5.4.8 T_MRD 寄存器

寄存器	地址	读/写	描述	复位值
P0T_MRD	0x7E00001C	读/写	16 位 DRAM 控制 t_MRD 寄存器	0x02
P1T_MRD	0x7E00101C	读/写	32 位 DRAM 控制 t_MRD 寄存器	0x02

PnT_MRD	位	描述	初始状态
	[31:7]	读未定义，写为 0	
t_MRD	[6:0]	内存时钟周期内，设置模式寄存器命令时间	0x02

5.4.9 T_RAS 寄存器

寄存器	地址	读/写	描述	复位值
P0T_RAS	0x7E000020	读/写	16 位 DRAM 控制 t_RAS 寄存器	0x7
P1T_RAS	0x7E001020	读/写	32 位 DRAM 控制 t_RAS 寄存器	0x7

PnT_RAS	位	描述	初始状态
	[31:4]	读未定义，写为 0	
t_RAS	[3:0]	内存时钟周期内，设置 RAS 到预充电延迟	0x7

5.4.10 T_RC 寄存器

寄存器	地址	读/写	描述	复位值
P0T_RC	0x7E000024	读/写	16 位 DRAM 控制 t_RC 寄存器	0xB
P1T_RC	0x7E001024	读/写	32 位 DRAM 控制 t_RC 寄存器	0xB

PnT_ RC	位	描述	初始状态
	[31:4]	读未定义，写为 0	
t_ RC	[3:0]	内存时钟周期内，设置有效组件 x 到有效组件 x 延迟	0xB

5.4.11 T_RCD 寄存器

寄存器	地址	读/写	描述	复位值
P0T_RCD	0x7E000028	读/写	16 位 DRAM 控制 t_ RCD 寄存器	0x1D
P1T_ RCD	0x7E001028	读/写	32 位 DRAM 控制 t_ RCD 寄存器	0x1D

PnT_ RCD	位	描述	初始状态
	[31:6]	读未定义，写为 0	
scheduled_RCD	[5:3]	ACLK 周期 3 内，设置 RAS 到 CAS 最小延迟	011
t_ RCD	[2:0]	内存时钟周期内，设置 RAS 到 CAS 最小延迟	101

5.4.12 T_RFC 寄存器

寄存器	地址	读/写	描述	复位值
P0T_RFC	0x7E00002C	读/写	16 位 DRAM 控制 t_ RFC 寄存器	0x212
P1T_ RFC	0x7E00102C	读/写	32 位 DRAM 控制 t_ RFC 寄存器	0x212

PnT_ RFC	位	描述	初始状态
	[31:10]	读未定义，写为 0	
scheduled_ RFC	[9:5]	在 ACLK 周期，设置自动刷新指令时间	0x10
t_ RFC	[4:0]	内存时钟周期内，设置自动刷新指令时间	0x12

5.4.13 T_RP 寄存器

寄存器	地址	读/写	描述	复位值
P0T_RP	0x7E000030	读/写	16 位 DRAM 控制 t_ RP 寄存器	0x1D
P1T_ RP	0x7E001030	读/写	32 位 DRAM 控制 t_ RP 寄存器	0x1D

PnT_ RP	位	描述	初始状态
	[31:6]	读未定义，写为 0	
scheduled_ RP	[5:3]	在 ACLK 周期内，设置预充电到 RAS 延迟	011
t_ RP	[2:0]	内存时钟周期内，设置预充电到 RAS 延迟	101

5.4.14 T_RRD 寄存器

寄存器	地址	读/写	描述	复位值
P0T_RRD	0x7E000034	读/写	16 位 DRAM 控制 t_ RRD 寄存器	0x2
P1T_ RRD	0x7E001034	读/写	32 位 DRAM 控制 t_ RRD 寄存器	0x2

PnT_ RRD	位	描述	初始状态
	[31:4]	读未定义，写为 0	
t_ RRD	[3:0]	内存时钟周期内，设置有效页 x 到有效页 y 延迟	0x2

5.4.15 T_WR 寄存器

寄存器	地址	读/写	描述	复位值
P0T_WR	0x7E000038	读/写	16 位 DRAM 控制 t_ WR 寄存器	0x3
P1T_ WR	0x7E001038	读/写	32 位 DRAM 控制 t_ WR 寄存器	0x3

PnT_ WR	位	描述	初始状态
	[31:3]	读未定义，写为 0	
t_ WR	[2:0]	内存时钟周期内，设置写到预充电延迟	011

5.4.16 T_WTR 寄存器

寄存器	地址	读/写	描述	复位值
P0T_WTR	0x7E00003C	读/写	16 位 DRAM 控制 t_ WTR 寄存器	0x2
P1T_ WTR	0x7E00103C	读/写	32 位 DRAM 控制 t_ WTR 寄存器	0x2

PnT_ WTR	位	描述	初始状态
	[31:3]	读未定义，写为 0	
t_ WTR	[2:0]	内存时钟周期内，设置写到读延迟	010

5.4.17 T_XP 寄存器

寄存器	地址	读/写	描述	复位值
P0T_XP	0x7E000040	读/写	16 位 DRAM 控制 t_ XP 寄存器	0x01
P1T_ XP	0x7E001040	读/写	32 位 DRAM 控制 t_ XP 寄存器	0x01

PnT_ XP	位	描述	初始状态
	[31:8]	读未定义，写为 0	
t_ XP	[7:0]	内存时钟周期内，设置退出掉电命令时间	0x01

5.4.18 T_XSR 寄存器

寄存器	地址	读/写	描述	复位值
P0T_XSR	0x7E000044	读/写	16 位 DRAM 控制 t_ XSR 寄存器	0x0A
P1T_ XSR	0x7E001044	读/写	32 位 DRAM 控制 t_ XSR 寄存器	0x0A

PnT_ XSR	位	描述	初始状态
	[31:8]	读未定义，写为 0	
t_ XSR	[7:0]	内存时钟周期内，设置退出自刷新命令时间	0x0A

5.4.19 T_ESR 寄存器

寄存器	地址	读/写	描述	复位值
P0T_ESR	0x7E000048	读/写	16 位 DRAM 控制 t_ ESR 寄存器	0x14
P1T_ ESR	0x7E001048	读/写	32 位 DRAM 控制 t_ ESR 寄存器	0x14

PnT_ ESR	位	描述	初始状态
	[31:8]	读未定义，写为 0	
t_ ESR	[7:0]	内存时钟周期内，设置自刷新命令时间	0x14

5.4.20 存储配置 2 寄存器

寄存器	地址	读/写	描述	复位值
P0MEMCFG2	0x7E00004C	读/写	16 位 DRAM 控制配置寄存器	0x0B00
P1MEMCFG2	0x7E00104C	读/写	32 位 DRAM 控制配置寄存器	0x0B40

PnMEMCFG2	位	描述	初始状态
Reserved	[31:13]	读未定义，写为 0。	
Read delay	[12:11]	当从信息包接口到允许用于传入读数据的低歪曲率时，使用编码延迟。 00=读延迟 0 周期（通常用于 SDR SDRAM）。 01=读延迟 1 周期（通常用于 DDR SDRAM 和移动 DDR SDRAM） 10，11=读延迟 2 周期。	01
Memory type	[10:8]	附加上 DRAM 控制器类型的 SDRAM： 000=SDR SDRAM 001=DDR SDRAM 011=移动的 DDR SDRAM 010=嵌入式的 SDRAM 1xx=保留	011
Memory width	[7:6]	外部存储器的宽度： 00=16 位　01=32 位　10=64 位　11=128 位	00 / 01
Bank bits	[5:4]	AXI 地址位的编码数包含的页地址： 00=2 位 01=1 位 10=0 位 11=保留	00
Reserved	[3]	读为 0，写为 0。	0
DQM init	[2]	当存储器重置时，为 DQM 状态	0
Clock config	[1:0]	时钟顺序支持： 00=AXI 时钟和内存时钟异步 01=AXI 时钟和内存时钟同步，和 AXI 时钟可以是相同的频率，或是慢于内存时钟 S3C6410 支持同步配置。如果这个值被设置为异步，则 S3C6410 性能将持续降低 10～11=保留	00

5.4.21 ID_N_CFG 寄存器

寄存器	地址	读/写	描述	复位值
P0_id_0_cfg ～P0_id_15_cfg	0x7E000100 ～0x7E00013C	读/写	16 位 DRAM 控制 id_<n>_cfg 寄存器	0x000
P1_id_0_cfg ～P1_id_15_cfg	0x7E001100 ～0x7E00113C	读/写	32 位 DRAM 控制 id_<n>_cfg 寄存器	0x000

Pn_id_<n>_cfg	位	描述	初始状态
	[31:10]	读未定义，写为 0	
QoS_MAX	[9:2]	设置一个最高质量的服务	0x00
QoS_MIN	[1]	设置一个最低质量的服务	0
QoS_Enable	[0]	使能一个服务质量的值以适用于存储器从地址 ID<n>读	0

5.4.22 CHIP_N_CFG 寄存器

寄存器	地址	读/写	描述	复位值
P0_chip_0_cfg P0_chip_1_cfg	0x7E000200 0x7E000204	读/写	16 位 DRAM 控制 chip_<n>_cfg 寄存器	0x0FF00
P1_chip_0_cfg P1_chip_1_cfg	0x7E001200 0x7E001204	读/写	32 位 DRAM 控制 chip_<n>_cfg 寄存器	0x0FF00

Pn_chip_<n>_cfg	位	描述	初始状态
	[31:17]	读未定义，写为 0	
BRC_RBC	[16]	选择存储器结构作为从 AXI 的解码地址： 0=行-页 -列结构 1=页-行-列结构	0

Address match	[15:8]	比较 AXI 地址位[31:24]的值，决定选择哪一个芯片	0xFF
Address mask	[7:0]	屏蔽 AXI 的地址位[31:24]，决定选择哪一个芯片 1=用于比较的相应地址位	0x00

5.4.23 用户身份寄存器

寄存器	地址	读/写	描述	复位值
P0_user_stat	0x7E000300	读	16 位 DRAM 控制用户身份寄存器	0x00
P1_user_stat	0x7E001300	读	32 位 DRAM 控制用户身份寄存器	0x00

Pn_user_stat	位	描述	初始状态
	[31:8]	读未定义，写为 0	
DQS[3] delay	[7:6]	显示输入 dqs[3]延迟	0
DQS[2] delay	[5:4]	显示输入 dqs[2]延迟	0
DQS[1] delay	[3:2]	显示输入 dqs[1]延迟	0
DQS[0] delay	[1:0]	显示输入 dqs[0]延迟	0

5.4.24 用户配置寄存器

寄存器	地址	读/写	描述	复位值
P0_user_cfg	0x7E000304	写	16 位 DRAM 控制用户配置寄存器。	0x00
P1_user_cfg	0x7E001304	写	32 位 DRAM 控制用户配置寄存器。	0x00

Pn_user_ cfg	位	描述	初始状态
	[31:8]	读未定义，写为 0	
DQS[3] delay	[7:6]	显示输入 dqs[3]延迟	0

DQS[2] delay	[5:4]	显示输入 dqs[2]延迟	0
DQS[1] delay	[3:2]	显示输入 dqs[1]延迟	0
DQS[0] delay	[1:0]	显示输入 dqs[0]延迟	0

5.5 DRAM 控制器 API 功能的软件实现举例

下面的程序主要是初始化一个特定 DMC 控制器：DMC0 => X16, mDDR (512Mb), DMC1=> X16*2, mDDR (1025Mb)。该部分程序中包含了一些寄存器的应用，通过分析程序可以对这些寄存器有更深的认识。

```
void DMC_Init(u8 uController, DMC_eStopCLK eStopClk, DMC_eAutoPD eAutoPD, u32 uPDprd )
{
    u32 uBaseAddress;
    u32 uPara_REF,uPara_RAS, uPara_RC, uPara_RCD, uScheduled_Para;
    u32 uPara_RFC, uPara_RP, uPara_RRD, uPara_WR, uPara_WTR, uPara_XSR;
// uController — DMC 控制器号
    if(uController == 0)
    {
        uBaseAddress = DMC0_BASE;
    }
    else if(uController == 1)
    {
        uBaseAddress = DMC1_BASE;
    }
    else
    {


    }
    g_DMCBase[uController] = (void *)uBaseAddress;

    // 获得系统时钟信息  g_HCLK
```

```
SYSC_GetClkInform();

//输入配置模式
Outp32(&DMC(uController)->rMEMCCMD, (0x4<<0));   // Memc_cmd = 0x4

//  定时参数设置
uPara_REF = (((g_HCLK /1000 *eDDR_REF) -1)/1000000 + 1);
Outp32(&DMC(uController)->rREFRESH, uPara_REF);

Outp32(&DMC(uController)->rCASLAT, (0x3<<1));  // CAS  延迟  = 3
Outp32(&DMC(uController)->rT_DQSS, 0x1);                // mDDR = 0x1 (0.75 ~ 1.25)
Outp32(&DMC(uController)->rT_MRD, 0x2);                 //  模式定时器 Cmd 定时
uPara_RAS = (((g_HCLK /1000 *eDDR_RAS) -1)/1000000 + 1);
Outp32(&DMC(uController)->rT_RAS, uPara_RAS); //  RAS(行激活时间)预加点延迟  ，(最小:45ns, 7@133MHz,  极限:1 clock)
uPara_RC = (((g_HCLK /1000 *eDDR_RC) -1)/1000000 + 1);
Outp32(&DMC(uController)->rT_RC, uPara_RC);             // RC(行周期时间)：激活 bank x  到激活  bank x 延迟  (最小:67.5ns, 10@133MHz,  极限:1clock)
uPara_RCD = (((g_HCLK /1000 *eDDR_RCD) -1)/1000000 + 1);
if (uPara_RCD <4)
      {
            uScheduled_Para = 3;
      }
else
      {
           uScheduled_Para = uPara_RCD;
      }
uPara_RCD = ((uScheduled_Para-3)<<3) | (uPara_RCD);
Outp32(&DMC(uController)->rT_RCD, uPara_RCD);       // RAS  到 CAS 延迟, (最小:22.5ns, 4@133MHz,
```