S3C2440 与 SDRAM 的地址连线分析

S3C2440 有 27 根地址线 ADDR[26:0]，8 根片选信号 ngcs0-ngcs7,对应 bank0-bank7，当访问 bankx 的地址空间，ngcsx 引脚为低电平，选中外设。

2^27=2^7 * 2^10 * 2^10 = 128Mbyte

8*128Mbyte = 1Gbyte

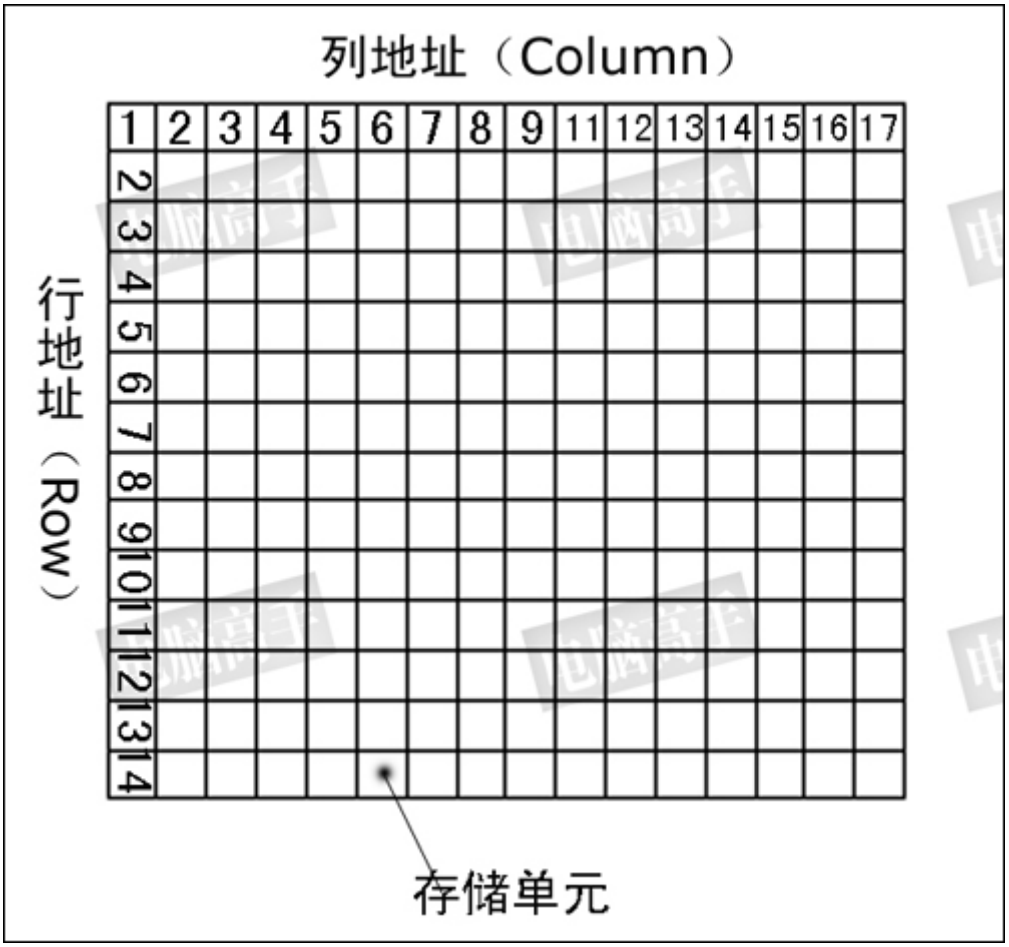
所以 S3C2440 总的寻址空间是 1Gbyte。

市面上很少有 32 位宽度的单片 SDRAM，一般选择 2 片 16 位 SDRAM 扩展得到 32 位 SDRAM.

选择的 SDARM 是 HY57V561620F，4Mbit * 4bank *16，共 32Mbyte。

首先了解下 SDRAM 的寻址原理。

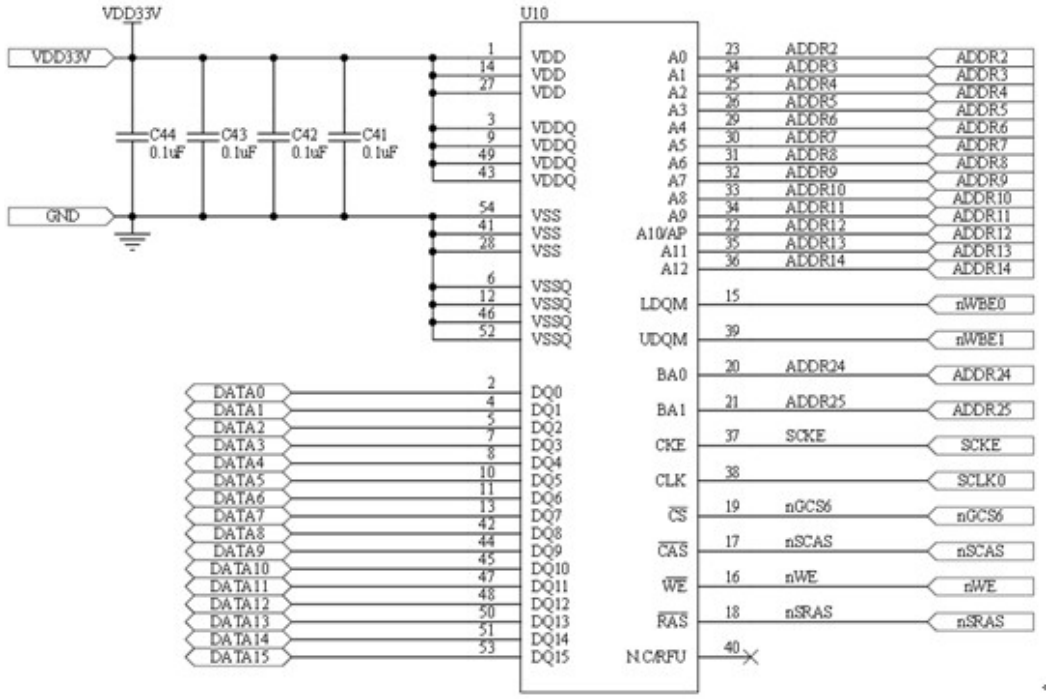
SDRAM 内部是一个存储阵列。可以把它想象成一个表格。和表格的检索原理一样，先指定行，再指定列，就可以准确找到所需要的存储单元。这个表格称为逻辑 BANK。目前的 SDRAM 基本都是 4 个 BANK。寻址的流程就是先指定 BANK 地址，再指定行地址，最后指定列地址。这就是 SDRAM 的寻址原理。存储阵列示意图如下：

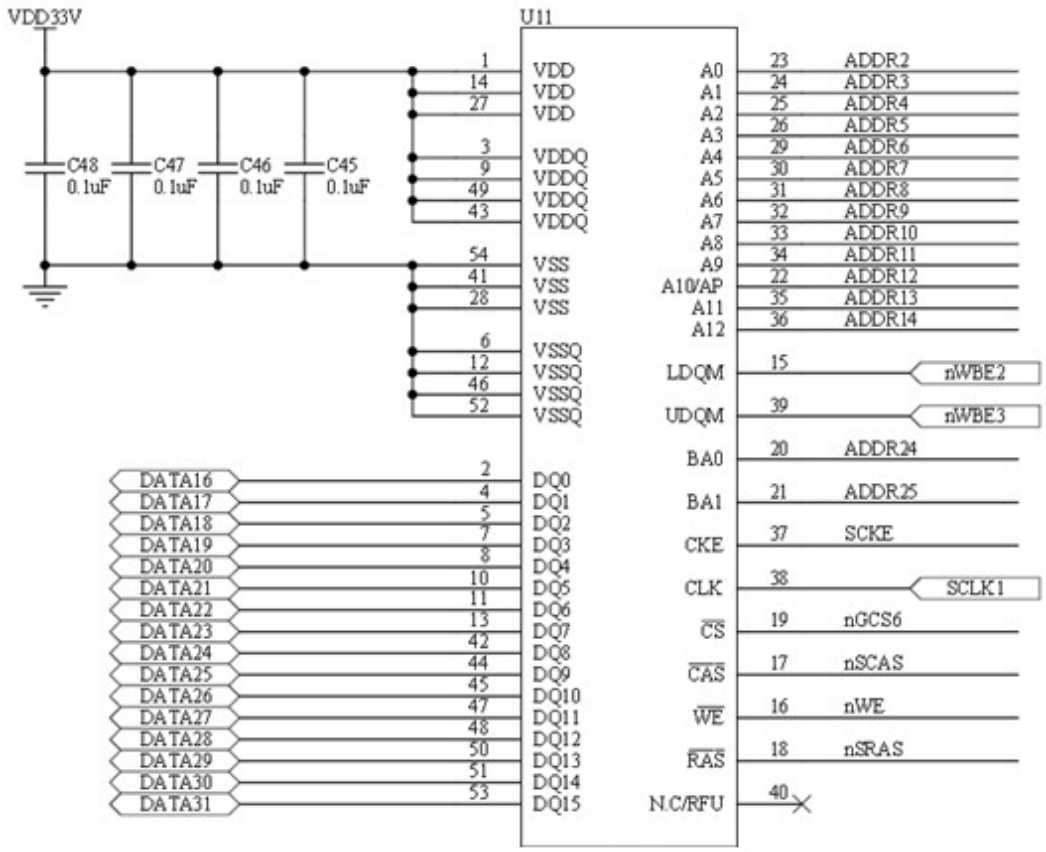


查看 HY57V561620F 的资料，这个 SDRAM 有
13 根行地址线　　RA0-RA12
9 根列地址线　　CA0-CA8
2 根 BANK 选择线　BA0-BA1

SDRAM 的地址引脚是复用的，在读写 SDRAM 存储单元时，操作过程是将读写的地址分两次输入到芯片中，每一次都由同一组地址线输入。两次送到芯片上去的地址分别称为行地址和列地址。它们被锁存到芯片内部的行地址锁存器和列地址锁存器。/RAS 是行地址锁存信号，该信号将行地址锁存在芯片内部的行地址锁存器中；/CAS 是列地址锁存信号，该信号将列地址锁存在芯片内部的列地址锁存器中。

地址连线如下图：





SDRAM 的 A0 接 S3C2440 的 ADDR2,很多初学者都对这里又疑问。A0 为什么不接 ADDR0?

要理解这种接法，首先要清楚在 CPU 的寻址空间中，字节（8 位）是表示存储容量的唯一单位。

用 2 片 HY57V561620F 扩展成 32 位 SDRAM，可以认为每个存储单元是 4 个字节。因此当它的地址线 A1:A0=01 时，处理器上对应的地址线应为 ADDR3:ADDR2=01（因为 CPU 的寻址空间是以 Byte 为单位的）。所以 SDRAM 的 A0 引脚接到了 S3C2440 的 ADDR2 地址线上。

同理，如果用 1 片 HY57V561620F，数据线是 16 位，因为一个存储单元是 2 个字节，这时 SDRAM 的 A0 要接到 S3C2440 的 ADDR1 上。

也就是说 SDRAM 的 A0 接 S3C2440 的哪一根地址线是根据整个 SDRAM 的数据位宽来决定的。

上面的接线图上，BA0,BA1 接 ADDR24,ADDR25，为什么用这两根地址线呢？BA0~BA1 代表了 SDRAM 的最高地址位。因为 CPU 的寻址空间是以字节（Byte）为单位的，本系统 SDRAM 容量为 64MByte，那就需要 A25~A0（64M=2^26）地址线来寻址，所以 BA1~BA0 地址线应该接到 2440 的 ADDR25~ADDR24 引脚上。

13 根行地址线+9 根列地址线 = 22 根。另外 HY57V561620F 一个存储单元是 2 个字节，相当于有了 23 根地址线。BA0,BA1 是最高地址位，所以应该接在 ADDR24,ADDR25 上。

也就是说 SDRAM 的 BA0,BA1 接 S3C2440 的哪几根地址线是根据整个 SDRAM 的容量来决定的。

S3C2440 与 NOR FLASH(AM29LV160DB)的接线分析

NOR FLASH 的特点是芯片内执行(XIP, eXecute In Place)，这样应用程序可以直接在 flash 闪存内运行，不必再把代码读到系统 RAM 中。NOR 的传输效率很高，在 1～4MB 的小容量时具有很高的成本效益，但是很低的写入和擦除速度大大影响了它的性能。

NOR FLASH 的地址线和数据线是分开的。

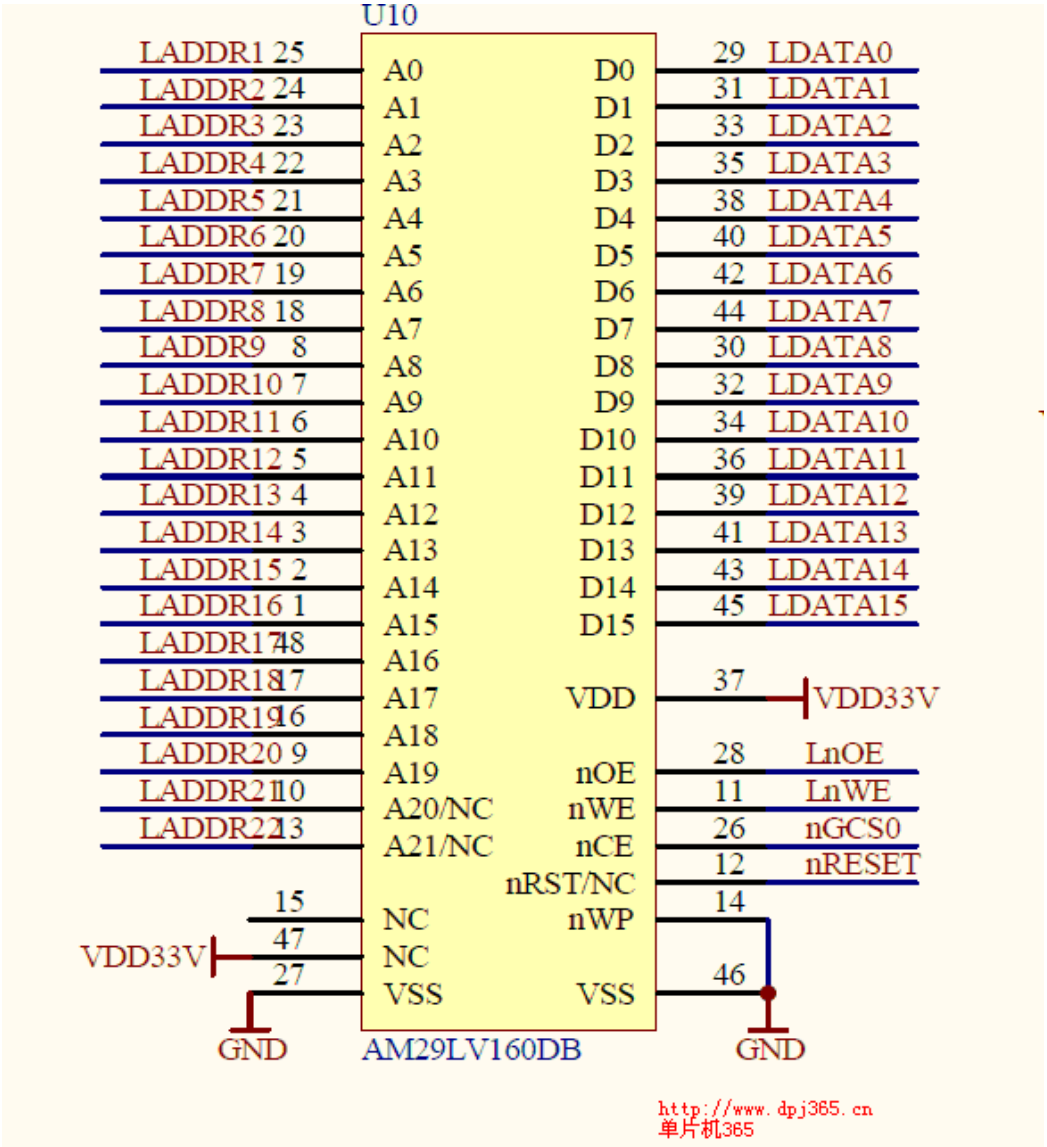
AM29LV160DB 是一个 2Mbyte 的 NOR FLASH，分区结构是：

1 个 16Kbyte 扇区，2 个 8Kbyte 扇区，1 个 32Kbyte 扇区，31 个 64Kbyte 扇区（字节模式）

1 个 8Kbyte 扇区，2 个 4Kbyte 扇区，1 个 16Kbyte 扇区，31 个 32Kbyte 扇区（半字模式）

共 35 个扇区。

下图是 TQ2440 开发板提供的 NOR FLASH 部分接线图。



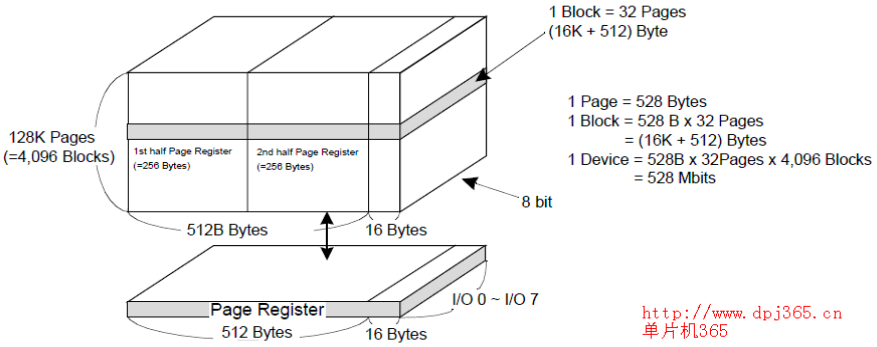
AM29LV160DB 第 47 脚是 BYTE#脚，BYTE#接高电平时，器件数据位是 16 位，接低电平时，数据位是 8 位。上图 BYTE#接 VCC，D0-D15 做为数据输入输出口。
A0-A19 是地址线，在半字模式下，D0-D15 做为数据输入输出口。因为数据位是 16 位，A0-A19 可以选择 2^20 = 1M *2BYTE = 2Mbyte。正好是 AM29LV160DB 的容量。S3C2440 的 ADDR1 要接 AM29LV160DB 的 A0。上图中 AM29LV160DB 的 A20,A21 是空脚，分别接的是 LADDR21,LADDR22。这应该是为了以后方便扩展 NOR FLASH 的容量。LADDR21,LADDR22 对 AM29LV160DB 是没用的。

当 BYTE#接低电平时，D0-D7 做为 8 位数据输入输出口，D15 做为地址线 A-1。相当于有了 A-1,A0-A19 共 21 根地址线。这个时候 S3C2440 的 ADDR0 应该接在 D15（A-1）。。。。ADDR20 接 A19。21 根地址线的寻址空间是 2^21 = 2Mbyte。正好是 AM29LV160DB 的容量。

S3C2440 与 NAND FLASH(K9F1208)的接线分析

NAND FLASH 的接线方式和 NOR FLASH, SDRAM 都不一样。以 TQ2440 开发板用的 K9F1208 为例，分析 NAND FLASH 的接线方式。

K9F1208 结构如下图：



K9F1208 位宽是 8 位。

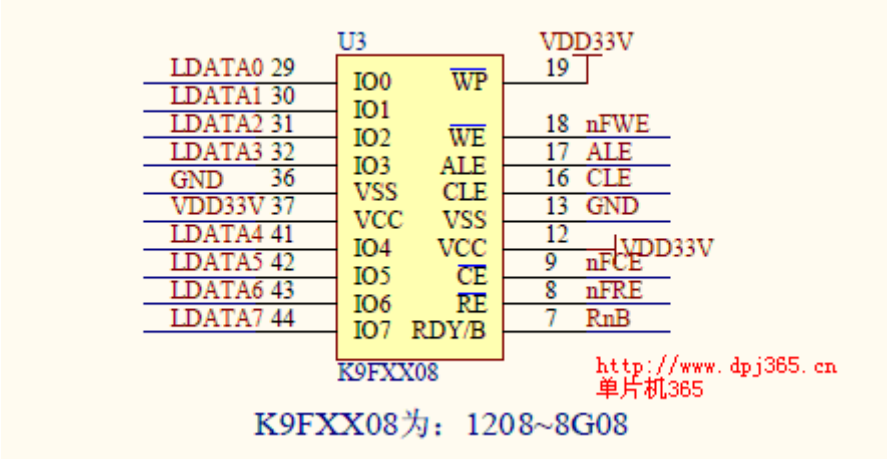
一页： 512byte + 16byte 最后 16byte 是用于存储校验码和其他信息用的，不能存放实际的数据。

一个块有 32 page：（16k+512）byte

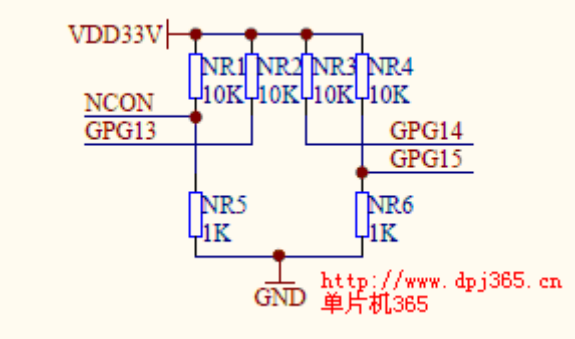
K9F1208 有 4096 个块：(64M+2M)byte，总共有 64Mbyte 可操作的芯片容量

NAND FLASH 以页为单位读写数据，以块为单位擦除数据。

S3C24440 和 K9F1208 的接线图如下：



下图是 S3C2440 的 NAND FLASH 引脚配置：



当选定一个 NAND FLASH 的型号后，要根据选定的 NAND FLASH 来确定 S3C2440 的 NCON,GPG13,GPG14,GPG15 的状态。
下图是 S3C2440 中 4 个脚位状态的定义：

NAND FLASH MEMORY CONFIGURATION TABLE

NCON0	GPG13	GPG14	GPG15
0: Normal NAND	0: 256Words	0: 3-Addr	0: 8-bit bus width
	1: 512Bytes	1: 4-Addr	
1: Advance NAND	0: 1Kwords	0: 4-Addr	1: 16-bit bus width
	1: 2Kbytes	1: 5-Addr	

K9F1208 的一页是 512byte，所以 NCON 接低电平，GPG13 接高电平。
K9F1208 需要 4 个寻址命令，所以 GPG14 接高电平
K9F1208 的位宽是 8，所以 GPG15 接低电平。

NAND FLASH 寻址
对 K9F1208 来说，地址和命令只能在 I/O[7:0]上传递，数据宽度是 8 位。
地址传递分为 4 步，如下图：

	I/O 0	I/O 1	I/O 2	I/O 3	I/O 4	I/O 5	I/O 6	I/O 7	
1st Cycle	A0	A1	A2	A3	A4	A5	A6	A7	Column Address
2nd Cycle	A9	A10	A11	A12	A13	A14	A15	A16	Row Address (Page Address)
3rd Cycle	A17	A18	A19	A20	A21	A22	A23	A24	
4th Cycle	A25	*L	*L	*L	*L	*L	*L	*L	

第 1 步发送列地址，既选中一页 512BYTE 中的一个字节。512byte 需要 9bit 来选择，这里只用了 A0-A7，原因是把一页分成了 2 部分，每部分 256 字节。通过发送的读命令字来确定是读的前 256 字节还是后 256 字节。

当要读取的起始地址（Column Address）在 0~255 内时我们用 00h 命令，当读取的起始地址是在 256~511 时，则使用 01h 命令。
一个块有 32 页，用 A9-A13 共 5 位来选择一个块中的某个页。
总共有 4096 个块，用 A14-A25 共 12 位来选择一个块。
K9F1208 总共有 64Mbyte，需要 A0-A25 共 26 个地址位。

例如要读 NAND FLASH 的第 5000 字节开始的内容。把 5000 分解成列地址和行地址。

Column_address = 5000%512 = 392

Page_address = (5000>>9) = 9

因为 column_address>255,所以用 01h 命令读

发送命令和参数的顺序是：

NFCMMD = 0x01;从后 256 字节开始读

NFADDR = column_address & 0xff;取 column_address 的低 8 位送到数据线

NFADDR = page_address & 0xff;发送 A9-A16

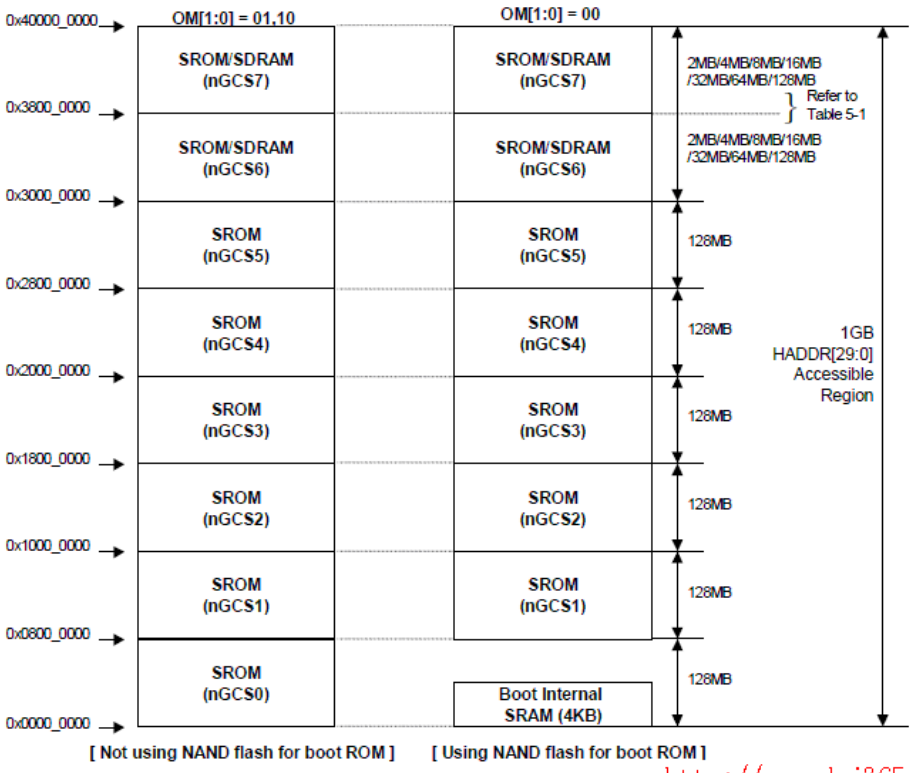
NFADDR = (page_address >>8) & 0xff; 发送 A17-A24

NFADDR = (page_address >> 16) & 0xff;发送 A25

上面的 NFCMMD,NFADDR.是 S3C2440 的 NAND FLASH 控制寄存器。读取的数据会放在 NFDATA 中。

S3C2440开发板中SDRAM \NOR FLASH\ NAND FLASH地址分配

前三篇文章里，我分析了 S3C2440 与 SDRAM,NOR FLASH,NAND FLASH 的连线。在 S3C2440 开发板这个系统中，这三种存储芯片的地址是如何分配的呢？
首先看下图：



这是 S3C2440 的存储器地址分配图，SDARM 只能接在 BANK6 或 BANK7.从分析 SDRAM 接线的文章里的 SDRAM 接线图可以看到，SDRAM 接的是 ngcs6,也就是接在 BANK6,因为选择的 SDRAM 是 2 片 32Mbyte，总容量是 64Mbyte，所以 SDRAM 的地址范围是
0x3000 0000 --- 0x33ff ffff。

S3C2440 的 OM0,OM1 脚决定系统启动模式：

OM1 (Operating Mode 1)	OM0 (Operating Mode 0)	Booting ROM Data width
0	0	Nand Flash Mode
0	1	16-bit
1	0	32-bit
1	1	Test Mode

TQ2440 开发板的 NOR FLASH 是 16bit 数据位宽，选择从 NOR FLASH 启动，所以 OM0 接 VDD,OM1 接 VSS,从分析 NOR FLASH 接线的文章里的接线图可

以看到，NOR FLASH 接的是 ngcs0,也就是接在 BANK0.因为选择的 NOR FLASH 是 2Mbyte，所以 NOR FLASH 的地址范围是 0x0000 0000 --- 0x001f ffff。上电时，程序会从 Norflash 中启动，ARM 直接取 Norflash 中的指令运行。

最后来看 NAND FLASH，NAND FLASH 以页为单位读写，要先命令，再给地址，才能读到 NAND 的数据。NAND FLASH 是接在 NAND FLASH 控制器上而不是系统总线上，所以没有在 8 个 BANK 中分配地址。如果 S3C2440 被配置成从 Nand Flash 启动，S3C2440 的 Nand Flash 控制器有一个特殊的功能,在 S3C2440 上电后,Nand Flash 控制器会自动的把 Nand Flash 上的前 4K 数据搬移到 4K 内部 SRAM 中,系统会从起始地址是 0x0000 0000 的内部 SRAM 启动。程序员需要完成的工作,是把最核心的启动程序放在 Nand Flash 的前 4K 中，也就是说,你需要编写一个长度小于 4K 的引导程序,作用是将主程序拷贝到 SDRAM 中运行。

由于 Nand Flash 控制器从 Nand Flash 中搬移到内部 RAM 的代码是有限的,所以在启动代码的前 4K 里,我们必须完成 S3C2440 的核心配置以及把启动代码(U-BOOT)剩余部分搬到 RAM 中运行，至于将 2440 当做单片机玩裸跑程序的时候，就不要做这样的事情，当代码小于 4K 的时候，只要下到 nand flash 中就会被搬运到内部 RAM 中执行了。

不管是从 NOR FLASH 启动还是从 NAND FLASH 启动，ARM 都是从 0x0000 0000 地址开始执行的。

S3C2440 的时钟设置

一个嵌入式系统中，晶振就像心脏。必须先确定晶振，设置好系统的时钟，WDT, UART, PWM, TIMER 等模块才能正常工作。
和 51 系列单片机相比，S3C2440 的时钟电路很复杂。
首先通过引脚 OM2, OM3 来选择时钟源。

Table 7-1. Clock Source Selection at Boot-Up

Mode OM[3:2]	MPLL State	UPLL State	Main Clock source	USB Clock Source
00	On	On	Crystal	Crystal
01	On	On	Crystal	EXTCLK
10	On	On	EXTCLK	Crystal
11	On	On	EXTCLK	EXTCLK

以 TQ2440 开发板为例，OM2,OM3 都接地，外接 12M 晶振，主时钟源和 USB 时钟源都是外部晶振。

S3C2440 具有 2 个 PLL(Phase Locked Loop：用来产生高频的电路)，一个是 MPLL，用于产生 FCLK， HCLK， PCLK 三种频率， 这三种频率分别有不同的用途：

FCLK 是 CPU 提供的时钟信号，如果提到 CPU 的主频是 400MHz，就是指的这个时钟信号。

HCLK 是为 AHB 总线提供的时钟信号， Advanced High-performance Bus，主要用于高速外设，比如内存控制器，中断控制器，LCD 控制器， DMA 以及 USB host 。

PCLK 是为 APB 总线提供的时钟信号，Advanced Peripherals Bus，主要用于低速外设，比如 WATCHDOG, IIS, I2C， SDI/MMC, GPIO，RTC ，UART, PWM, ADC and SPI 等等。

另外一个是 UPLL，专门用于驱动 USB host/Device。并且驱动 USB host/Device 的频率必须为 48MHz。

时钟电路相关寄存器总共有 7 个，下面分别介绍。
MPLLCON(0X4C00 0004)和 UPLLCON(0X4C00 0008)
这两个寄存器用来设置主锁相环产生的时钟和 USB 锁相环产生的时钟。
MPLL=(2*m*Fin)/(p*2^s)　　UPLL=(m*Fin)/(p*2^s)

其中 m=(MDIV+8), p=(PDIV+2), s=SDIV

P,M 范围：1<=P<=62,1<=M<=248

注意：MDIV[19:12],PDIV[9:4],SDIV[1:0],当设置 MPLL 和 UPLL 值的时候，需要先设置 UPLL 再设置 MPLL。
例如：MPLLCON = (92<<12) | (1<<4) |(1);//FCLK=400M
这里 MDIV=92,PDIV=1,SDIV=1，那么 m=100,p=3,s=1,且 Fin=12M,所以 FCLK=400M

CLKCON(0X4C00 000C) 控制各种模块如 SPI,IIC,UART 等的时钟电路开关以及系统的 SLEEP,IDLE 模式，以便降低系统功耗。默认值是全部时钟电路打开，系统工作在正常模式。

CLKSLOW(0X4C00 0010)用来选择系统是否进入慢模式，以及是否关闭 MPLL 或 UPLL,和在慢模式下的分频率。

CLKDIVN(0X4C00 0014)和 CAMDIVN(0X4C00 0018 照相机时钟分割寄存器)两个寄存器配合来确定 FCLK,HCLK,PCLK 的比例。

LOCKTIME(0X4C00 0000)设置 MPLL,UPLL 的锁存时间，采用默认值 0XFFFFFFFF,锁存时间各为 300us。
说到锁存时间，就要分析下 S3C2440 的时钟工作过程。
在系统复位时，晶振起振稳定后，PLL 开始按照默认值开始工作，但是在复位时，PLL 工作是不稳定的，所以 S3C2440 用 FIN（12M）作为 MPLL。只到一个新的值写入 MPLLCON,UPLLCON,即使用户不想改变复位后 PLLCON 的默认值，仍然需要把这个默认值写入 PLLCON,写入值之后，系统会自动插入一个 PLL LOCK TIME，也就是 LOCKTIME 寄存器中设置的 300us。300us 后，PLL 就开始正常工作。
在系统正常工作时，如果要改变 FCLK，写入新值到 MPLLCON 后，系统也会插入一个 PLL LOCK TIME,300us 后 FCLK 就变成新的频率。

S3C2440 文档里有个注意的地方：
如果 HDIVN 不为 0，根据如下指令，CPU 总线模式从 Fast Bus Mode 变为 Asynchronous（异步总线模式），2440 没有同步总线模式）

```
MMU_setAsyncBusMode
mrc  p15,0,r0,c0,0
orr  r0,r0,#R1_nF:OR:R1_iA
mcr  P15,0,r0,c1,c0,0
```

如果 HDIVN 不为 0，且 CPU 总线模式为 Fast Bus mode,CPU 的时钟为 HCLK.,这种方式可以用在将 CPU 频率降低，但是却又不改变 HCLK 和 PCLK.

查了些资料，ARM 内核在启动后工作在快速模式，改变 HDIVN 值后，就要进入异步模式。转换的具体原因涉及到 ARM 内核。目前还没办法弄懂，ARM 的学习就像走进一条不知道多长的隧道，难见光明。

S3C2440 的定时器设置

S3C2440 有 5 个 16 位定时器，定时器 0-3 有 PWM 功能，定时器 4 有一个没有输出引脚的内部定时器，定时器 0 有一个用于大电流设备的死区生成器。

有 2 个 8 位预分频器和 2 个 4 位分频器。定时器 0 和定时器 1 共用一个 8 位预分频器。定时器 2，定时器 3，定时器 4 共用另一个 8 位预分频器。
定时器的时钟源是 PCLK，首先经过预分频器降低频率后，进入第二个分频，可以生成 5 种不同的分频信号（1/2,1/4,1/8,1/16 和 TCLK）。其中 TCLK0，TCLK1 是 S3C2440 的外部时钟信号输入管脚。

所有定时器都是递减计数。
预分频器值（prescaler value）在 TCFG0(0X5100 0000)中设置
0-7 位设置 TIMER0,TIMER1 的预分频值。
8-15 位设置 TIMER2,3,4 的预分频值。
第二个分频器的值（divider value）在 TCFG1（0X5100 0004）中设置
0-3 位设置 TIMER0 的分频值
4-7 位设置 TIMER1 的分频值
8-11 位设置 TIMER2 的分频值
12-15 位设置 TIMER3 的分频值
16-19 位设置 TIMER4 的分频值
定时器输入频率的计算公式是：
Timer input clock Frequency = PCLK / {prescaler value+1} / {divider value}
{prescaler value} = 0~255
{divider value} = 2, 4, 8, 16

TCON(0X5100 0008)是 5 个定时器的控制位寄存器，以 TIMER0 为例：
0-4 位是 TIMER0 的控制位
Bit0：停止或开始 TIMER0 计时
Bit1：手动更新 TCNTB0 和 TCMPB0
Bit2：TOUT0 逻辑电平是否翻转
Bit3：0-一次性脉冲模式，1-自动装载模式
Bit4：使能或禁止死区功能

Timer0-3 各有 TCNTBn，TCNTn，TCMPBn，TCMPn，TCNTOn 共 5 个寄存器，其中 TCNTn，TCMPn 是内部寄存器，没有对应的地址，通过读 TCNTOn 的值可以得到 TCNTn 的值。当定时器计数到 0，TCNTBn 和 TCMPBn 的值装入 TCNTn 和 TCMPn，如果

中断使能，同时产生中断。在计数过程中，TCNTBn 和 TCMPBn 的值是不变的，变的是 TCNTn 的值。Timer0-3 各有一个对应的输出脚 TOUT0-3。

Timer4 有 TCNTB4,TCNT4,TCNTO4 共 3 个寄存器，其中 TCNT4 是内部寄存器，Timer4 没有对应的输出脚。

按下面的步骤启动一个定时器：

1、 初始化 TCNTBn 和 TCMPBn。

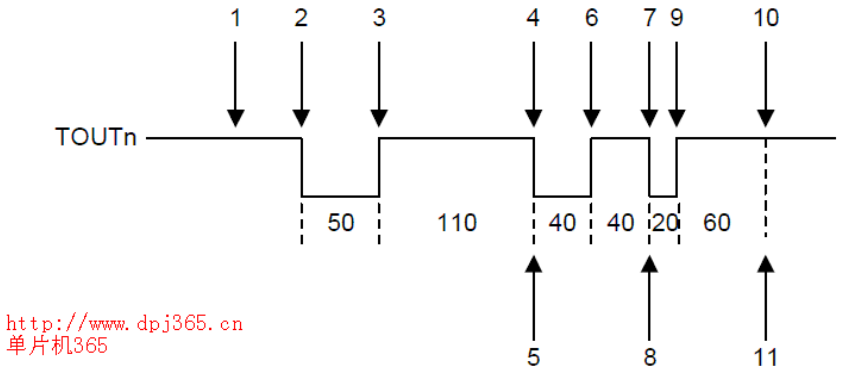
2、 把相应定时器的手动更新位置 1，不管是否使用 TOUTn 极性转换功能，推荐都配置下极性转换功能位。

3、 设置相应定时器的启动位启动定时器，同时清除手动更新位。

不管定时器是否运行，只要 TOUTn 极性转换位改变，TOUTn 逻辑电平也会变改变。所以推荐极性转换位和人工加载位一起设置。

定时器运行时，当 TCNTn 等于 TCMPBn 时，TOUTn 输出的电平会翻转，而当 TCNTn 减为 0 时，产生中断，TOUTn 的电平又会翻转过来。

S3C2440 的 datasheet 里举了一个例子，很好的说明的定时器的工作过程，我把它翻译出来。过程图如下：



1、使能自动装载功能，TCNTBn 设为 160（50+110），TCMPBn 设为 110，置为手动更新标志，把 TCNTBn，TCMPBn 的值装入 TCNTn，TCMPn。TOUTn 翻转功能关闭，然后把 TCNTBn 设为 80（40+40），TCMPBn 设为 40，这是为下一次装载设置的。

2、使能定时器开始计时位，清零手动更新位，定时器开始向下计时。

3、当 TCNTn 的值和 TCMPn 的值相等时，TOUTn 从低变高。

4、当 TCNTn 等于 0 时，产生中断，TCNTBn，TCMPBn 重新装载进 TCNTn，TCMPn，这次的值是 80 和 40，TOUTn 从高变低。

5、在定时器中断程序中，TCNTBn 和 TCMPBn 的值设置成 80（20+60）和 60，这是为下一次装载准备的。

6、当 TCNTn 的值和 TCMPn 相等时，TOUTn 从低变高。

7、当 TCNTn 等于 0，产生中断，TCNTBn，TCMPBn 重新装载进 TCNTn，TCMPn，这次的值是 80 和 60.
8、在定时器中断程序中，关闭定时器自动装载和中断功能。
9、当 TCNTn 的值和 TCMPn 相等时，TOUTn 从低变高。
10、TCNTn 等于 0，TCNTn 不在自动装载，定时器停止计时。
11、不会有中断产生

5V,3.3V 电压匹配

一个产品中有两个芯片，主芯片工作在 3.3V，单片机工作在 5v，两个芯片需要通讯时，电平匹配是一个重要的问题。
如果工作在 5V 的单片机的高电平门限值是 0.7VDD，就是 3.5V，那么直接把 3.3V 的信号接到 5V 单片机的输入口，可能会造成通讯失败。这个问题硬件和软件各有一个解决办法。
硬件方面，因为 5V 单片机的输入口一般都接有上拉电阻，3.3V 的输入信号接一个电阻后再连接到 5V 单片机的输入口，因为所接电阻的分压作用，会使接到 5V 单片机输入口的电压>3.3V,调整所接电阻值就能使输入电压>0.7VDD.
软件方面，在主芯片输出 3.3V 高电平时，把要输出高电平的 IO 口设置成高阻状态，那么这个时候因为单片机输入 IO 口的上拉电阻，单片机 IO 的输入电压就是 5V 了。在主芯片输出低电平时，把主芯片的 IO 口设置成输出口输出 0 就可以了。