“黑色经典”系列之《ARM 嵌入式系统开发典型模块》

第 3 章　SDRAM 模块功能简介

3.1 SDRAM 模块功能简介

SDRAM 与 Flash 不同，它不具有掉电保持数据的特性，但其存取速度大大高于 Flash 存储器，且具有读/写的属性，因此 SDRAM 在系统中主要用作程序的运行空间，数据及堆栈区。当系统启动时，CPU 首先从复位地址 0x0 处读取启动代码，在完成系统的初始化后，程序代码一般应调入 SDRAM 中运行，以提高系统的运行速度，同时，系统及用户堆栈、运行数据也都放在 SDRAM 中。

SDRAM 具有单位空间存储容量大和价格便宜的优点，已广泛应用在各种嵌入式系统中。SDRAM 的存储单元可以理解为一个电容，总是倾向于放电，为避免数据丢失，必须定时刷新（充电）。因此，要在系统中使用 SDRAM，就要求微处理器具有刷新控制逻辑，或是在系统中另外加入刷新控制逻辑电路。S3C4510B 和 S3C44B0X 芯片及其他一些 ARM 芯片在片内具有独立的 SDRAM 刷新控制逻辑，可方便地与 SDRAM 接口。但某些 ARM 芯片则没有 SDRAM 刷新控制逻辑，不能直接与 SDRAM 接口，在进行系统设计时应注意这一点。

SDRAM 是高速的动态随机存取存储器，它的同步接口和完全流水线的内部结构使其拥有极大的数据速率，目前 SDRAM 时钟频率已达 100MHz 以上。另外它们的行、列地址线共用，由行地址选通（(CAS)、列地址选通（RAS）信号分时控制。基本存储单元是内存芯片中储存信息的最小的单位，每个存储单元可以存储 1bit 的信息，并且有一个由行地址和列地址共同定义的唯一的地址。我们都知道 8bit 可以在一起组成 1byte（这也就意味着 lbyte 具有 256 种可能的数值），而字节是内存中最小的可寻址的单元。虽然内存基本存储单元具有唯一的地址，但是并不能进行独立的寻址，这将要求内存芯片有数以百计的引脚同计算机通信，显然这是不可能的。现在内存架构是处于同一列的基本存储单元共用一条列地址线，而处于同一行的基本存储单元共用一条行地址线，组成一个基本存储单元构成的矩阵架构。而这些矩阵架构构成一个内存 Bank，SDRAM 内部以 Bank 为组织，可由行、列地址寻址。另外为了保持内部数据还必须进行刷新。

3.2 SDRAM 的结构特点

3.2.1 DRAM 器件的结构特点

DRAM 存储一个位的消息只需要一只晶体管，但是需要周期性地充电，才能使保存的信息不消失。DRAM 的一个存储位单元结构如图 3.1 所示。

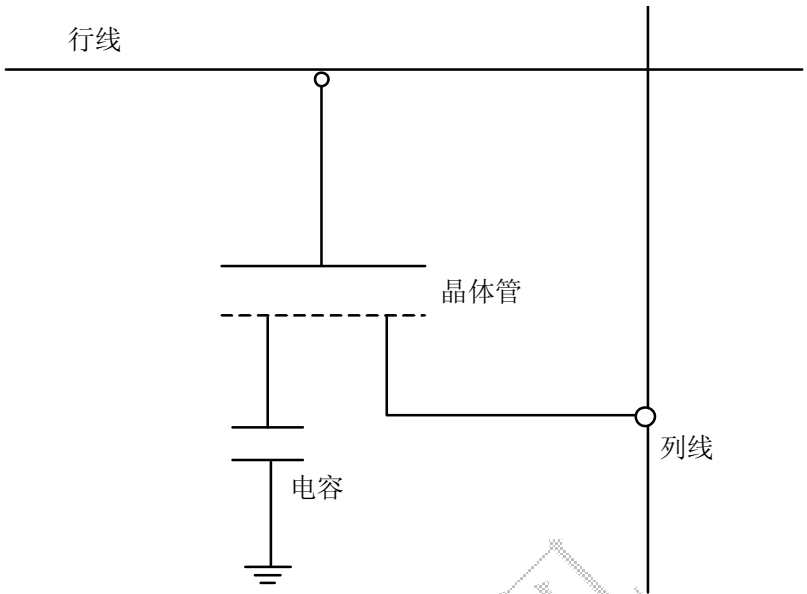

图 3.1　DRAM 的存储单元结构

上图只是 DRAM 一个基本单位的结构示意图，电容器的状态决定了这个 DRAM 单位的逻辑状态是 1 还是 0。一个电容器可以存储一定量的电子或者电荷。一个充电的电容器被认为是逻辑上的 1，而“空”的电容器则是 0。但是电容器被利用的这个特性也是它的缺点。因为电容器不能持久地保持存储的电荷，所以内存需要不断定时刷新，才能保持暂存的数据。电容器可以由电流来充电（当然这个电流是有一定限制的，否则会把电容击穿）；同时电容的充放电需要一定的时间，虽然对于内存基本单位中的电容来说这个时间很短，大约只有 0.18～0.2μs，但是这个期间内存是不能执行存取操作的。

DRAM 制造商的一些资料中显示，内存至少要每 64ms 刷新一次，这也就意味着内存有 1%的时间要用来刷新。内存的自动刷新对于内存厂商来说不是一个难题，而关键在于，当对内存单元进行读取操作时要保持内存的内容不变——所以 DRAM 单元每次读取操作之后都要进行刷新，也就是执行一次“回写”操作，以为读取操作会破坏内存中的电荷。因此，内存不但要每 64ms 刷新一次，而且每次读操作之后还要刷新一次，这样就增加了存取操作的周期。

一般 DRAM 中存储单元的内部结构如图 3.2 所示。

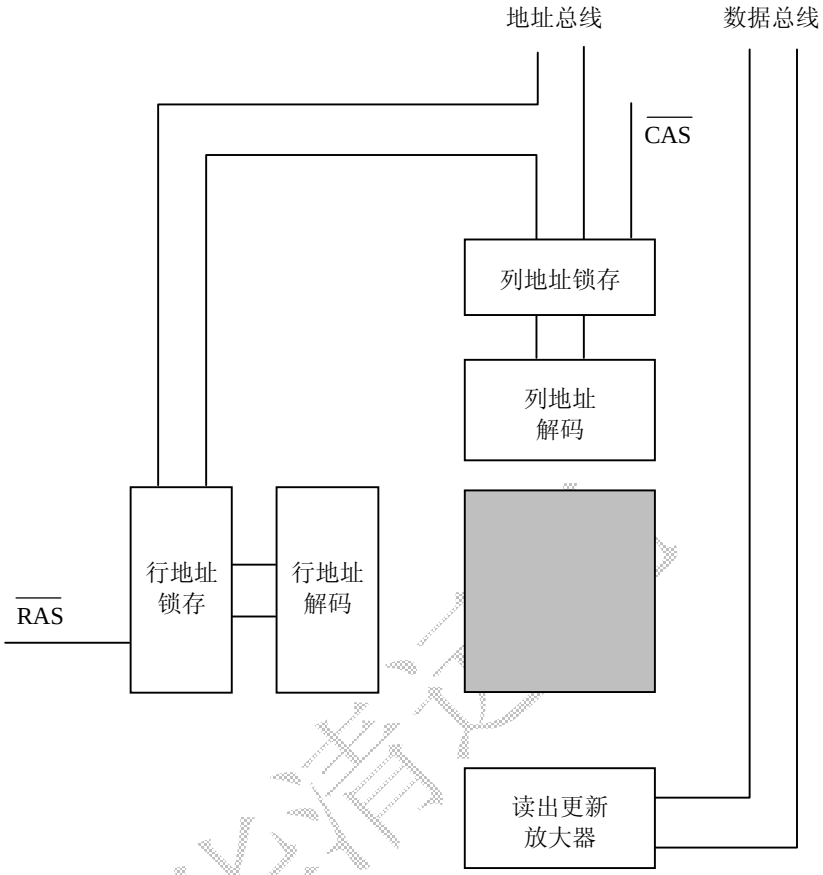

图 3.2　DRAM 内部结构示意图

DRAM 的读取过程按以下步骤进行。

（1）处理器通过地址总线将行地址传输到 DRAM 的地址引脚。

（2）$\overline{\text{RAS}}$ 引脚被激活，这样，行地址被传送到地址锁存器中。

（3）行地址解码器根据接受到的数据选择相应的行。

（4）$\overline{\text{WE}}$ 引脚被确定为不被激活，所以 DRAM 知道它不会进行写入操作。

（5）处理器通过地址总线将列地址传输到 DRAM 的地址引脚。

（6）$\overline{\text{CAS}}$ 引脚被激活，这样列地址被传送到列地址锁存器中。

（7）$\overline{\text{CAS}}$ 同样还具有 $\overline{\text{OE}}$ 引脚的功能，所以这个时候 D_{out} 引脚知道需要向外输出数据。

（8）$\overline{\text{RAS}}$ 和 $\overline{\text{CAS}}$ 都失效，这样就可以进行下一个周期的数据操作了。

DRAM 的写入过程和读取过程基本一样，这里就不再详细介绍了。

在 DRAM 读取方式中，当一个读取周期结束后，$\overline{\text{CAS}}$ 和 $\overline{\text{RAS}}$ 都必须失效，然后再进行一个回写过程才能进入到下一次的读取周期中。

3.2.2 SDRAM 器件的构成原理和应用特点

SDRAM 的存储单元的基本原理同前面提到的 DRAM 基本一样，但是这些存储位单元的组织和控制与 DRAM 就有相当大的差别了。

SDRAM 是多 Bank 结构，例如在一个具有两个 Bank 的 SDRAM 的模组中，其中一个 Bank 在进行预充电期间，另一个 Bank 却马上可以被读取，这样当进行一次读取后，又马上去读取已经预充电 Bank 的数据时，就无需等待而是可以直接读取了，这也就大大提高了存储器的访问速度。

为了实现这个功能，SDRAM 需要增加对多个 Bank 的管理，实现控制其中的 Bank 进行预充电。在一个具有 2 个以上 Bank 的 SDRAM 中，一般会多一根叫做 BAn 的引脚，用来实现在多个 Bank 之间的选择。

3.3 SDRAM 的内部操作

SDRAM 具有多种工作模式，内部操作是一个复杂的状态机。SDRAM 器件的引脚分为以下几类。

（1）控制信号：包括片选、时钟、时钟使能、行列地址选择、读写有效及数据有效。

（2）地址信号：时分复用引脚，根据行列地址选择引脚，控制输入的地址为行地址或列地址。

（3）数据信号：双向引脚，受数据有效控制。

SDRAM 的所有操作都同步于时钟。根据时钟上升沿控制管脚和地址输入的状态，可以产生多种输入命令。

- 模式寄存器设置命令。
- 激活命令。
- 预充命令。
- 读命令。
- 写命令。
- 带预充的读命令。
- 带预充的写命令。
- 自动刷新命令。
- 自我刷新命令。
- 突发停命令。
- 空操作命令。

根据输入命令，SDRAM 状态在内部状态间转移。内部状态包括模式寄存器设置状态、激活状态、预充状态、写状态、读状态、预充读状态、预充写状态、自动刷新状态及自我刷

新状态。下面以三星公司生产的 32M×8 位 SDRAM 器件 K4S560832A 为例介绍 SDRAM 的内部操作。

K4S560832A 支持的操作命令有初始化配置、预充电、行激活、读操作、写操作、自动刷新、自刷新等。所有的操作命令通过控制线 CS#、RAS#、CAS#、WE#和地址线 A12～A0、体选地址 BA 输入。K4S560832A 的命令及命令码如表 3.1 所示。各命令执行后 K4S560832A 的内部控制器状态转换如图 3.3 所示。

表 3.1　　K4S560832A 命令码

当前状态	CS#	RAS#	CAS#	WE#	BA	地址	命令助记符	功能
空闲	H	×	×	×	×	×	NOP	空操作
	L	H	H	H	×	×	NOP	空操作
	L	L	H	H	BA	RA	ACT	行/体激活；行地址锁存
	L	L	H	L	BA	A10/AP	NOP	空操作
	L	L	L	H	×	×	REFA	自动刷新或自刷新
	L	L	L	L	操作码	操作码	MRS	模式寄存器设置
行激活	H	×	×	×	×	×	NOP	空操作
	L	H	H	H	×	×	NOP	空操作
	L	H	L	H	BA	CA,10/AP	READ	开始读；锁定列地址；确定 AP
	L	H	L	L	BA	CA,10/AP	WRITE	开始写；锁定列地址；确定 AP
	L	L	H	L	BA	A10/AP	PRE	预充电
读/写	H	×	×	×	×	×	NOP	继续猝发到结束，进入行激活
	L	H	H	H	×	×	NOP	继续猝发到结束，进入行激活
	L	H	H	L	×	×	BURST STOP	结束猝发，进入行激活状态
	L	H	L	H	BA	CA,10/AP	READ	结束猝发；开始新的读，确定 AP
	L	H	L	L	BA	CA,10/AP	WRITEA	结束猝发；开始新的写，确定 AP
	L	L	H	L	BA	A10/AP	PRE	结束猝发；为准备读/写做预充电

带预充电读/写	H	×	×	×	×	×	NOP	继续猝发到结束，进入预充电
	L	H	H	H	×	×	NOP	继续猝发到结束，进入预充电
预充电	H	×	×	×	×	×	NOP	延时 T_{RP} 后进入空闲
	L	H	H	H	×	×	NOP	延时 T_{RP} 后进入空闲
	L	L	H	L	BA	A10/AP	NOP	延时 T_{RP} 后进入空闲
激活行	H	×	×	×	×	×	NOP	延时 T_{RCD} 后进入行激活
	L	H	H	H	×	×	NOP	延时 T_{RCD} 后进入行激活
刷新	H	×	×	×	×	×	NOP	延时 T_{RC} 后进入空闲
	L	H	H	×	×	×	NOP	延时 T_{RC} 后进入空闲
模式设置	H	×	×	×	×	×	NOP	延时 2 个时钟节拍后进入空闲
	L	H	H	H	×	×	NOP	延时 2 个时钟节拍后进入空闲

K4S560832A 内部控制器状态转换图如图 3.3 所示。

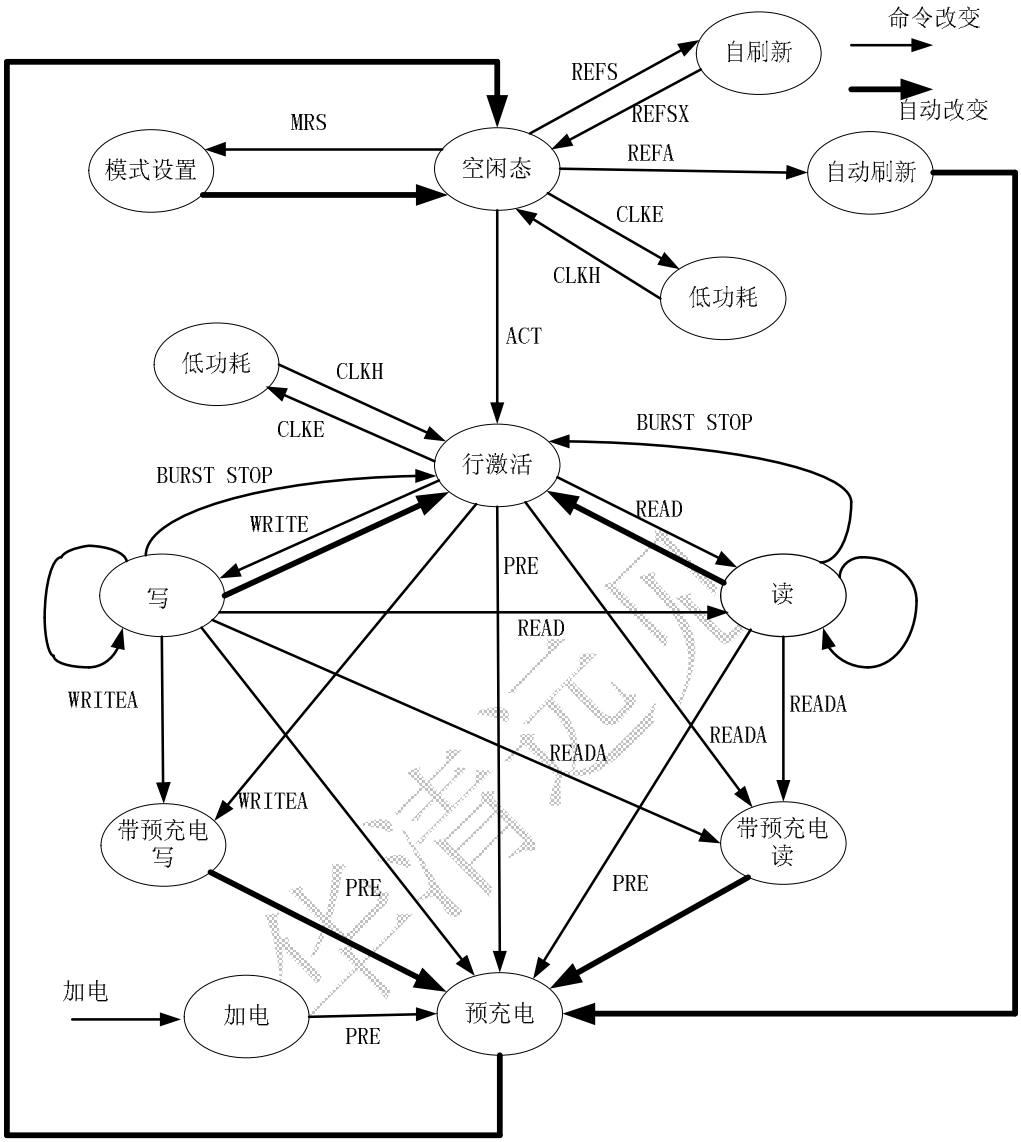

图 3.3　K4S560832A 内部控制器状态转换图

K4S560832A 各操作状态下只能输入表中所列的命令，其余命令均为非法命令（限制输入，否则将导致内部状态机出错）或无效命令（可以输入但不起作用），其中空操作命令 NOP 不指示任何新的操作，仅仅用作命令之间的间隔。

1．行激活

行激活命令选择处于空闲状态存储体的任意一个行，使之进入准备读/写状态。从体激活到允许输入读/写命令的间隔时钟节拍数取决于内部特征延时和时钟频率。K4S560832A 内部

有 4 个体，为了减少器件门数，4 个体之间的部分电路是公用的，因此它们不能同时被激活，而且从一个体的激活过渡到另一个体的激活也必须保证有一定的时间间隔。

2．预充电

预充电命令用于对已激活的行进行预充电即结束活动状态。预充电命令可以作用于单个体，也可以同时作用于所有体（通过所有体预充电命令）。对于猝发写操作必须保证在写入预充电命令前写操作已经完成，并使用 DQM 禁止继续写入数据。预充电结束后回到空闲状态，也可以再次被激活，此时也可以输入进入低功耗、自动刷新、自刷新和模式设置等操作命令。

3．自动预充电

如果在猝发读或猝发写命令中，A10/AP 位置为“1”，在猝发读写操作完成后自动附加一个预充电动作。操作行结束活动状态，但在内部状态机回到空闲态之前不能给器件发送新的操作命令。

4．猝发读

猝发读命令允许某个体中的一行被激活后，连续读出若干个数据。第一个数据在经过指定的 CAS 延时节拍后呈现在数据线上，以后每个时钟节拍都会读出一个新的数据。猝发读的行内初始列地址可以任意，不要求对界，但地址的变化范围被限制在 A1～A0 或 A2～A0，从“0”开始的 4 或 8 个字节内。猝发读操作结束后数据总线呈高阻态，可以在适当的节拍上再次发出猝发读命令使数据线上保持没有空隙。猝发读操作可以被同体或不同体的新的猝发读/写命令或同一体的预充电命令及猝发停止命令中止。

5．猝发写

猝发写命令与猝发读命令类似，允许某个体中的一行被激活后，连续写入若干个数据。第一个写数据与猝发写命令同时在数据线上给出，以后每个时钟节拍给出一个新的数据，输入缓冲在猝发数据量满足要求后停止接受数据。猝发写操作可以被猝发读/写命令或 DQM 数据输入屏蔽命令和预充电命令或猝发停止命令中止。

6．自动刷新

由于动态存储器存储单元存在漏电现象，为了保持每个存储单元数据的正确性，K4S560832A 必须保证在 64ms 内对所有的存储单元刷新一遍。一个自动刷新周期只能刷新存储单元的一个行，每次刷新操作后内部刷新地址计数器自动加“1”。只有在所有体都空闲（因为 4 个体的对应行同时刷新）并且未处于低功耗模式时才能启动自动刷新操作，刷新操作执行期间只能输入空操作，刷新操作执行完毕后所有体都进入空闲状态。该器件可以每间隔 7.8μs 执行一次自动刷新命令，也可以在 64ms 内的某个时间段对所有单元集中刷新一遍。

7．自刷新

自刷新是动态存储器的另一种刷新方式，通常用于在低功耗模式下保持 SDRAM 的数

据。在自刷新方式下，SDRAM 禁止所有的内部时钟和输入缓冲（CKE 除外）。为了降低功耗，刷新地址和刷新时间全部由器件内部产生。一旦进入自刷新方式只有通过 CKE 变低才能激活，其他的任何输入都将不起作用。给出退出自刷新方式命令后必须保持一定节拍的空操作输入，以保证器件完成从自刷新方式的退出。如果在正常工作期间采用集中式自动刷新方式，则在退出自刷新模式后必须进行一遍（8192 个）集中的自动刷新操作。

8．时钟和时钟屏蔽

时钟信号是所有操作的同步信号，上升沿有效。时钟屏蔽信号 CKE 决定是否把时钟输入施加到内部电路。在读写操作期间，CKE 变低后的下一个节拍冻结输出状态和猝发地址，直到 CKE 变高为止。在所有的体都处于空闲状态时，CKE 变低后的下一个节拍 SDRAM 进入低功耗模式并一直保持到 CKE 变高为止。

9．DQM 操作

DQM 用于屏蔽输入输出操作，对于输出相当于开门信号，对于输入禁止把总线上的数据写入存储单元。对读操作 DQM 延迟 2 个时钟周期开始起作用，对写操作则是当拍有效。

3.4 常见的 SDRAM 器件简介

目前常用的 SDRAM 为 8 位/16 位的数据宽度，工作电压一般为 3.3V，主要的生产厂商为 HYUNDAI、Winbond 等。他们生产的同型器件一般具有相同的电气特性和封装形式，可通用。

下面主要以 HYUNGAU 公司的 HY57V641620 芯片为例介绍一下 SDRAM 芯片的基本特性及使用方法。

HYUNGAU 公司的 HY57V641620。HY57V641620 的存储容量为 4 组×16M 位（8MB），工作电压为 3.3V，常见封装为 54 脚 TSOP，兼容 LVTTL 接口，支持自动刷新（Auto-Refresh）和自刷新（Self-Refresh），16 位数据宽度。

HY57V641620 引脚分布及信号描述分别如图 3.4 和表 3.2 所示。

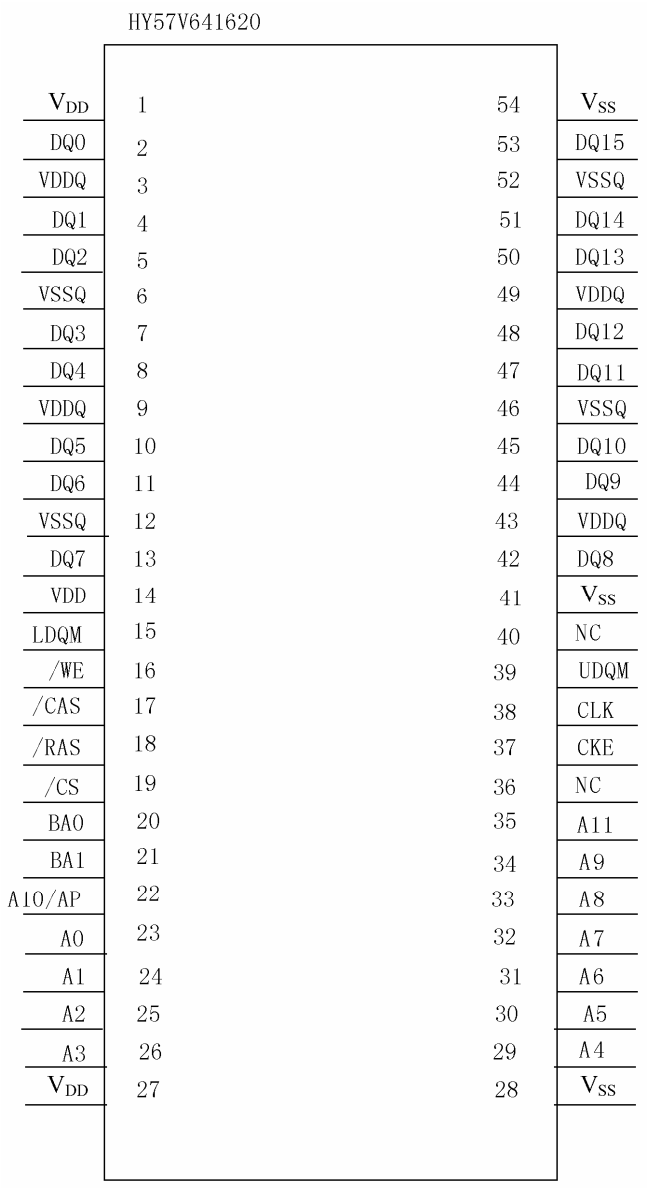

图 3.4　HY57V641620 引脚分布图

HY57V641620 引脚信号描述如表 3.2 所示。

表 3.2　　HY57V641620 引脚信号描述

引脚	名称	描述
CLK	时钟	芯片时钟输入
CKE	时钟使能	片内时钟信号控制
/CS	片选	禁止或使能 CLK、CKE 和 DQM 外的所有输入信号

BA0,BA1	组地址选择	用于片内 4 个组的选择
A11～A0	地址总线	行地址：A11～A0，列地址：A7～A0，自动预充电标志：A10
/RAS /CAS /WE	行地址锁存 列地址锁存 写使能	行、列地址锁存和写使能信号引脚
LDQM，UDQM	数据 I/O 屏蔽	在读模式下控制输出缓冲；在写模式下屏蔽输入数据
DQ15～DQ0	数据总线	数据输入输出引脚
VDD/V_{SS}	电源/地	内部电路及输入缓冲电源/地
VDDQ/VSSQ	电源/地	输出缓冲电源/地
NC	未连接	未连接

以上为一款常见的 SDRAM 芯片 HY57V641620 的简介，更具体的内容可参考 HY57V641620 的用户手册。

3.5 SDRAM 的硬件设计

3.5.1 SDRAM 的接口电路

下面以 S3C44B0X 芯片和 HY57V641620 芯片为例说明 SDRAM 存储器实际接口电路。

S3C44BOX 芯片的 SDRAM 接口引脚如图 3.5 所示，引脚描述如下。

nSRAS[1:0]：SDRAM 行地址选通信号。

nSCAS[3:0]：SDRAM 列地址选通信号。

nSCS[1:0]：DRAM 芯片选择信号。

DQM[3:0]：SDRAM 数据屏蔽信号。

SCLK：SDRAM 时钟信号。

SDLK：SDRAM 时钟允许信号。

其中，各主要信号定义如下。

- 数据总线：DATA0～DATA15 分别连接 CPU 的数据总线 DATA0～DATA15，表示 16 位数据宽度。
- 地址总线：A0～A11 分别连接 CPU 的地址总线 ADDRl～ADDR12。
- 片选信号：SDRAM 的 19 脚为片选信号，本系统中采用 nSCSO（即 nGCS6）来进行控制，表示其映射后的起始地址为 0x0C00_0000。

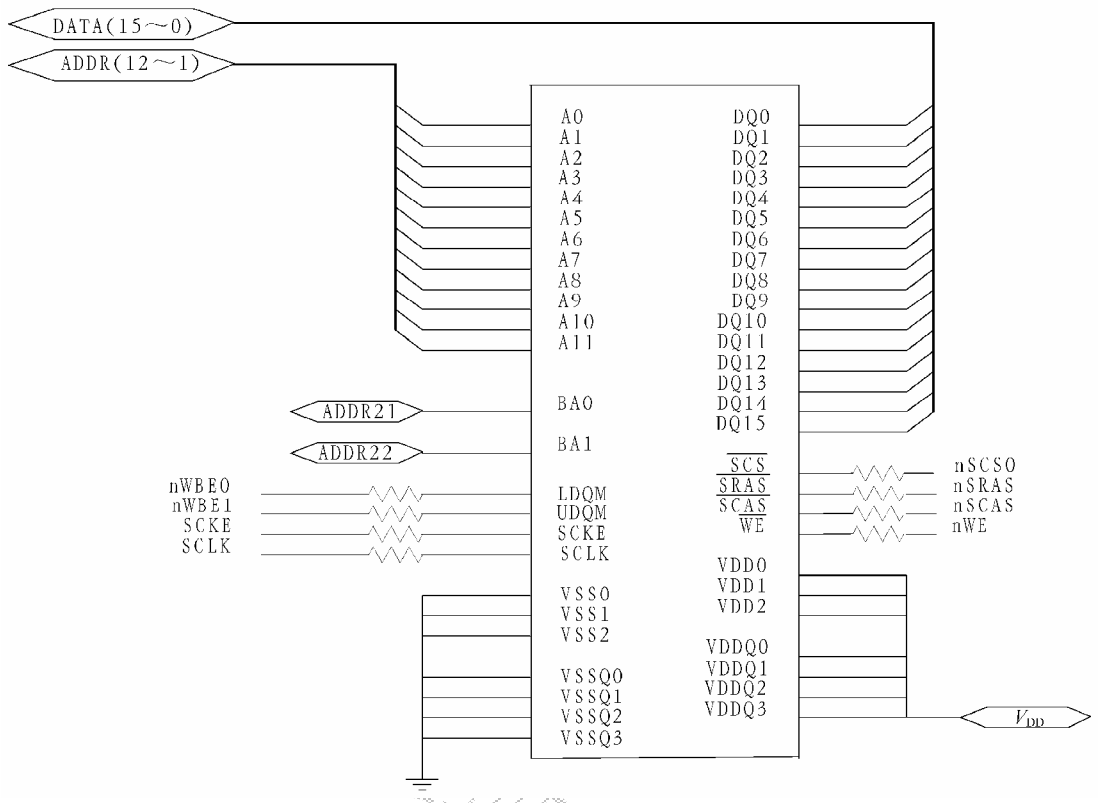

图 3.5　SDRAM 的接口电路图

- 时钟信号：时钟使能信号接 CPU 的外部时钟使能信号 SCKE，时钟信号由 CPU 的 SCLK 引脚提供。
- 字节写允许信号：LDQM 和 UDQM 信号表示 SDRAM 字节写允许信号，分别由 CPU 的字节写允许信号引脚 nWBE [1,0]控制。
- 行/列选通信号：nSRAS 和 nSCAS 分别为行、列地址选通信号，分别由 CPU 的 nSRAS 和 nSCAS 提供。

3.5.2　用 16 位 SDRAM 芯片构成 32 位存储系统

根据系统需求，可构建 16 位或 32 位的 SDRAM 存储器系统，但为充分发挥 32 位 CPU 的数据处理能力，大多数系统采用 32 位的 SDRAM 存储器系统。下面以 HY57V641620 芯片为例介绍用 2 片 16 位 SDRAM 芯片构成 32SDRAM 存储系统与 S3C4510B 相连接。

HY57V641620 为 16 位数据宽度，单片容量为 8MB，选用的 2 片 HY57V641620 并联构建 32 位的 SDRAM 存储器系统，共 16MB 的 SDRAM 空间，可满足嵌入式操作系统及各种相对较复杂的算法的运行要求。具体步骤如下。

（1）2 片 HY57V641620 并联构建 32 位的 SDRAM 存储器系统，其中一片为高 16 位，另一片为低 16 位。可将 2 片 HY57V641620 作为一个整体配置到 DRAM/SDRAM Bank0～DRAM/SDRAM Bank3 的任一位置，一般配置到 DRAM/SDRAM Bank0，即将 S3C4510B 的 nSDCS<0>（Pin89）接至 2 片 HY57V641620 的/CS 端。

（2）2 片 HY57V641620 的 CLK 端接 S3C4510B 的 SDCLK 端（Pin77）。

（3）2 片 HY57V641620 的 CLE 端接 S3C4510B 的 CLE 端（Pin97）。

（4）2 片 HY57V641620 的/RAS、/CAS、/WE 端分别接 S3C4510B 的 nSDRAS 端（Pin95）、nSDCAS 端（Pin96）、nDWE 端（Pin99）。

（5）2 片 HY57V641620 的 A11～A0 接 S3C4510B 的地址总线 ADDR11～ADDR0。

（6）2 片 HY57V641620 的 BA1、BA0 接 S3C4510B 的地址总线 ADDR13、ADDR12；

（7）高 16 位片的 DQ15～DQ0 接 S3C4510B 的数据总线的高 16 位 XDATA31～XDATA16，低 16 位片的 DQ15～DQ0 接 S3C4510B 的数据总线的低 16 位 XDATA15～XDATA0。

（8）高 16 位片的 UDQM、LDQM 分别接 S3C4510B 的 nWEB3、nWEB2，低 16 位片 UDQM、LDQM 分别接 S3C4510B 的 nWEB1、nWEB0。

图 3.6 为 32 位 SDRAM 存储器系统的实际应用电路图。

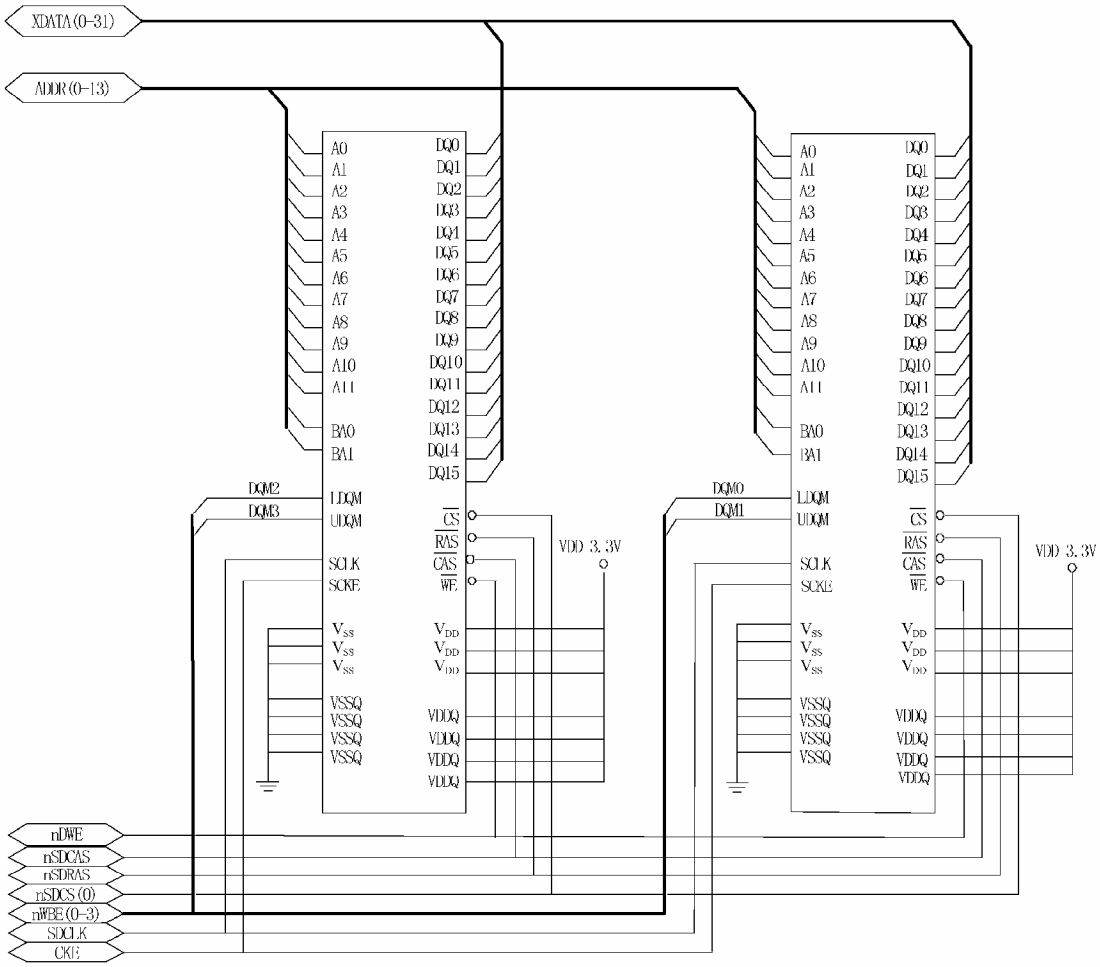

图 3.6　32 位 SDRAM 存储器系统的实际应用电路图

3.6 SDRAM 存储器软件设置

下面以基于 S3C44B0X 的系统为例介绍 SDRAM 存储器的软件设计。由于 S3C44BOX 内部存储器控制器件部件提供了专门与 SDRAM 器件接口的控制信号，因此，对 SDRAM 的读和写都不需要特殊编程操作。只需要根据所选择的存储器件的特性，在系统初始化时对与 S3C44B0X 相关的寄存器进行设置就可以了。

3.6.1 地址分配

在 ARM 系统中，如果将 SDRAM 映射在 Bank6，那么 SDRAM 的地址分配如表 3.3 所示。

表 3.3 SDRAM 地址分配表

Bank6	开始地址	结束地址	大小
	0xC000000	0xC7FFFFF	8MB

S3C44B0X 共有 8 个存储器 Bank，其中只有 Bank6 和 Bank7 可以同时作为 ROM，SRAM，FP/EDO/SDRAM 等类型存储器的存储空间。S3C44B0X 支持异步或同步 DRAM，并在对 DRAM/SDRAM 的接口中，支持对器件的自刷新模式。

S3C44B0X 对与之接口的同步动态 RAM（SDRAM）特性要求如下。

（1）SDRAM 的最大列地址为 10 位。

（2）$\overline{\text{CAS}}$ 延迟为 2 或 3 个周期。

SDRAM 的 Bank 选择线 BAn 与 S3C44B0X 地址线的对应关系，依照表 3.4 选择连接。

表 3.4 SDRAM Bank 地址配置表

Bank 大小	总线宽度	单个器件	存储器空间配置	Bank 地址
2MB	×8	16MB	(1MB×8×2Bank)×1	A20
	×16		(512KB×16×2Bank)×1	
4MB	×8	16MB	(2MB×4×2Bank)×2	A21
	×16		(1MB×8×2Bank)×2	
	×32		(512KB×16×2Bank)×2	
8MB	×16	16MB	(2MB×4×2Bank)×4	A22
	×32		(1MB×8×2Bank)×4	
	×8	64MB	(4MB×8×2Bank)×1	
	×8		(2MB×8×4Bank)×1	A22～21
	×16		(2MB×16×2Bank)×1	A22

	×16		(1MB×16×4Bank)×1	A22～21
	×32		(512KB×32×4Bank)×1	

续表

Bank 大小	总线宽度	单个器件	存储器空间配置	Bank 地址
16MB	×32	16MB	(2MB×4×2Bank)×8	A23
	×8	64MB	(8MB×4×2Bank)×2	
	×8		(4MB×4×4Bank)×2	A23～22
	×16		(4MB×8×2Bank)×2	A23
	×16		(2MB×8×4Bank)×2	A23～22
	×32		(2MB×16×2Bank)×2	A23
	×32		(1MB×16×4Bank)×2	
	×8	128MB	(4MB×8×4Bank)×1	A23～22
	×16		(2MB×16×4Bank)×1	
32MB	×16	64MB	(8MB×4×2Bank)×4	A24
	×16		(4MB×4×4Bank)×4	A24～23
	×32		(4MB×8×2Bank)×4	A24
	×32		(2MB×8×4Bank)×4	
	×16	128MB	(4MB×8×4Bank)×2	
	×32		(2MB×16×4Bank)×2	A24～23
	×8	256MB	(8MB×8×4Bank)×1	
	×16		(4MB×16×4Bank)×1	

3.6.2 寄存器设置

在系统进入 C 语言之前，需要对 S3C44B0X 的存储器控制器进行初始化。其中对与 SDRAM（Bank6）相关的寄存器进行了特殊的设置，以使 SDRAM 能够正常工作。由于 C 语言程序使用的数据空间和堆栈空间都定位在 SDRAM 上，因此，如果没有对 SDRAM（Bank6）的正确初始化，系统就无法正确启动。下面介绍与 SDRAM 相关的寄存器设置。

1．BWSCON 寄存器

BWSCON 寄存器主要用来设置外接存储器的总线宽度和等待状态。在 BWSCON 中，除了 Bank0，其他 7 个 bank 都各对应 4 个相关位的设置，分别为 STn，WSn 和 DWn。这里只需要对 DWn 进行设置，例如 SDRAM（Bank6）采用 16 位总线宽度，因此，DW6=01，其他 2 位采用缺省值。BWSCON 寄存器在 Bank6 上的位定义如表 3.5 所示。

表 3.5　　BWSCON 寄存器在 Bank6 上的位定义

BWSCON	位	描述	初始化状态
ST6	27	这个位决定 SRAM 在 Bank6 上是否采用 UB/LB 0：不采 UB/LB（引脚 14～11 表示为 nWBE3～0） 1：采 UB/LB（引脚 14～11 表示为 nWBE3～0）	0

续表

BWSCON	位	描述	初始化状态
WS6	26	这个位决定 Bank6 的 WAIT 状态 0：WAIT 禁止 1：WAIT 使能	0
DW6	25～24	这 2 位决定 Bank6 的数据总线宽度 00：8 位　01：16 位　10：32 位	0

2．BANKCONn 寄存器的设置

S3C44B0X 有 8 个 BANKCONn 寄存器，分别对应着 Bank0～Bank7。由于 Bank6～Bank7 可以作为 FP/EDO/SDRAM 等类型存储器的映射空间，因此与其他 bank 的相应寄存器有所不同，其中 MT 位定义了存储器的类型。BANKCONn 寄存器在 Bank6 和 Bank7 上的位定义如表 3.6 所示。

表 3.6　　BANKCONn 寄存器在 Bank6 和 Bank7 上的位定义

BANKCONn	位	描述	起始状态
MT	16～15	这 2 位决定了 Bank6 和 Bank7 的存储器类型 00：ROM 或 SRAM　　01：FP　DRAM 10：EDO　DRAM　　11：SDRAM	11

MT 的取值又定义该寄存器余下几位的作用。当 MT=11（即 SDRAM 型存储器）时，BANKCONn 寄存器余下的几位定义如表 3.7 所示。

表 3.7　　BANKCONn 寄存器在 MT=11 时的相关位定义

BANKCONn	位	描述	起始状态
t_{rcd}	3～2	$\overline{RAS}$ 到 $\overline{CAS}$ 的延时 00：2 时钟 01：3 时钟 10：4 时钟	10
SCAN	1～0	列地址位数 00：8 位　01：9 位　10：10 位	00

t_{rcd} 是从行使能到列使能的延迟，取 00。SCAN 为列地址线数量。

3．REFRESH 寄存器

REFRESH 寄存器是 DRAM/SDRAM 的刷新控制器。位定义如表 3.8 所示。

表 3.8　　REFRESH 寄存器位定义

REFRESH	位	描述	初始状态
REFEN	23	DRAM/SDRAM 刷新使能 0：禁止　1：使能（自刷新或 CBR/自动刷新）	1
TREFMD	22	DRAM/SDRAM 刷新模式 0：CBR/自动刷新　1：自刷新 在自刷新时，DRAM/SDRAM 控制线需要适当的电平驱动	0

续表

REFRESH	位	描述	初始状态
t_{rp}	21～20	DRAM/SDRAM $\overline{RAS}$ 预充电时间 DRAM 00：1．5 时钟　01：2．5 时钟　10：3．5 时钟 11：4．5 时钟 SDRAM 00：2 时钟　01：3 时钟　10：4 时钟　11 不支持	10
t_{rc}	19～18	SDRAM RC 最小时间 00：4 时钟　01：5 时钟　1：06 时钟　11：7 时钟	11
t_{chr}	17～16	$\overline{CAS}$ 保持时间（DRAM） 00：1 时钟　01：2 时钟　10：3 时钟　11：4 时钟	00
保留	15～12	未用	0000
刷新记数器	10～0	DRAM/SDRAM 刷新记数值	0

4．BANKSIZE 和 MRSR 寄存器

表 3.9　　BANKSIZE 寄存器定义

BANKSIZE	位	描述	初始状态
SCLKEN	4	SCLK 只在 SDRAM 被操作时产生，该特性用于减小功耗 0：普通 SCLK　1：减小功耗	0
保留	3	未用	0
BK76MAP	2～0	Bank6～Bank7 存储空间分布 000：32MB/32MB　100：2MB/2MB　101：4MB/4MB 110：8MB/8MB　111：16MB/16MB	000

MRSR 寄存器有 2 个，分别对应 MRSRB6 和 MRSRB7，对应着 Bank6 和 Bank7。见表 3.10。

表 3.10　　MRSRn 寄存器定义

MSR	位	描述	初始状态
保留	11～10	未用	—
WBL	9	触发值长度 0 为推荐设置值	X
TM	8～7	测试模式 00：寄存器模式设置 01，10，11：保留	XX
CL	6～4	$\overline{\text{CAS}}$ 延迟 000：1 时钟，010：2 时钟，011：3 时钟 其他=保留	XXX
BT	3	触发类型 0：连续的（推荐） 1：不用	X

续表

MSR	位	描述	初始状态
BL	2～0	触发长度 000：1 其他：不用	XXX

X 为任意二进制值。

3.7　应用程序设计

```
#define DebugOut Uart_Printf
#define min(x1,x2) ((x1<x2)? x1:x2)
#define max(x1,x2) ((x1>x2)? x1:x2)
#define ONESEC0 (62500)  //16us resolution, max 1.04 sec
#define ONESEC1 (31250)  //32us resolution, max 2.09 sec
#define ONESEC2 (15625)  //64us resolution, max 4.19 sec
#define ONESEC3 (7812)   //128us resolution, max 8.38 sec
#define ONESEC4 (MCLK/128/(0xff+1))   //@60Mhz, 128*4us resolution, max 32.53 sec
#define NULL 0
```

```
#define EnterPWDN(clkcon) ((void (*)(int))0xe0)(clkcon)
#define DOWNLOAD_ADDRESS _RAM_STARTADDRESS
/*44blib.c*/
void Delay(int time); //Watchdog Timer is used.
void *malloc(unsigned nbyte);
void free(void *pt);
void Port_Init(void);
void Cache_Flush(void);
void ChangeMemCon(unsigned *pMemCfg);
void Uart_Select(int ch);
void Uart_TxEmpty(int ch);
void Uart_Init(int mclk,int baud);
char Uart_Getch(void);
char Uart_GetKey(void);
int    Uart_GetIntNum(void);
void Uart_SendByte(int data);
void Uart_Printf(char *fmt,...);
void Uart_SendString(char *pt);
void Timer_Start(int divider); //应用看门狗定时器
int Timer_Stop(void);                  //应用看门狗定时器
//void restart(void);
//void run(void);
void Led_Display(int LedStatus);
void Beep(int BeepStatus);
void ChangePllValue(int m,int p,int s);
/******************Option.h******************/
#ifndef __OPTION_H__
#define __OPTION_H__
// ************* OPTIONS **************
#define MCLK 64000000
#define WRBUFOPT (0x8)     //写使能
#define SYSCFG_0KB (0x0|WRBUFOPT)
#define SYSCFG_4KB (0x2|WRBUFOPT)
#define SYSCFG_8KB (0x6|WRBUFOPT)
#define DRAM         1                    //使用 DRAM
#define SDRAM        2                    //使用  SDRAM
#define BDRAMTYPE     SDRAM       //在  power.c,44blib.c 中使用
```

```
//BUSWIDTH; 16,32
#define BUSWIDTH        (16)
#define CACHECFG        SYSCFG_8KB
#define _RAM_STARTADDRESS 0xc000000
#define _ISR_STARTADDRESS 0xc7fff00         //GCS6:64M DRAM/SDRAM
#define Non_Cache_Start     (0x2000000)
#define Non_Cache_End       (0xc000000)
// NOTE: rom.mak,option.a have to be changed
#endif /*__OPTION_H__*/
/*************Main.C*************/
#include <string.h>
#include <stdio.h>
#include "Target\44blib.h"
#define    WR_sdram(addr,dat)      *((volatile unsigned short *)(addr))=(unsigned short)dat
#define    RD_sdram(addr)                  *((volatile unsigned short *)(addr))
#define    BEGINADDR      0x0c400000
/*******************************************/
//     ARMSYS 实验三：SDRAM  读写测试
//     描述：对 SDRAM 进行操作
/*******************************************/
void Main(void)
{
      char aa;
      int i,j,k;
      Port_Init();
      Uart_Init(0,115200);
      Led_Display(0xf);
      Delay(0);
      Beep(0x1);
      Uart_Select(0); //Select UART0//
      Uart_Printf("\n\n*********************************************************************");
      Beep(0x0);
      Uart_Printf("\n*----Begin to Start SDRAM test,OK? (Y/N)-------------------*");
      Led_Display(0x0);
      aa= Uart_Getch();
      if((aa= ='Y')||(aa= ='y'))
      {
```

```
        Uart_Printf("\nBegin to write 0xc400000--0xc4000ff with 0xaa55...");
        j=BEGINADDR;
        for(i=0;i<255;i++)
        {
            WR_sdram(j,0xaa55);
            j+=2;
        }
        j=BEGINADDR;
        k=0;
        for(i=0;i<255;i++)
        {
            if((RD_sdram(j))= =0xaa55)
                k++;
            j+=2;
        }
        if(k= =255)
            Uart_Printf("\nOK, Successfully!");
        else
            Uart_Printf("\nFailed!");
    }
}
```