```
    ONENAND_EnableECCCorrection(Controller, ONENAND_WITH_CORRECT);
    ONENAND_EnableIOBE(Controller, 1);
    ONENAND_SetBurstLatency(Controller, eOND_LATENCY4);
    ONENAND_SetSyncMode(Controller, eOND_SYNC_BURST16);

    //ONENAND_EnableAllInterrupt(Controller);
    //ONENAND_DisableInterrupt(Controller,    eOND_RDYACT);

#if (OND_TRANS_MODE == OND_DMA)
    DMAC_InitCh(DMA0, DMA_ALL, &g_oONDDmac0);
    INTC_SetVectAddr(NUM_DMA0,    ONENAND_DmaISR);
    INTC_Enable(NUM_DMA0);
#endif

    return TRUE;
}
```

7.8.2.设置脉冲读取延迟

```
//eLatency - (OneNAND_eLATENCY)等待周期
void ONENAND_SetBurstLatency(u32 Controller, OneNAND_eLATENCY eLatency)
{
    u32 uBurstReadLatency;

    uBurstReadLatency = Inp32(&ONENAND(Controller)->rMemCfg);
    uBurstReadLatency &= ~(0x7<<12);

    uBurstReadLatency |= (eLatency<<12);
```

```
    Outp32(&ONENAND(Controller)->rMemCfg, uBurstReadLatency);

}
```

7.8.3.设置 ONENAND 内部闪存时钟

```
void ONENAND_SetFlashClock(u32 Controller, OneNAND_eFlashClock eFlashClock)
{
    u32 uClkDivider;

    uClkDivider = SYSC_GetDIV0();
    uClkDivider = (uClkDivider & ~(3<<16)) | (eFlashClock<<16);

    SYSC_SetDIV0_all(uClkDivider);
}
```

7.8.4.设置周期数

该段程序是用来设置覆盖闪存设备访问时间的周期数。

```
void ONENAND_SetAccClock(u32 Controller)
{
    u32 uClkDivider, uAccClock;

    uClkDivider = SYSC_GetDIV0();
    uClkDivider = (uClkDivider & (3<<16))>>16;

    OneNand_Inform[Controller].uFlashClock = g_HCLKx2/(uClkDivider+1);
```

```
    uAccClock = (35*(OneNand_Inform[Controller].uFlashClock/1000))/1000000 + 1;
    Outp32(&ONENAND(Controller)->rAccClock, uAccClock&0x7);
}
```

7.8.5.解锁存储器模块

```
bool ONENAND_UnlockBlock(u32 Controller, u32 uStartBlkAddr, u32 uEndBlkAddr)
{
    //u32 uData;

    OneNandT_oIntFlag.IntActInt = 0;
    //uStartBlkAddr –解锁的开始块，uEndBlkAddr –  解锁的结尾块
    if(uStartBlkAddr == uEndBlkAddr)
    {
        ONENAND_WriteCmd(Controller, uEndBlkAddr, 0, 0x09);
    }
    else
    {
        ONENAND_WriteCmd(Controller, uStartBlkAddr, 0, 0x08);
        ONENAND_WriteCmd(Controller, uEndBlkAddr, 0, 0x09);
    }

    //中断
    while(!OneNandT_oIntFlag.IntActInt);

#if 0
    ONENAND_DirectRead(Controller, OND_WPROT_STATUS, &uData);

    if ((uData&0xFFFF) != 0x4)
```

```
    {
        return TRUE;
    }
#endif

    return FALSE;
}
```

7.8.6.锁定模块

```
bool ONENAND_LockBlock(u32 Controller, u32 uStartBlkAddr, u32 uEndBlkAddr)
{
    //u32 uData;

    OneNandT_oIntFlag.IntActInt = 0;

    if(uStartBlkAddr == uEndBlkAddr)
    {
        ONENAND_WriteCmd(Controller, uEndBlkAddr, 0, 0x0B);
    }
    else
    {
        ONENAND_WriteCmd(Controller, uStartBlkAddr, 0, 0x0A);
        ONENAND_WriteCmd(Controller, uEndBlkAddr, 0, 0x0B);
    }

    //中断
```

```
    while(!OneNandT_oIntFlag.IntActInt);

#if 0
    ONENAND_DirectRead(Controller, OND_WPROT_STATUS, &uData);

    if ((uData&0xFFFF) != 0x2)
    {
        //UART_Printf(" Failed lock block, locked status (0x%x)\n", uData);
        return TRUE;
    }
#endif

    return FALSE;
}
```

8 NAND FLASH 控制器

目前的 NOR FLASH 存储器价格比较昂贵，而 SDRAM 和 NAND FLASH 存储器的价格相对来说比较合适，这就是为什么消费者更喜欢从 NAND FLASH 启动引导系统，而在 SDRAM 上执行主程序代码的原因。

S3C6410 恰好满足这一要求,它可以实现从 NAND FLASH 上执行引导程序。S3C6410 具备了一个内部 SRAM 缓冲器，叫做“STEPPINGSTONE”，支持 NAND FLASH 的系统引导。当系统启动时，NAND FLASH 存储器的前面 4KB 将被自动载入到 STEPPINGSTONE 中，然后系统自动执行这些载入的引导代码。

通常情况下，这 4K 的引导代码需要将 NAND FLASH 中程序内容拷贝到 SDRAM 中，在引导代码执行完毕后跳转到 SDRAM 执行。使用 S3C6410 内部硬件 ECC 功能可以对 NAND FLASH 的数据进行有效性的检测。

8.1 NAND FLASH 控制器的特性

NAND FLASH 控制器的特性如下：

（1）自动导入模式：复位后，引导代码被送入 4KB 的 STEPPINGSTONE 中，引导代码移动完毕，引导代码将在 STEPPINGSTONE 中执行。

注：在导入期间，NAND FLASH 控制器不支持 ECC 纠正。

（2）NAND FLASH 控制器 I/F：支持 512 字节和 2KB 页。

（3）软件模式：用户可以直接访问 NAND FLASH 控制器。例如这个特性可以用于读/擦/编程 NAND FLASH 存储器。

（4）接口：8 位 NAND FLASH 存储器接口总线。

（5）硬件 ECC 产生、检测和标志（软件纠正）。

（6）支持 SLC 和 MLC 的 NAND FLASH 控制器：1 位 ECC 用于 SLC，4 位 ECC 用于 MLC 的 NAND FLASH。

（7）特殊功能寄存器 I/F：支持字节/半字/字数据的访问和 ECC 的数据寄存器，用字来访问其他寄存器。

（8）STEPPINGSTONE I/F：支持字节/半字/字的访问。

（9）4KB 内部 SRAM 缓冲器 STEPPINGSTONE，在 NAND FLASH 引导后可以作为其他用途使用。

NAND FLASH 控制器的结构，如图 8-1 所示。

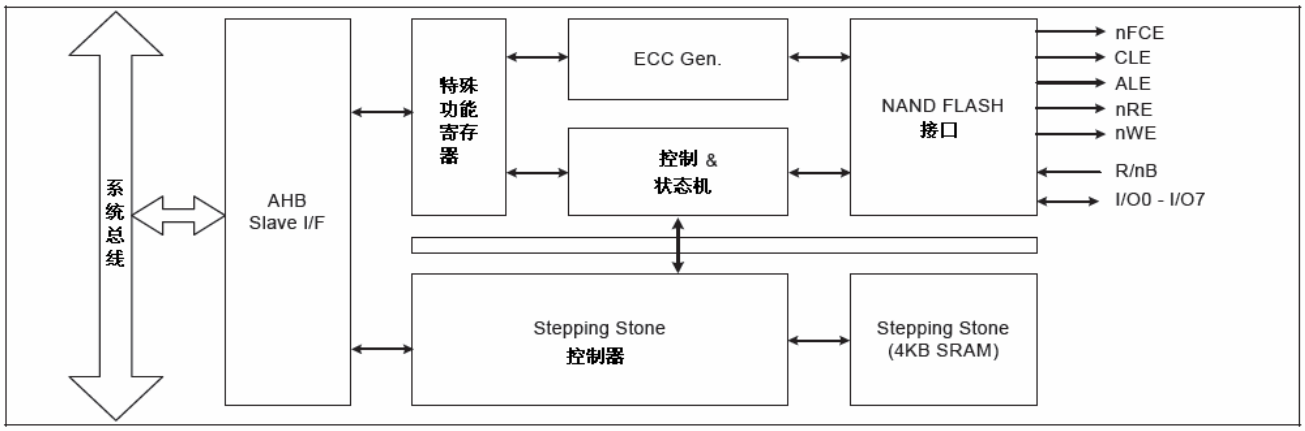

图 8-1 NAND FLASH 控制器结构图

1. NAND FLASH 控制器工作机制

NAND FLASH 控制器的工作机制，如图 8-2 所示。在上电复位时，NAND FLASH 控制器将通过 XOM（参照引脚配置）引脚状态来获得关于连接 NAND FLASH 的信息。在上电或系统复位之后，NAND FLASH 控制器自动加载 4KB 的启动代码。加载完成后，启动代码将在 STEPPINGSTONE 中被执行。

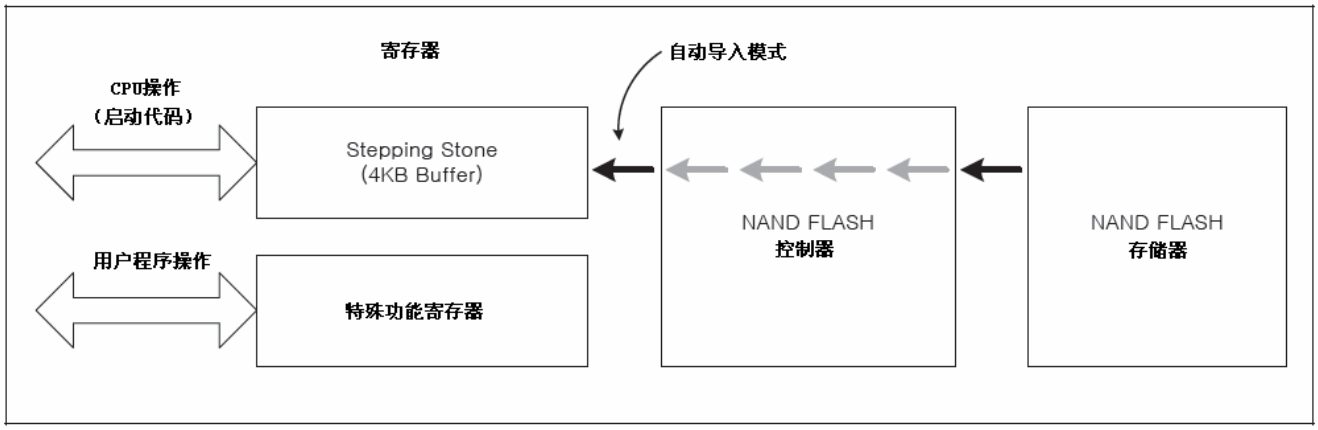

图 8-2 NAND FLASH 控制器的工作机制

注：在自动导入期间，ECC 是未被选中状态。因此，前 4KB 的 NAND FLASH 绝不能有位错误。

2. 引脚配置

以下是相应的引脚配置如表 8-1 所示。

表 8-1 引脚配置

OM[4:0]	Adv 闪存	页大小	地址周期	总线宽度
0000x	0: Normal NAND	1: 512 字节	0: 3 周期	0: 8 位数据总线
0001x	0: Normal NAND	1: 512 字节	1: 4 周期	0: 8 位数据总线
0010x	1: Advance NAND	1: 2K 字节	0: 4 周期	0: 8 位数据总线
0011x	1: Advance NAND	1: 2K 字节	1: 5 周期	0: 8 位数据总线

以上的引脚配置可适用于 NAND FLASH 被用作启动存储器的情况，如果 NAND FLASH 不能用作启动存储器，引脚的配置可以通过设置 NFCON SFR ’ NFCONF’ (0x70200000)来改变。

3. NAND FLASH 存储器时序

NAND FLASH 存储器时序，CLE 和 ALE 时序如图 8-3 所示，nWE 和 nRE 时序如图 8-4 所示。

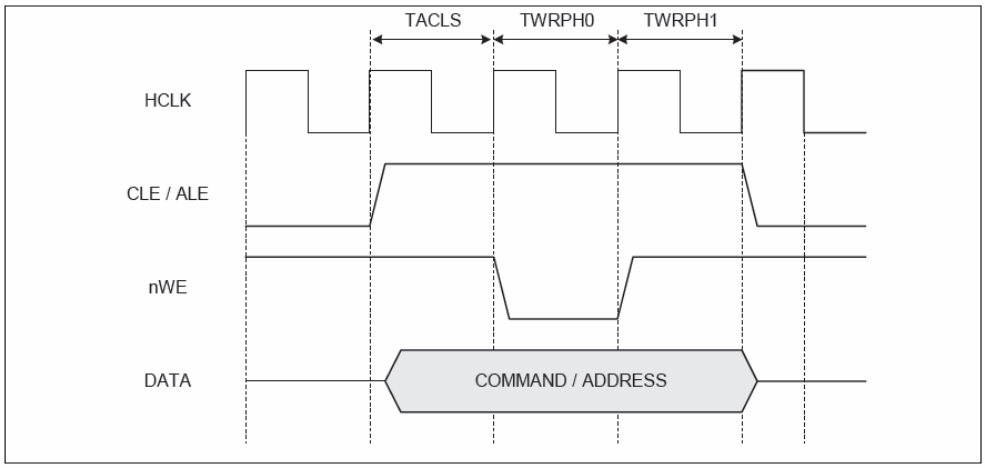

图 8-3 CLE 和 ALE 时序 (TACLS=1, TWRPH0=0, TWRPH1=0)

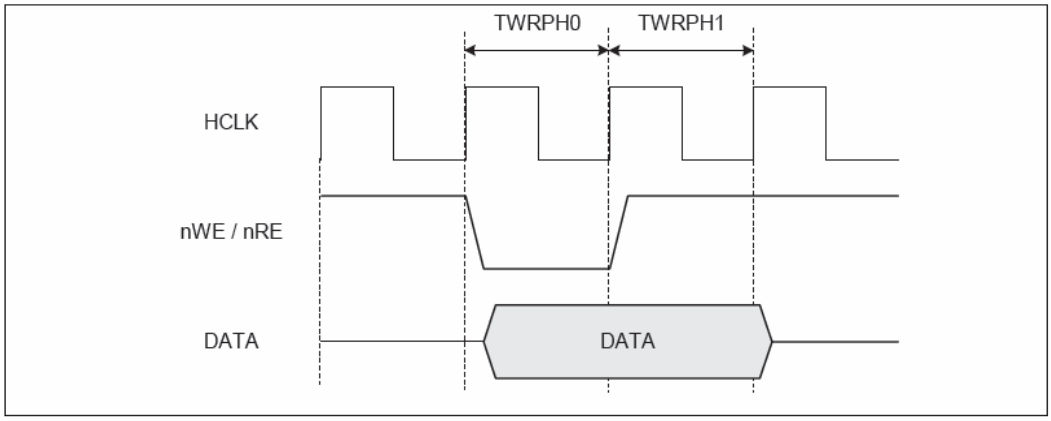

图 8-4 nWE 和 nRE 时序 (TWRPH0=0, TWRPH1=0)

4. 软件模式

S3C6410 仅支持软件模式访问。使用这种模式完全地进入 NAND FLASH 存储器。NAND FLASH 控制器支持直接访问 NAND FLASH 存储器接口。

（1）写命令寄存器=NAND FLASH 存储器命令周期。

（2）写地址寄存器=NAND FLASH 存储器地址周期。

（3）写数据寄存器=写数据到 NAND FLASH 存储器（写周期）。

（4）读数据寄存器=从 NAND FLASH 存储器读数据（读周期）。

（5）读主 ECC 寄存器和备用 ECC 寄存器=从 NAND FLASH 存储器读数据。

注：在软件模式下，必须通过利用检测和中断来检查 RnB 输入引脚的状态。

5. 数据寄存器配置

8 位 NAND FLASH 存储器接口。

A. 字访问

NAND FLASH 存储器接口的字访问，如表 8-2 所示。

表 8-2 字访问描述

寄存器	端	位[31:24]	位[23:16]	位[15:8]	位[7:0]
NFDATA	小端	4th I/O[7:0]	3rd I/O[7:0]	2nd I/O[7:0]	1st I/O[7:0]

B．半字访问

NAND FLASH 存储器接口的半字访问，如表 8-3 所示。

表 8-3 半字访问描述

寄存器	端	位[31:24]	位[23:16]	位[15:8]	位[7:0]
NFDATA	小端	无效值	无效值	2nd I/O[7:0]	1st I/O[7:0]

C．字节访问

NAND FLASH 存储器接口的字节访问，如表 8-4 所示。

表 8-4 字节访问描述

寄存器	端	位[31:24]	位[23:16]	位[15:8]	位[7:0]
NFDATA	小端	无效值	无效值	无效值	1st I/O[7:0]

8.2 STEPPINGSTONE (4KB SRAM)

NAND FLASH 控制器使用 Steppingstone 作为缓冲器引导，也可以使用它进行各种其他用途。

1.SLC / MLC ECC（错误纠正码）

NAND FLASH 控制器有 4 个 ECC（错误纠正码）模块用于 SLC 类型的 NAND FLASH 存储器，1 个 ECC 模块用于 MLC 类型的 NAND FLASH 存储器。

SLC 类型的 NAND FLASH 存储器接口，NAND FLASH 控制器组成 4 个 ECC 模块。这些模块可用于（高达）2048 字节 ECC 奇偶校验码的产生，以及其他可用于（高达）4 字节 ECC 奇偶校验码的产生。

MLC 类型的 NAND FLASH 存储器接口，NAND FLASH 控制器组成 1 个 ECC 模块。这些模块仅用于 512 字节的 ECC 奇偶校验码的产生。8 位存储器接口，MLC 的 ECC 模块每 512 字节生成奇偶校验码。然而，SLC 的 ECC 模块生成奇偶校验码是将每个字节分开。

以下是 ECC 奇偶校验码和两个 SLC 的 ECC 模块表格：

- 28 位 ECC 奇偶校验码=22 位行奇偶校验+6 位列奇偶校验
- 10 位 ECC 奇偶校验码=4 位行奇偶校验+6 位列奇偶校验

（1）2048 字节 SLC 的 ECC 奇偶校验码分配表，如表 8-5 所示。