD\_OFFSET 0xc48

#define GPH2DRV\_OFFSET 0xc4c

/\* GPH3 \*/

#define GPH3CON\_OFFSET 0xc60

#define GPH3DAT\_OFFSET 0xc64

#define GPH3PUD\_OFFSET 0xc68

#define GPH3DRV\_OFFSET 0xc6c

#define GPICON\_OFFSET 0x220

#define GPIPUD\_OFFSET 0x228

#define GPIDRV\_OFFSET\_SR 0x22C

#define GPIPUDPDN\_OFFSET 0x234

#define GPJ0CON\_OFFSET 0x240

#define GPJ0DAT\_OFFSET 0x244

#define GPJ0PUD\_OFFSET 0x248

#define GPJ0DRV\_SR\_OFFSET 0x24C

#define GPJ0CONPDN\_OFFSET 0x250

#define GPJ0PUDPDN\_OFFSET 0x254

#define GPJ1CON\_OFFSET 0x260

#define GPJ1DAT\_OFFSET 0x264

#define GPJ1PUD\_OFFSET 0x268

#define GPJ1DRV\_SR\_OFFSET 0x26C

#define GPJ1CONPDN\_OFFSET 0x270

#define GPJ1PUDPDN\_OFFSET 0x274

#define GPJ2CON\_OFFSET 0x280

#define GPJ2DAT\_OFFSET 0x284

#define GPJ2PUD\_OFFSET 0x288

#define GPJ2DRV\_SR\_OFFSET 0x28C

#define GPJ2CONPDN\_OFFSET 0x290

#define GPJ2PUDPDN\_OFFSET 0x294

#define GPJ3CON\_OFFSET 0x2A0

#define GPJ3DAT\_OFFSET 0x2A4

#define GPJ3PUD\_OFFSET 0x2A8

#define GPJ3DRV\_SR\_OFFSET 0x2AC

#define GPJ3CONPDN\_OFFSET 0x2B0

#define GPJ3PUDPDN\_OFFSET 0x2B4

#define GPJ4CON\_OFFSET 0x2C0

#define GPJ4DAT\_OFFSET 0x2C4

#define GPJ4PUD\_OFFSET 0x2C8

#define GPJ4DRV\_SR\_OFFSET 0x2CC

#define GPJ4CONPDN\_OFFSET 0x2D0

#define GPJ4PUDPDN\_OFFSET 0x2D4

/\*

\* Interrupt

\*/

#define ELFIN\_VIC0\_BASE\_ADDR (0xF2000000)

#define ELFIN\_VIC1\_BASE\_ADDR (0xF2100000)

#define ELFIN\_VIC2\_BASE\_ADDR (0xF2200000)

#define ELFIN\_TZIC0\_BASE\_ADDR (0xF2800000)

#define ELFIN\_TZIC1\_BASE\_ADDR (0xF2900000)

#define ELFIN\_TZIC2\_BASE\_ADDR (0xF2A00000)

#define oINTMOD (0x0C)// VIC INT SELECT (IRQ or FIQ)

#define oINTUNMSK (0x10)// VIC INT EN (Unmask by writing 1)

#define oINTMSK (0x14)// VIC INT EN CLEAR (Mask by writing 1)

#define oINTSUBMSK (0x1C)// VIC SOFT INT CLEAR

#define oVECTADDR (0xF00)// VIC ADDRESS

/\*

\* Watchdog timer

\*/

#define ELFIN\_WATCHDOG\_BASE 0xE2700000

#define WTCON\_OFFSET 0x00

#define WTDAT\_OFFSET 0x08

#define WTCNT\_OFFSET 0x0C

#define WTCON\_REG \_\_REG(ELFIN\_WATCHDOG\_BASE+WTCON\_OFFSET)

#define WTDAT\_REG \_\_REG(ELFIN\_WATCHDOG\_BASE+WTDAT\_OFFSET)

#define WTCNT\_REG \_\_REG(ELFIN\_WATCHDOG\_BASE+WTCNT\_OFFSET)

/\*

\* UART

\*/

#define ELFIN\_UART\_BASE 0XE2900000

#define ELFIN\_UART0\_OFFSET 0x0000

#define ELFIN\_UART1\_OFFSET 0x0400

#define ELFIN\_UART2\_OFFSET 0x0800

#define ELFIN\_UART3\_OFFSET 0x0c00

#if defined(CONFIG\_SERIAL0)

#define ELFIN\_UART\_CONSOLE\_BASE (ELFIN\_UART\_BASE + ELFIN\_UART0\_OFFSET)

#elif defined(CONFIG\_SERIAL1)

#define ELFIN\_UART\_CONSOLE\_BASE (ELFIN\_UART\_BASE + ELFIN\_UART1\_OFFSET)

#elif defined(CONFIG\_SERIAL2)

#define ELFIN\_UART\_CONSOLE\_BASE (ELFIN\_UART\_BASE + ELFIN\_UART2\_OFFSET)

#elif defined(CONFIG\_SERIAL3)

#define ELFIN\_UART\_CONSOLE\_BASE (ELFIN\_UART\_BASE + ELFIN\_UART3\_OFFSET)

#else

#define ELFIN\_UART\_CONSOLE\_BASE (ELFIN\_UART\_BASE + ELFIN\_UART0\_OFFSET)

#endif

#define ULCON\_OFFSET 0x00

#define UCON\_OFFSET 0x04

#define UFCON\_OFFSET 0x08

#define UMCON\_OFFSET 0x0C

#define UTRSTAT\_OFFSET 0x10

#define UERSTAT\_OFFSET 0x14

#define UFSTAT\_OFFSET 0x18

#define UMSTAT\_OFFSET 0x1C

#define UTXH\_OFFSET 0x20

#define URXH\_OFFSET 0x24

#define UBRDIV\_OFFSET 0x28

#define UDIVSLOT\_OFFSET 0x2C

#define UINTP\_OFFSET 0x30

#define UINTSP\_OFFSET 0x34

#define UINTM\_OFFSET 0x38

#define UTRSTAT\_TX\_EMPTY BIT2

#define UTRSTAT\_RX\_READY BIT0

#define UART\_ERR\_MASK 0xF

/\*

\* PWM timer

\*/

#define ELFIN\_TIMER\_BASE 0xE2500000

#define TCFG0\_REG \_\_REG(0xE2500000)

#define TCFG1\_REG \_\_REG(0xE2500004)

#define TCON\_REG \_\_REG(0xE2500008)

#define TCNTB0\_REG \_\_REG(0xE250000c)

#define TCMPB0\_REG \_\_REG(0xE2500010)

#define TCNTO0\_REG \_\_REG(0xE2500014)

#define TCNTB1\_REG \_\_REG(0xE2500018)

#define TCMPB1\_REG \_\_REG(0xE250001c)

#define TCNTO1\_REG \_\_REG(0xE2500020)

#define TCNTB2\_REG \_\_REG(0xE2500024)

#define TCMPB2\_REG \_\_REG(0xE2500028)

#define TCNTO2\_REG \_\_REG(0xE250002c)

#define TCNTB3\_REG \_\_REG(0xE2500030)

#define TCMPB3\_REG \_\_REG(0xE2500034)

#define TCNTO3\_REG \_\_REG(0xE2500038)

#define TCNTB4\_REG \_\_REG(0xE250003c)

#define TCNTO4\_REG \_\_REG(0xE2500040)

#define TINT\_CSTAT \_\_REG(0xE2500044)

/\* Fields \*/

#define fTCFG0\_DZONE Fld(8,16) /\* the dead zone length (= timer 0) \*/

#define fTCFG0\_PRE1 Fld(8,8) /\* prescaler value for time 2,3,4 \*/

#define fTCFG0\_PRE0 Fld(8,0) /\* prescaler value for time 0,1 \*/

#define fTCFG1\_MUX4 Fld(4,16)

/\* bits \*/

#define TCFG0\_DZONE(x) FInsrt((x), fTCFG0\_DZONE)

#define TCFG0\_PRE1(x) FInsrt((x), fTCFG0\_PRE1)

#define TCFG0\_PRE0(x) FInsrt((x), fTCFG0\_PRE0)

#define TCON\_4\_AUTO (1 << 22) /\* auto reload on/off for Timer 4 \*/

#define TCON\_4\_UPDATE (1 << 21) /\* manual Update TCNTB4 \*/

#define TCON\_4\_ONOFF (1 << 20) /\* 0: Stop, 1: start Timer 4 \*/

#define COUNT\_4\_ON (TCON\_4\_ONOFF\*1)

#define COUNT\_4\_OFF (TCON\_4\_ONOFF\*0)

#define TCON\_3\_AUTO (1 << 19) /\* auto reload on/off for Timer 3 \*/

#define TIMER3\_ATLOAD\_ON (TCON\_3\_AUTO\*1)

#define TIMER3\_ATLAOD\_OFF FClrBit(TCON, TCON\_3\_AUTO)

#define TCON\_3\_INVERT (1 << 18) /\* 1: Inverter on for TOUT3 \*/

#define TIMER3\_IVT\_ON (TCON\_3\_INVERT\*1)

#define TIMER3\_IVT\_OFF (FClrBit(TCON, TCON\_3\_INVERT))

#define TCON\_3\_MAN (1 << 17) /\* manual Update TCNTB3,TCMPB3 \*/

#define TIMER3\_MANUP (TCON\_3\_MAN\*1)

#define TIMER3\_NOP (FClrBit(TCON, TCON\_3\_MAN))

#define TCON\_3\_ONOFF (1 << 16) /\* 0: Stop, 1: start Timer 3 \*/

#define TIMER3\_ON (TCON\_3\_ONOFF\*1)

#define TIMER3\_OFF (FClrBit(TCON, TCON\_3\_ONOFF))

/\* macros \*/

#define GET\_PRESCALE\_TIMER4(x) FExtr((x), fTCFG0\_PRE1)

#define GET\_DIVIDER\_TIMER4(x) FExtr((x), fTCFG1\_MUX4)

#define MP01CON\_OFFSET 0x2E0

#define MP01DAT\_OFFSET 0x2E4

#define MP01PUD\_OFFSET 0x2E8

#define MP01DRV\_SR\_OFFSET 0x2EC

#define MP01CONPDN\_OFFSET 0x2E0

#define MP01PUDPDN\_OFFSET 0x2E4

#define MP02CON\_OFFSET 0x300

#define MP02DAT\_OFFSET 0x304

#define MP02PUD\_OFFSET 0x308

#define MP02DRV\_SR\_OFFSET 0x30c

#define MP02CONPDN\_OFFSET 0x310

#define MP02PUDPDN\_OFFSET 0x314

#define MP03CON\_OFFSET 0x320

#define MP03DAT\_OFFSET 0x324

#define MP03PUD\_OFFSET 0x328

#define MP03DRV\_SR\_OFFSET 0x32c

#define MP03CONPDN\_OFFSET 0x330

#define MP03PUDPDN\_OFFSET 0x334

#define MP06CON\_OFFSET 0x380

#define MP06DAT\_OFFSET 0x384

#define MP06PUD\_OFFSET 0x388

#define MP06DRV\_SR\_OFFSET 0x38C

#define MP06CONPDN\_OFFSET 0x390

#define MP06PUDPDN\_OFFSET 0x394

#define MP07CON\_OFFSET 0x3A0

#define MP07DAT\_OFFSET 0x3A4

#define MP07PUD\_OFFSET 0x3A8

#define MP07DRV\_SR\_OFFSET 0x3AC

#define MP07CONPDN\_OFFSET 0x3B0

#define MP07PUDPDN\_OFFSET 0x3B4

/\*

\* Nand flash controller

\*/

#define ELFIN\_NAND\_BASE 0xB0E00000

#define ELFIN\_NAND\_ECC\_BASE 0xB0E20000

#define NFCONF\_OFFSET 0x00

#define NFCONT\_OFFSET 0x04

#define NFCMMD\_OFFSET 0x08

#define NFADDR\_OFFSET 0x0c

#define NFDATA\_OFFSET 0x10

#define NFMECCDATA0\_OFFSET 0x14

#define NFMECCDATA1\_OFFSET 0x18

#define NFSECCDATA0\_OFFSET 0x1c

#define NFSBLK\_OFFSET 0x20

#define NFEBLK\_OFFSET 0x24

#define NFSTAT\_OFFSET 0x28

#define NFESTAT0\_OFFSET 0x2c

#define NFESTAT1\_OFFSET 0x30

#define NFMECC0\_OFFSET 0x34

#define NFMECC1\_OFFSET 0x38

#define NFSECC\_OFFSET 0x3c

#define NFMLCBITPT\_OFFSET 0x40

#define NFECCCONF\_OFFSET 0x000 // R/W ECC configuration register 0x0000\_0000

#define NFECCCONT\_OFFSET 0x020 // R/W ECC control register 0x0000\_0000

#define NFECCSTAT\_OFFSET 0x030 // R ECC status register 0x0000\_0000

#define NFECCSECSTAT\_OFFSET 0x040 // R ECC sector status register 0x0000\_0000

#define NFECCPRGECC0\_OFFSET 0x090 // R ECC parity code0 register for page program 0x0000\_0000

#define NFECCPRGECC1\_OFFSET 0x094 // R ECC parity code1 register for page program 0x0000\_0000

#define NFECCPRGECC2\_OFFSET 0x098 // R ECC parity code2 register for page program 0x0000\_0000

#define NFECCPRGECC3\_OFFSET 0x09C // R ECC parity code3 register for page program 0x0000\_0000

#define NFECCPRGECC4\_OFFSET 0x0A0 // R ECC parity code4 register for page program 0x0000\_0000

#define NFECCPRGECC5\_OFFSET 0x0A4 // R ECC parity code5 register for page program 0x0000\_0000

#define NFECCPRGECC6\_OFFSET 0x0A8 // R ECC parity code6 register for page program 0x0000\_0000

#define NFECCERL0\_OFFSET 0x0C0 // R ECC error byte location0 register 0x0000\_0000

#define NFECCERL1\_OFFSET 0x0C4 // R ECC error byte location1 register 0x0000\_0000

#define NFECCERL2\_OFFSET 0x0C8 // R ECC error byte location2 register 0x0000\_0000

#define NFECCERL3\_OFFSET 0x0CC // R ECC error byte location3 register 0x0000\_0000

#define NFECCERL4\_OFFSET 0x0D0 // R ECC error byte location4 register 0x0000\_0000

#define NFECCERL5\_OFFSET 0x0D4 // R ECC error byte location5 register 0x0000\_0000

#define NFECCERL6\_OFFSET 0x0D8 // R ECC error byte location6 register 0x0000\_0000

#define NFECCERL7\_OFFSET 0x0DC // R ECC error byte location7 register 0x0000\_0000

#define NFECCERP0\_OFFSET 0x0F0 // R ECC error bit pattern0 register 0x0000\_0000

#define NFECCERP1\_OFFSET 0x0F4 // R ECC error bit pattern1 register 0x0000\_0000

#define NFECCERP2\_OFFSET 0x0F8 // R ECC error bit pattern2 register 0x0000\_0000

#define NFECCERP3\_OFFSET 0x0FC // R ECC error bit pattern3 register 0x0000\_0000

#define NFECCCONECC0\_OFFSET 0x110 // R/W ECC parity conversion code0 register 0x0000\_0000

#define NFECCCONECC1\_OFFSET 0x114 // R/W ECC parity conversion code1 register 0x0000\_0000

#define NFECCCONECC2\_OFFSET 0x118 // R/W ECC parity conversion code2 register 0x0000\_0000

#define NFECCCONECC3\_OFFSET 0x11C // R/W ECC parity conversion code3 register 0x0000\_0000

#define NFECCCONECC4\_OFFSET 0x120 // R/W ECC parity conversion code4 register 0x0000\_0000

#define NFECCCONECC5\_OFFSET 0x124 // R/W ECC parity conversion code5 register 0x0000\_0000

#define NFECCCONECC6\_OFFSET 0x128 // R/W ECC parity conversion code6 register 0x0000\_0000

#define NFCONF (ELFIN\_NAND\_BASE+NFCONF\_OFFSET)

#define NFCONT (ELFIN\_NAND\_BASE+NFCONT\_OFFSET)

#define NFCMMD (ELFIN\_NAND\_BASE+NFCMMD\_OFFSET)

#define NFADDR (ELFIN\_NAND\_BASE+NFADDR\_OFFSET)

#define NFDATA (ELFIN\_NAND\_BASE+NFDATA\_OFFSET)

#define NFMECCDATA0 (ELFIN\_NAND\_BASE+NFMECCDATA0\_OFFSET)

#define NFMECCDATA1 (ELFIN\_NAND\_BASE+NFMECCDATA1\_OFFSET)

#define NFSECCDATA0 (ELFIN\_NAND\_BASE+NFSECCDATA0\_OFFSET)

#define NFSBLK (ELFIN\_NAND\_BASE+NFSBLK\_OFFSET)

#define NFEBLK (ELFIN\_NAND\_BASE+NFEBLK\_OFFSET)

#define NFSTAT (ELFIN\_NAND\_BASE+NFSTAT\_OFFSET)

#define NFESTAT0 (ELFIN\_NAND\_BASE+NFESTAT0\_OFFSET)

#define NFESTAT1 (ELFIN\_NAND\_BASE+NFESTAT1\_OFFSET)

#define NFMECC0 (ELFIN\_NAND\_BASE+NFMECC0\_OFFSET)

#define NFMECC1 (ELFIN\_NAND\_BASE+NFMECC1\_OFFSET)

#define NFSECC (ELFIN\_NAND\_BASE+NFSECC\_OFFSET)

#define NFMLCBITPT (ELFIN\_NAND\_BASE+NFMLCBITPT\_OFFSET)

#define NFECCCONF (ELFIN\_NAND\_ECC\_BASE+NFECCCONF\_OFFSET)

#define NFECCCONT (ELFIN\_NAND\_ECC\_BASE+NFECCCONT\_OFFSET)

#define NFECCSTAT (ELFIN\_NAND\_ECC\_BASE+NFECCSTAT\_OFFSET)

#define NFECCSECSTAT (ELFIN\_NAND\_ECC\_BASE+NFECCSECSTAT\_OFFSET)

#define NFECCPRGECC0 (ELFIN\_NAND\_ECC\_BASE+NFECCPRGECC0\_OFFSET)

#define NFECCPRGECC1 (ELFIN\_NAND\_ECC\_BASE+NFECCPRGECC1\_OFFSET)

#define NFECCPRGECC2 (ELFIN\_NAND\_ECC\_BASE+NFECCPRGECC2\_OFFSET)

#define NFECCPRGECC3 (ELFIN\_NAND\_ECC\_BASE+NFECCPRGECC3\_OFFSET)

#define NFECCPRGECC4 (ELFIN\_NAND\_ECC\_BASE+NFECCPRGECC4\_OFFSET)

#define NFECCPRGECC5 (ELFIN\_NAND\_ECC\_BASE+NFECCPRGECC5\_OFFSET)

#define NFECCPRGECC6 (ELFIN\_NAND\_ECC\_BASE+NFECCPRGECC6\_OFFSET)

#define NFECCERL0 (ELFIN\_NAND\_ECC\_BASE+NFECCERL0\_OFFSET)

#define NFECCERL1 (ELFIN\_NAND\_ECC\_BASE+NFECCERL1\_OFFSET)

#define NFECCERL2 (ELFIN\_NAND\_ECC\_BASE+NFECCERL2\_OFFSET)

#define NFECCERL3 (ELFIN\_NAND\_ECC\_BASE+NFECCERL3\_OFFSET)

#define NFECCERL4 (ELFIN\_NAND\_ECC\_BASE+NFECCERL4\_OFFSET)

#define NFECCERL5 (ELFIN\_NAND\_ECC\_BASE+NFECCERL5\_OFFSET)

#define NFECCERL6 (ELFIN\_NAND\_ECC\_BASE+NFECCERL6\_OFFSET)

#define NFECCERL7 (ELFIN\_NAND\_ECC\_BASE+NFECCERL7\_OFFSET)

#define NFECCERP0 (ELFIN\_NAND\_ECC\_BASE+NFECCERP0\_OFFSET)

#define NFECCERP1 (ELFIN\_NAND\_ECC\_BASE+NFECCERP1\_OFFSET)

#define NFECCERP2 (ELFIN\_NAND\_ECC\_BASE+NFECCERP2\_OFFSET)

#define NFECCERP3 (ELFIN\_NAND\_ECC\_BASE+NFECCERP3\_OFFSET)

#define NFECCCONECC0 (ELFIN\_NAND\_ECC\_BASE+NFECCCONECC0\_OFFSET)

#define NFECCCONECC1 (ELFIN\_NAND\_ECC\_BASE+NFECCCONECC1\_OFFSET)

#define NFECCCONECC2 (ELFIN\_NAND\_ECC\_BASE+NFECCCONECC2\_OFFSET)

#define NFECCCONECC3 (ELFIN\_NAND\_ECC\_BASE+NFECCCONECC3\_OFFSET)

#define NFECCCONECC4 (ELFIN\_NAND\_ECC\_BASE+NFECCCONECC4\_OFFSET)

#define NFECCCONECC5 (ELFIN\_NAND\_ECC\_BASE+NFECCCONECC5\_OFFSET)

#define NFECCCONECC6 (ELFIN\_NAND\_ECC\_BASE+NFECCCONECC6\_OFFSET)

#define NFCONF\_REG \_\_REG(ELFIN\_NAND\_BASE+NFCONF\_OFFSET)

#define NFCONT\_REG \_\_REG(ELFIN\_NAND\_BASE+NFCONT\_OFFSET)

#define NFCMD\_REG \_\_REG(ELFIN\_NAND\_BASE+NFCMMD\_OFFSET)

#define NFADDR\_REG \_\_REG(ELFIN\_NAND\_BASE+NFADDR\_OFFSET)

#define NFDATA\_REG \_\_REG(ELFIN\_NAND\_BASE+NFDATA\_OFFSET)

#define NFDATA8\_REG \_\_REGb(ELFIN\_NAND\_BASE+NFDATA\_OFFSET)

#define NFMECCDATA0\_REG \_\_REG(ELFIN\_NAND\_BASE+NFMECCDATA0\_OFFSET)

#define NFMECCDATA1\_REG \_\_REG(ELFIN\_NAND\_BASE+NFMECCDATA1\_OFFSET)

#define NFSECCDATA0\_REG \_\_REG(ELFIN\_NAND\_BASE+NFSECCDATA0\_OFFSET)

#define NFSBLK\_REG \_\_REG(ELFIN\_NAND\_BASE+NFSBLK\_OFFSET)

#define NFEBLK\_REG \_\_REG(ELFIN\_NAND\_BASE+NFEBLK\_OFFSET)

#define NFSTAT\_REG \_\_REG(ELFIN\_NAND\_BASE+NFSTAT\_OFFSET)

#define NFESTAT0\_REG \_\_REG(ELFIN\_NAND\_BASE+NFESTAT0\_OFFSET)

#define NFESTAT1\_REG \_\_REG(ELFIN\_NAND\_BASE+NFESTAT1\_OFFSET)

#define NFMECC0\_REG \_\_REG(ELFIN\_NAND\_BASE+NFMECC0\_OFFSET)

#define NFMECC1\_REG \_\_REG(ELFIN\_NAND\_BASE+NFMECC1\_OFFSET)

#define NFSECC\_REG \_\_REG(ELFIN\_NAND\_BASE+NFSECC\_OFFSET)

#define NFMLCBITPT\_REG \_\_REG(ELFIN\_NAND\_BASE+NFMLCBITPT\_OFFSET)

#define NFCONF\_ECC\_MLC (1<<24)

#define NFCONF\_ECC\_1BIT (0<<23)

#define NFCONF\_ECC\_4BIT (2<<23)

#define NFCONF\_ECC\_8BIT (1<<23)

#define NFCONT\_ECC\_ENC (1<<18)

#define NFCONT\_WP (1<<16)

#define NFCONT\_MECCLOCK (1<<7)

#define NFCONT\_SECCLOCK (1<<6)

#define NFCONT\_INITMECC (1<<5)

#define NFCONT\_INITSECC (1<<4)

#define NFCONT\_INITECC (NFCONT\_INITMECC | NFCONT\_INITSECC)

#define NFCONT\_CS (1<<1)

#define NFSTAT\_ECCENCDONE (1<<25)

#define NFSTAT\_ECCDECDONE (1<<24)

#define NFSTAT\_RnB (1<<0)

#define NFESTAT0\_ECCBUSY (1<<31)

/\* Access Controller (TZPC) \*/

#define TZPC\_DECPROT0SET\_OFFSET 0x804

#define TZPC\_DECPROT1SET\_OFFSET 0x810

#define TZPC\_DECPROT2SET\_OFFSET 0x81C

#define TZPC\_DECPROT3SET\_OFFSET 0x828

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* OneNAND Controller

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define ELFIN\_ONENAND\_BASE 0xB0000000

#define ELFIN\_ONENANDCON\_BASE (ELFIN\_ONENAND\_BASE + 0x600000)

#define ONENAND\_IF\_CTRL\_OFFSET 0x100

#define ONENAND\_IF\_CMD\_OFFSET 0x104

#define ONENAND\_IF\_ASYNC\_TIMING\_CTRL\_OFFSET

0x108

#define ONENAND\_IF\_STATUS\_OFFSET 0x10C

#define DMA\_SRC\_ADDR\_OFFSET 0x400

#define DMA\_SRC\_CFG\_OFFSET 0x404

#define DMA\_DST\_ADDR\_OFFSET 0x408

#define DMA\_DST\_CFG\_OFFSET 0x40C

#define DMA\_TRANS\_SIZE\_OFFSET 0x414

#define DMA\_TRANS\_CMD\_OFFSET 0x418

#define DMA\_TRANS\_STATUS\_OFFSET 0x41C

#define DMA\_TRANS\_DIR\_OFFSET 0x420

#define SQC\_SAO\_OFFSET 0x600

#define SQC\_CMD\_OFFSET 0x608

#define SQC\_STATUS\_OFFSET 0x60C

#define SQC\_CAO\_OFFSET 0x610

#define SQC\_REG\_CTRL\_OFFSET 0x614

#define SQC\_REG\_VAL\_OFFSET 0x618

#define SQC\_BRPAO0\_OFFSET 0x620

#define SQC\_BRPAO1\_OFFSET 0x624

#define INTC\_SQC\_CLR\_OFFSET 0x1000

#define INTC\_DMA\_CLR\_OFFSET 0x1004

#define INTC\_ONENAND\_CLR\_OFFSET 0x1008

#define INTC\_SQC\_MASK\_OFFSET 0x1020

#define INTC\_DMA\_MASK\_OFFSET 0x1024

#define INTC\_ONENAND\_MASK\_OFFSET 0x1028

#define INTC\_SQC\_PEND\_OFFSET 0x1040

#define INTC\_DMA\_PEND\_OFFSET 0x1044

#define INTC\_ONENAND\_PEND\_OFFSET 0x1048

#define INTC\_SQC\_STATUS\_OFFSET 0x1060

#define INTC\_DMA\_STATUS\_OFFSET 0x1064

#define INTC\_ONENAND\_STATUS\_OFFSET 0x1068

/\*

\* Memory controller

\*/

#define ELFIN\_SROM\_BASE 0xE8000000

#define SROM\_BW\_REG \_\_REG(ELFIN\_SROM\_BASE+0x0)

#define SROM\_BC0\_REG \_\_REG(ELFIN\_SROM\_BASE+0x4)

#define SROM\_BC1\_REG \_\_REG(ELFIN\_SROM\_BASE+0x8)

#define SROM\_BC2\_REG \_\_REG(ELFIN\_SROM\_BASE+0xC)

#define SROM\_BC3\_REG \_\_REG(ELFIN\_SROM\_BASE+0x10)

#define SROM\_BC4\_REG \_\_REG(ELFIN\_SROM\_BASE+0x14)

#define SROM\_BC5\_REG \_\_REG(ELFIN\_SROM\_BASE+0x18)

/\*

\* SDRAM Controller

\*/

#define APB\_DMC\_0\_BASE 0xF0000000

#define APB\_DMC\_1\_BASE 0xF1400000

#define ASYNC\_MSYS\_DMC0\_BASE 0xF1E00000

#define DMC\_CONCONTROL 0x00

#define DMC\_MEMCONTROL 0x04

#define DMC\_MEMCONFIG0 0x08

#define DMC\_MEMCONFIG1 0x0C

#define DMC\_DIRECTCMD 0x10

#define DMC\_PRECHCONFIG 0x14

#define DMC\_PHYCONTROL0 0x18

#define DMC\_PHYCONTROL1 0x1C

#define DMC\_RESERVED 0x20

#define DMC\_PWRDNCONFIG 0x28

#define DMC\_TIMINGAREF 0x30

#define DMC\_TIMINGROW 0x34

#define DMC\_TIMINGDATA 0x38

#define DMC\_TIMINGPOWER 0x3C

#define DMC\_PHYSTATUS 0x40

#define DMC\_CHIP0STATUS 0x48

#define DMC\_CHIP1STATUS 0x4C

#define DMC\_AREFSTATUS 0x50

#define DMC\_MRSTATUS 0x54

#define DMC\_PHYTEST0 0x58

#define DMC\_PHYTEST1 0x5C

#define DMC\_QOSCONTROL0 0x60

#define DMC\_QOSCONFIG0 0x64

#define DMC\_QOSCONTROL1 0x68

#define DMC\_QOSCONFIG1 0x6C

#define DMC\_QOSCONTROL2 0x70

#define DMC\_QOSCONFIG2 0x74

#define DMC\_QOSCONTROL3 0x78

#define DMC\_QOSCONFIG3 0x7C

#define DMC\_QOSCONTROL4 0x80

#define DMC\_QOSCONFIG4 0x84

#define DMC\_QOSCONTROL5 0x88

#define DMC\_QOSCONFIG5 0x8C

#define DMC\_QOSCONTROL6 0x90

#define DMC\_QOSCONFIG6 0x94

#define DMC\_QOSCONTROL7 0x98

#define DMC\_QOSCONFIG7 0x9C

#define DMC\_QOSCONTROL8 0xA0

#define DMC\_QOSCONFIG8 0xA4

#define DMC\_QOSCONTROL9 0xA8

#define DMC\_QOSCONFIG9 0xAC

#define DMC\_QOSCONTROL10 0xB0

#define DMC\_QOSCONFIG10 0xB4

#define DMC\_QOSCONTROL11 0xB8

#define DMC\_QOSCONFIG11 0xBC

#define DMC\_QOSCONTROL12 0xC0

#define DMC\_QOSCONFIG12 0xC4

#define DMC\_QOSCONTROL13 0xC8

#define DMC\_QOSCONFIG13 0xCC

#define DMC\_QOSCONTROL14 0xD0

#define DMC\_QOSCONFIG14 0xD4

#define DMC\_QOSCONTROL15 0xD8

#define DMC\_QOSCONFIG15 0xDC

/\*

\* Memory Chip direct command

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Definitions for memory configuration

Set memory configurationactive\_chips

= 1'b0 (1 chip)qos\_master\_chip = 3'b000(ARID[3:0])memory burst

= 3'b010(burst 4)stop\_mem\_clock= 1'b0(disable dynamical stop)auto\_power\_down = 1'b0(disable auto power-down mode)power\_down\_prd= 6'b00\_0000(0 cycle for auto power-down)ap\_bit

= 1'b0 (bit position of auto-precharge is 10)row\_bits

= 3'b010(# row address 13)column\_bits

= 3'b010(# column address 10 )

Set user configuration2'b10=SDRAM/mSDRAM, 2'b11=DDR, 2'b01=mDDR

Set chip select for chip [n]row bank control, bank address 0x3000\_0000 ~ 0x37ff\_ffffCHIP\_[n]\_CFG=0x30F8, 30: ADDR[31:24], F8: Mask[31:24]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define INF\_REG\_BASE 0xE010F000

#define INF\_REG0\_REG \_\_REG(INF\_REG\_BASE+INF\_REG0\_OFFSET)

#define INF\_REG1\_REG \_\_REG(INF\_REG\_BASE+INF\_REG1\_OFFSET)

#define INF\_REG2\_REG \_\_REG(INF\_REG\_BASE+INF\_REG2\_OFFSET)

#define INF\_REG3\_REG \_\_REG(INF\_REG\_BASE+INF\_REG3\_OFFSET)

#define INF\_REG4\_REG \_\_REG(INF\_REG\_BASE+INF\_REG4\_OFFSET)

#define INF\_REG5\_REG \_\_REG(INF\_REG\_BASE+INF\_REG5\_OFFSET)

#define INF\_REG6\_REG \_\_REG(INF\_REG\_BASE+INF\_REG6\_OFFSET)

#define INF\_REG7\_REG \_\_REG(INF\_REG\_BASE+INF\_REG7\_OFFSET)

#define NFCONF\_VAL (7<<12)|(7<<8)|(7<<4)|(0<<3)|(0<<2)|(1<<1)|(0<<0)

#defineNFCONT\_VAL (0x1<<23)|(0x1<<22)|(0<<18)|(0<<17)|(0<<16)|(0<<10)|(0<<9)|(0<<8)|(0<<7)|(0<<6)|(0x2<<1)|(1<<0)

#define MP03CON\_VAL (1<<29)|(1<<25)|(1<<21)|(1<<17)|(1<<13)|(1<<9)|(1<<5)|(1<<1)

#define ELFIN\_TZPC0\_BASE 0xF1500000

#define ELFIN\_TZPC1\_BASE 0xFAD00000

#define ELFIN\_TZPC2\_BASE 0xE0600000

#define ELFIN\_TZPC3\_BASE 0xE1C00000­­­­­