SYSTEM DESIGN

OVERVIEW

The S3C44B0X, SNASUMG's 16/32-bit RISC microcontroller is cost-effective and high performance microcontroller solution for hand-held device and general application. The integrated on-chip functions of S3C44B0X are

- 2.5V Static ARN7TDMI CPU Core with 8KB cache (SAMBA bus architecture up to 75MHz)
- External memory controller (FP/EDO/SDRAM Control, Chip Select logic)
- LCD Controller (up to 256 color DSTN) with 1-ch LCD-dedicated DMA
- 2-ch general DMAs / 2-ch peripheral DMAs with external request pins
- 2-ch UART / 1-ch SIO (IRDA1.0, 16-byte FIFO)
- 1-ch multi-master IIC-BUS controller & 1-ch IIS-BUS controller
- 5-ch PWM Timers & 1-ch internal timer
- Watch Dog Timer
- 71-bit general purpose I/O ports / 8-ch External Interrupt Source
- Power control : Normal, Slow, Idle and Stop mode
- 8-ch 10-bit ADC
- RTC with calendar function
- On-chip clock generator with PLL

Therefore, you can use S3C44B0X as amount types of system.

APPLICABLE SYTEM WITH S3C44B0X

If your product need to be networked, the S3C44B0X, SNASUMG's 16/32-bit RISC microcontroller can be reduce your system cost. There are sample system, it can be designed with S3C44B0X.

- GPS phone
- PDA (Personal Data Assistance)
- Fish Finder
- Portable Game Machine
- Fingerprint Identification System
- TWM (Two Way Messaging) Terminal
- Car Navigation System
- MP3 Player etc.

MEORY INTERFACE DESIGN

BOOT ROM DESIGN

When system reset, a S3C44B0X access 0x00000000 address. And S3C44B0X should be configure some system variable after reset. Therefore this special code (BOOT ROM image) should be located on address 0x00000000. A boot ROM can have a various width of data bus, and it is controlled by OM[1:0] pins.

Table 4-1. Data Bus Width for ROM Bank 0

OM[1:0]	Data Bus Width
00	8-bit (byte)
01	16-bit (half-word)
10	32-bit (word)
11	Test Mode

ONE BYTE BOOT ROM DESIGN

A design with one byte boot ROM is shown in Figure 4-1.

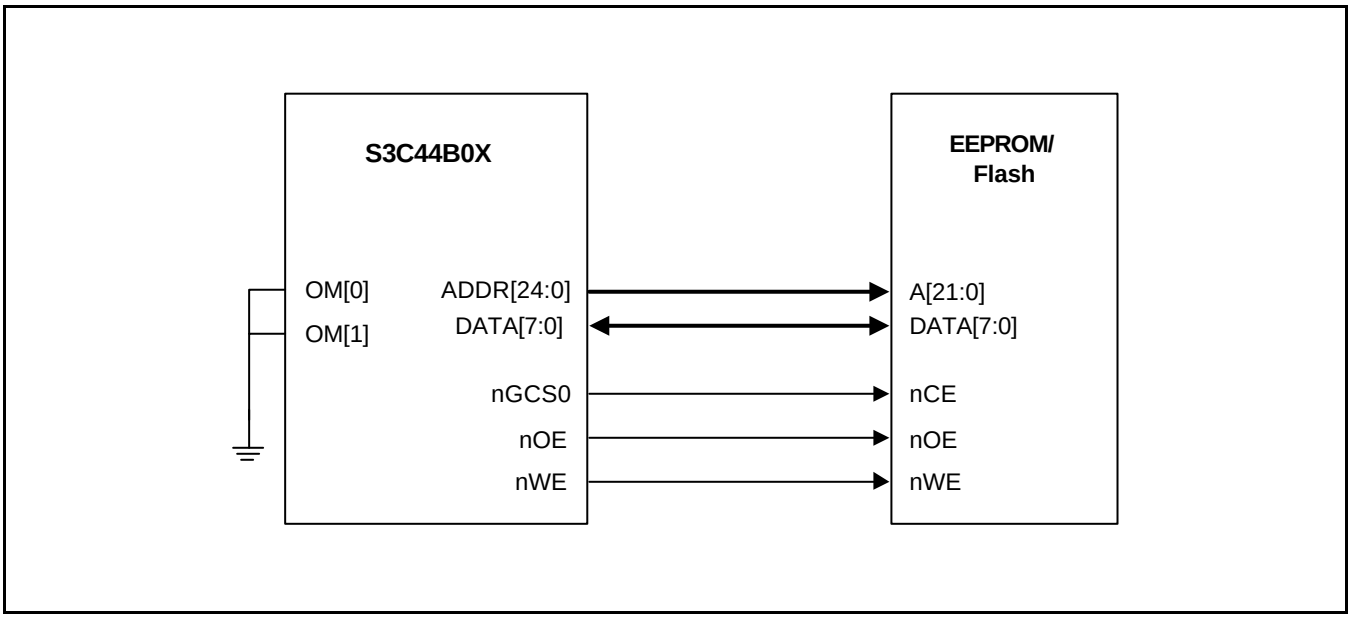


Figure 4-1. One Byte Boot ROM Design

MAKE AND FUSING ONE BYTE ROM IMAGE

When make one byte ROM image, you can use the binary file that made from compile and link.

HALF-WORD BOOT ROM DESIGN WITH BYTE EEPROM/FLASH

A design with half-word boot ROM with byte EEPROM/Flash is shown in Figure 4-2.

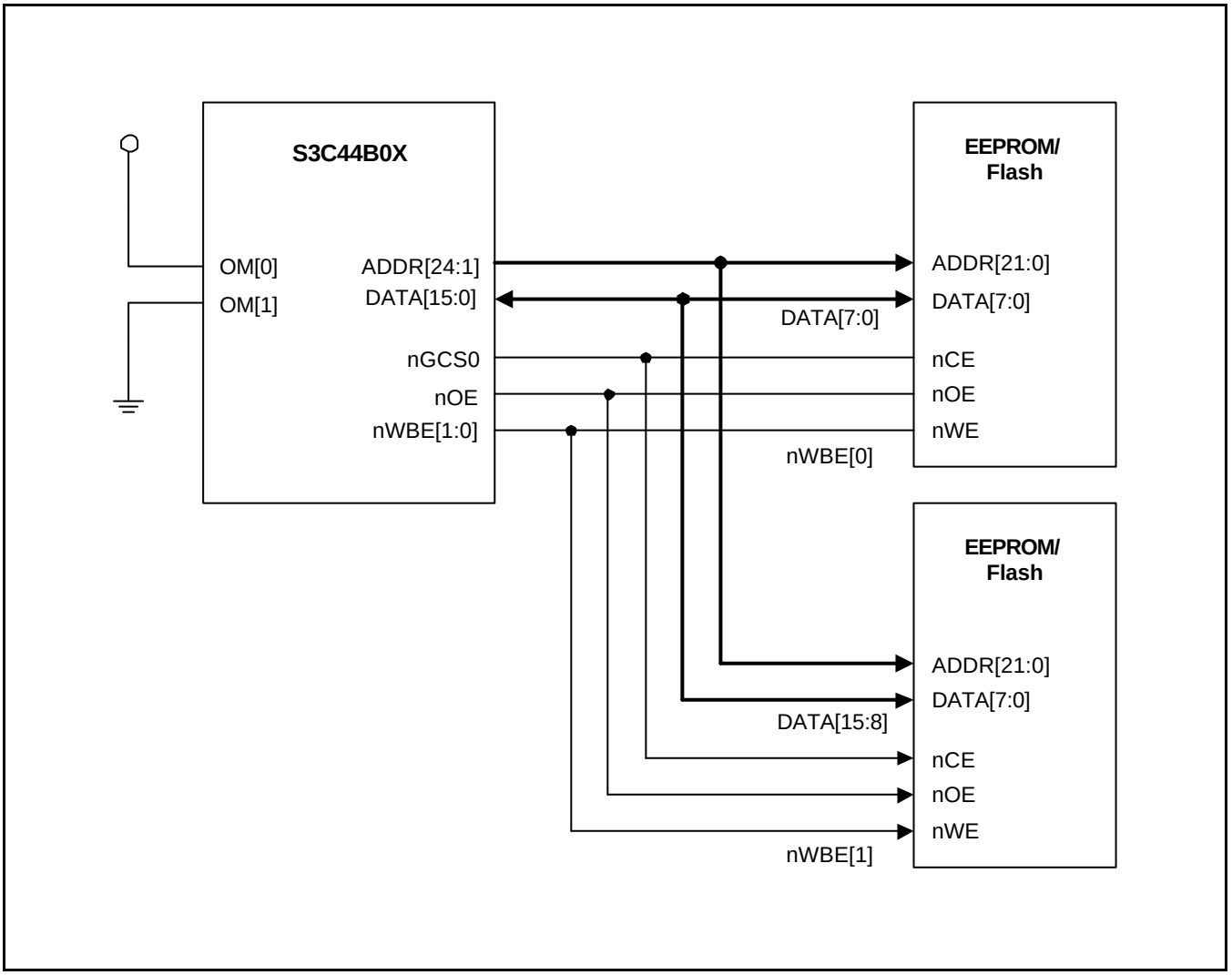


Figure 4-2. The Half-Word Boot ROM Design with Byte EEPROM/Flash

MAKE AND FUSING HALF-WORD ROM IMAGE WITH BYTE EEPROM/FLASH

When make half-word ROM image, you can split two image files, EVEN and ODD.

Table 4-2. Relationship ROM Image and Endian

	Big Endian	Little Endian
DATA[7:0]	Odd	Even
DATA[15:8]	Even	Odd

HALF-WORD BOOT ROM DESIGN WITH HALF-WORD EEPROM/FLASH

A design with half-word boot ROM with byte EEPROM/Flash is shown in Figure 4-3.

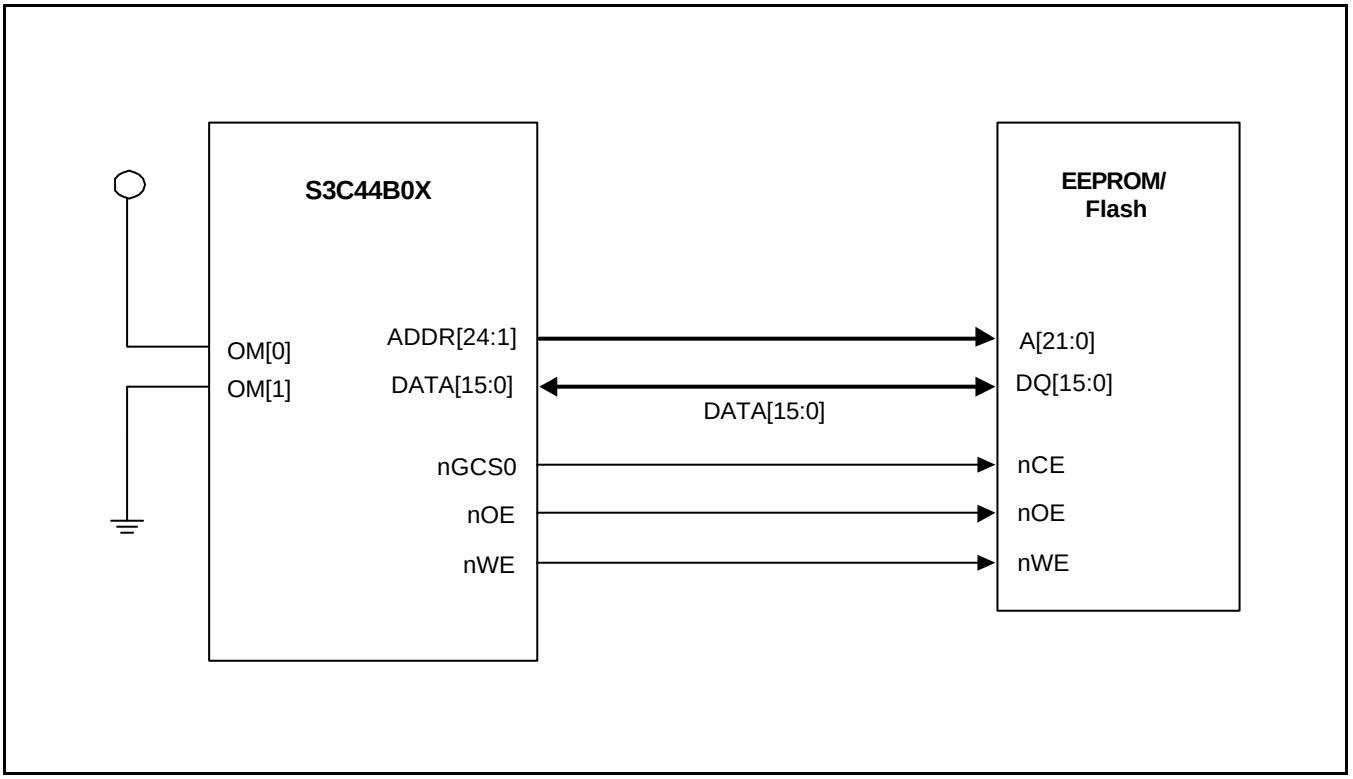


Figure 4-3. The Half-Word Boot ROM Design with Half-Word EEPROM/Flash

WORD BOOT ROM DESIGN WITH HALF-WORD EEPROM/FLASH

A design with word boot ROM with byte EEPROM/Flash is shown in Figure 4-4.

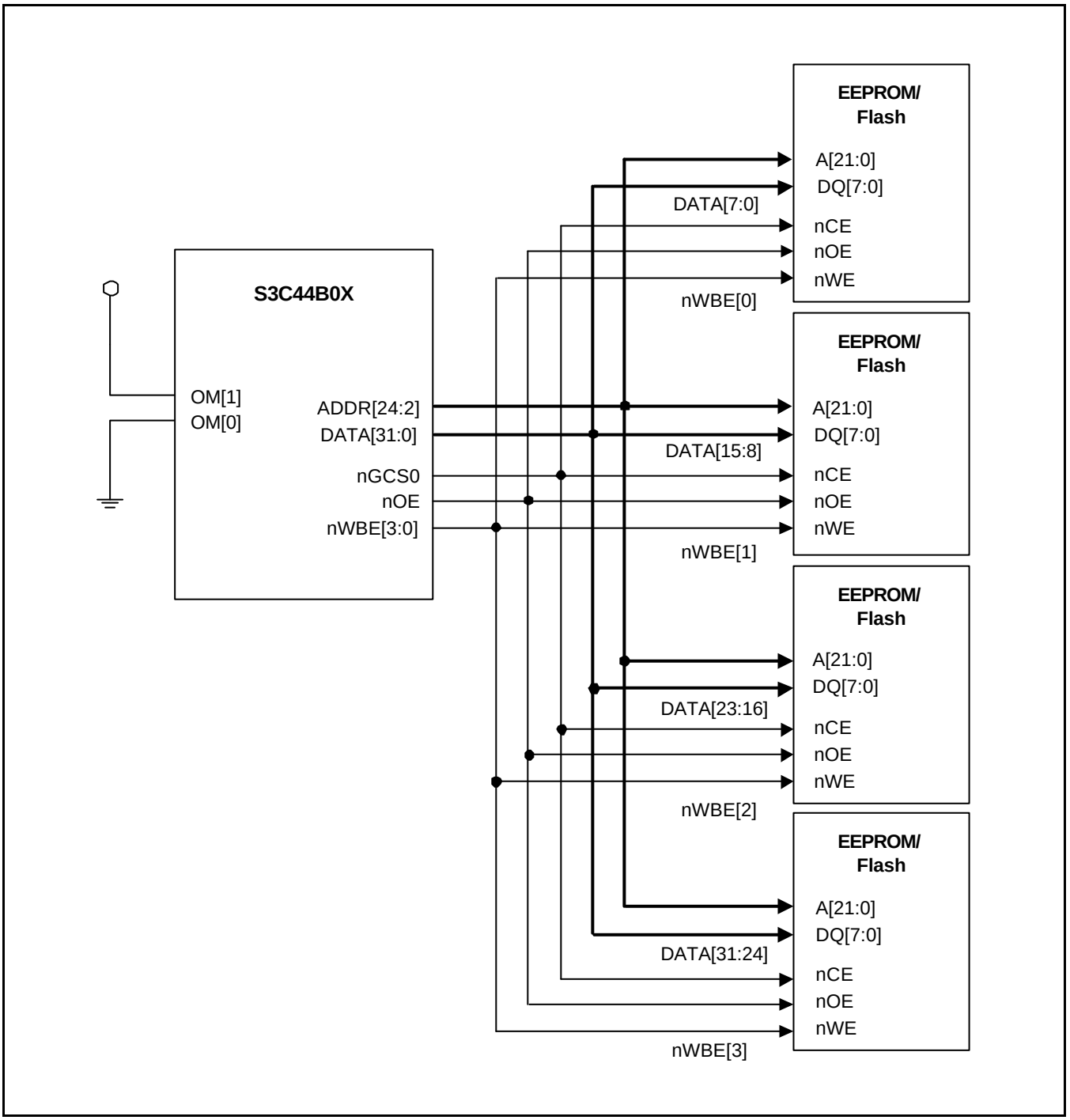
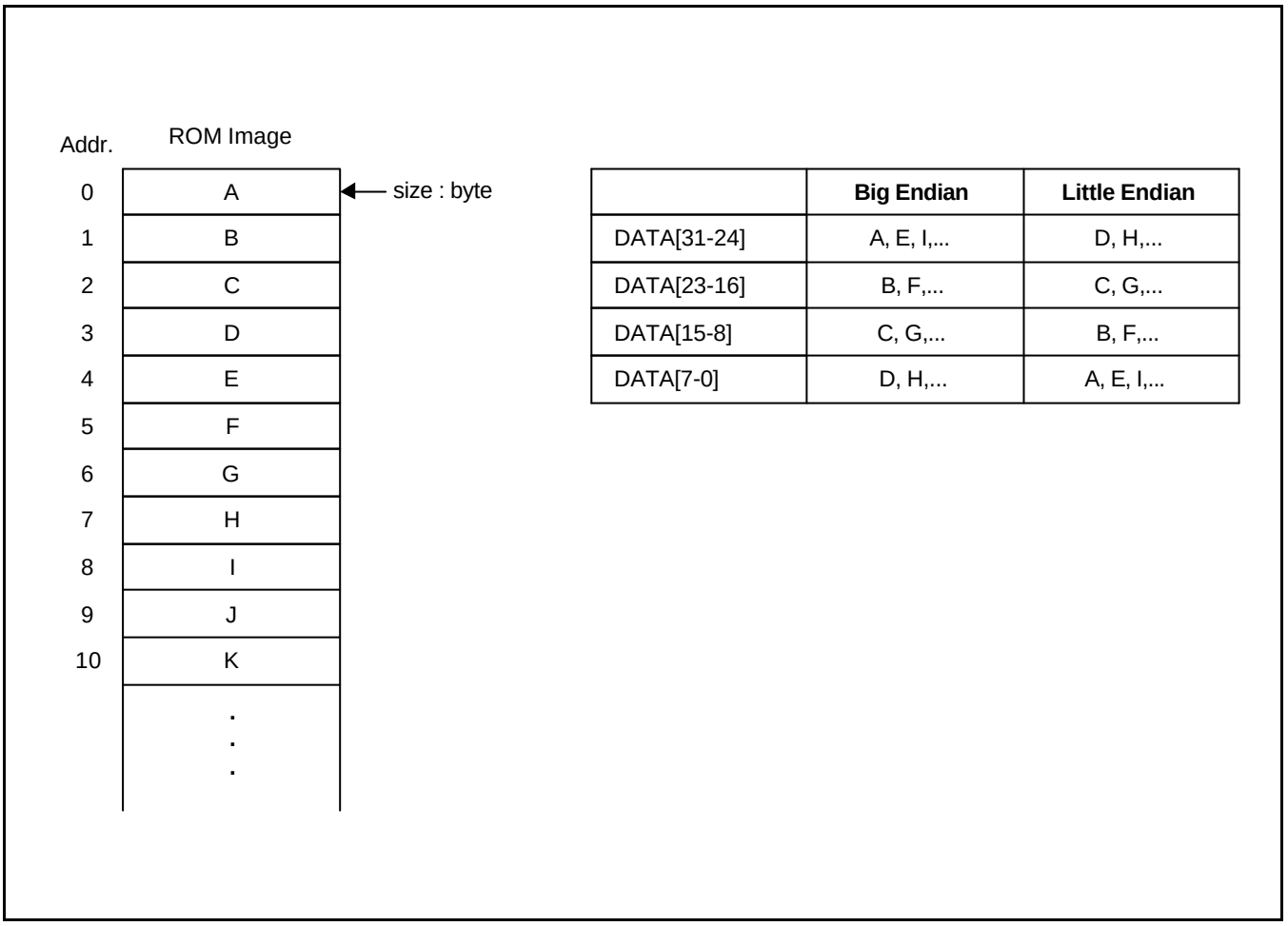


Figure 4-4. The Word Boot ROM Design with Byte EEPROM/Flash

MAKE AND FUSING WORD ROM IMAGE WITH BYTE EEPROM/FLASH

When you make word ROM image, you can split four image file.



	Big Endian	Little Endian
DATA[31-24]	A, E, I,...	D, H,...
DATA[23-16]	B, F,...	C, G,...
DATA[15-8]	C, G,...	B, F,...
DATA[7-0]	D, H,...	A, E, I,...

Figure 4-5. Relationship ROM Image and Endian

MEMORY BANKS DESIGN AND CONTROL

The S3C44B0X has 6 ROM/SRAM banks (ROM0 bank for boot ROM) and 2 ROM/SRAM/FP/EDO/SDRAM banks. The system manager on S3C44B0X can control access time, data bus width for each banks by S/W. The access time of ROM/SRAM banks and FP/EDO/SDRAM banks is controlled by BANKCON0~7 and BANKCON6~7 control register on system manager. The type memory of bank6 & 7 has to be same. (example ROM & ROM, SDRAM & SDRAM) The data bus width for each ROM/SRAM/DRAM banks is controlled by BWSCON control register.

The ROM bank 0 is used for boot ROM bank, therefore bank 0 is controlled by H/W, OM[1:0] is used for this purpose.

The control of BWSCON, BANKCON0-7, REFRESH, BANKSIZE, MRSRB6/7 is performed when system reset, by special command, LDMIA and STMIA. Sample code for special register configuration is described below.

Sample code for special register configuration

```
        LDR     r0, =SMRDATA
        LDMIA   r0, {r1-r13}
        LDR     r0, =0x01c80000         ;BWSCON Address
        STMIA   r0, {r1-r13}

. . . . . .

SMRDATA
        DCD 0x22221210          ;BWSCON
        DCD 0x00000600          ;GCS0
        DCD 0x00000700          ;GCS1
        DCD 0x00000700          ;GCS2
        DCD 0x00000700          ;GCS3
        DCD 0x00000700          ;GCS4
        DCD 0x00000700          ;GCS5
        ;DCD 0x0001002a         ;GCS6 EDO DRAM(Trcd=3,Tcas=2,Tcp=1,CAN=10)
        ;DCD 0x0001002a         ;GCS7 EDO DRAM(Trcd=3,Tcas=2,Tcp=1,CAN=10)
        DCD 0x00018000          ;GCS6 SDRAM(Trcd=2,SCAN=8)
        DCD 0x00018000          ;GCS7 SDRAM(Trcd=2,SCAN=8)
        DCD 0x00a60000+953      ;Refresh(REFEN=1,TREFMD=0,Trp=3.5(D)or 4(SD),
                                ;       Trc=5(S), Tchr=3(D),Ref CNT)
        DCD 0x0                 ;Bank size, 32MB/32MB
        DCD 0x20                ;MRSR 6(CL=2)
        DCD 0x20                ;MRSR 7(CL=2)
```

ROM/SRAM BANKS DESIGN

The ROM/SRAM banks 1-7, can have a various width of data bus, and the bus width is controlled by S/W. A sample design for ROM/SRAM bank 1-7 is shown in Figure 4-6, Figure 4-7, Figure 4-8 and Figure 4-9.

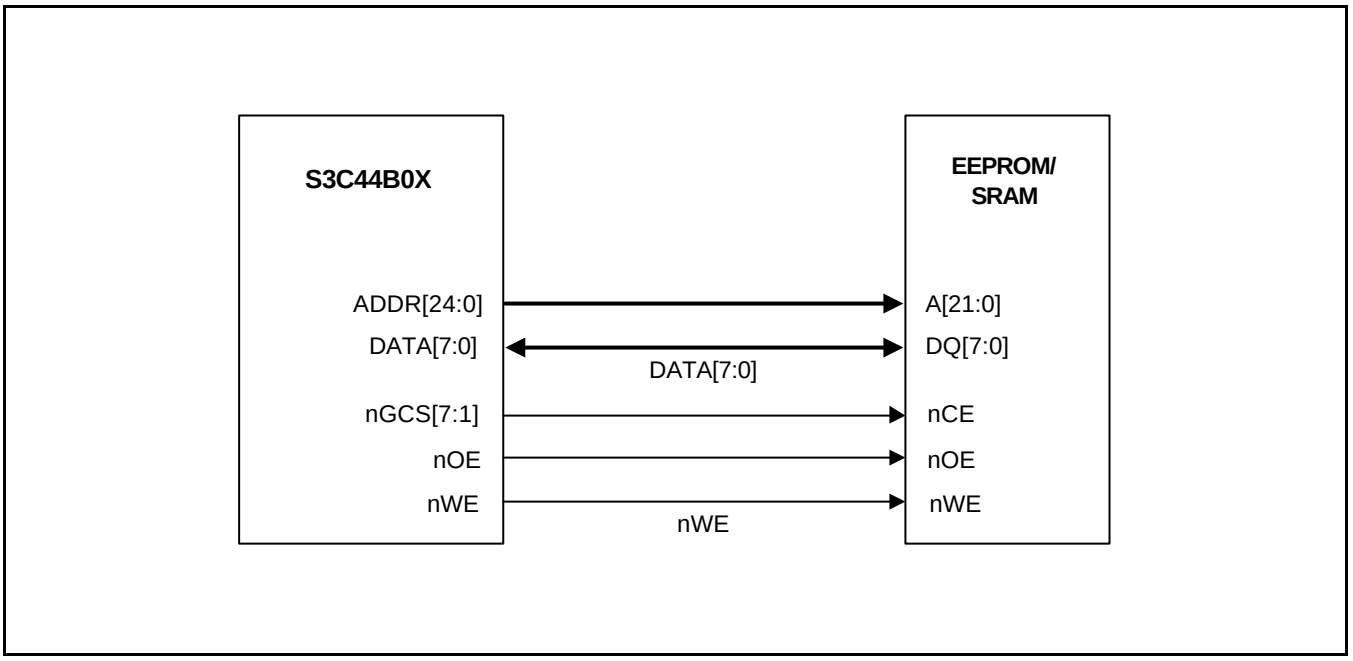


Figure 4-6. One-byte EEPROM/SRAM Banks Design

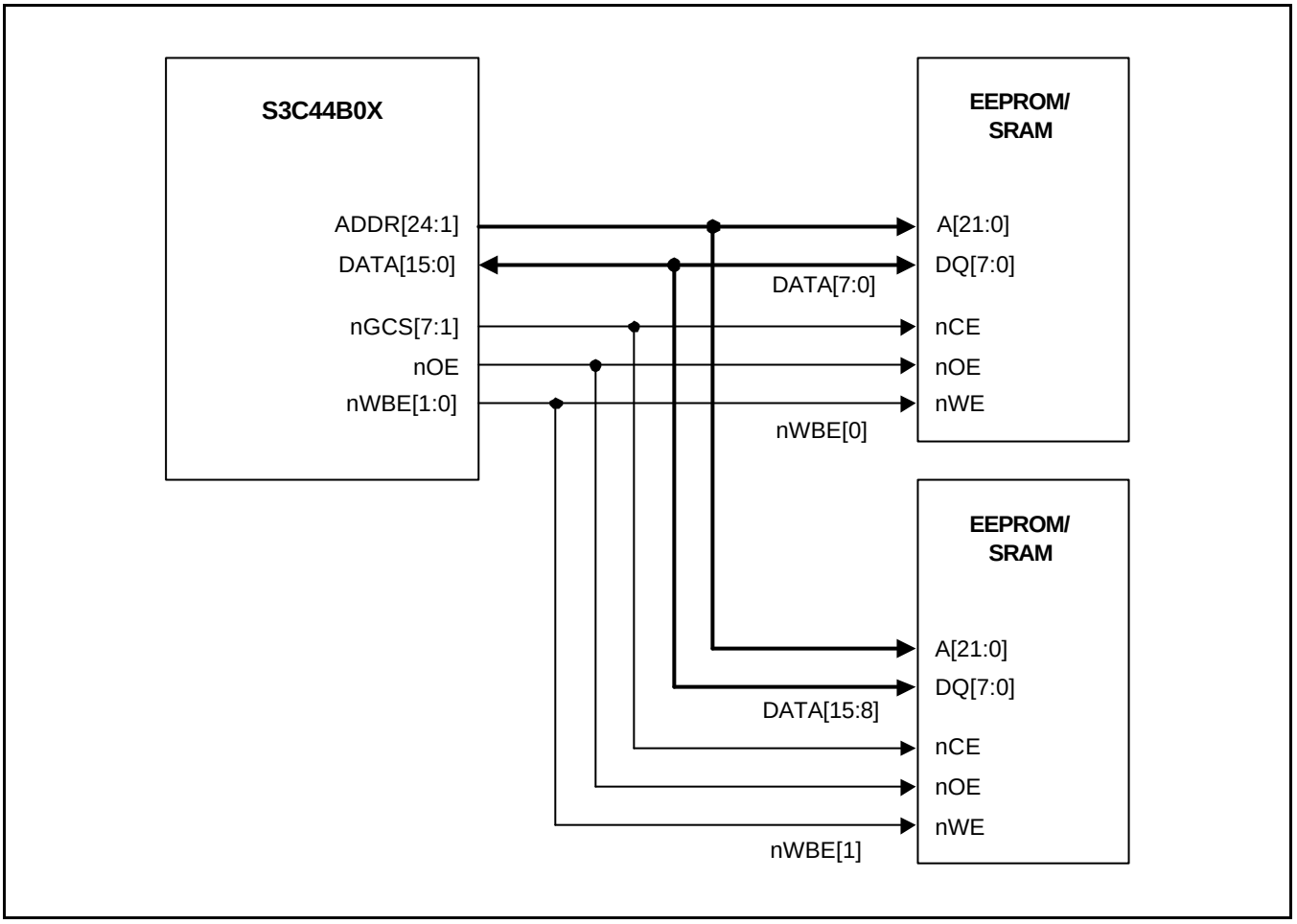


Figure 4-7. Half-word EEPROM/SRAM Banks Design

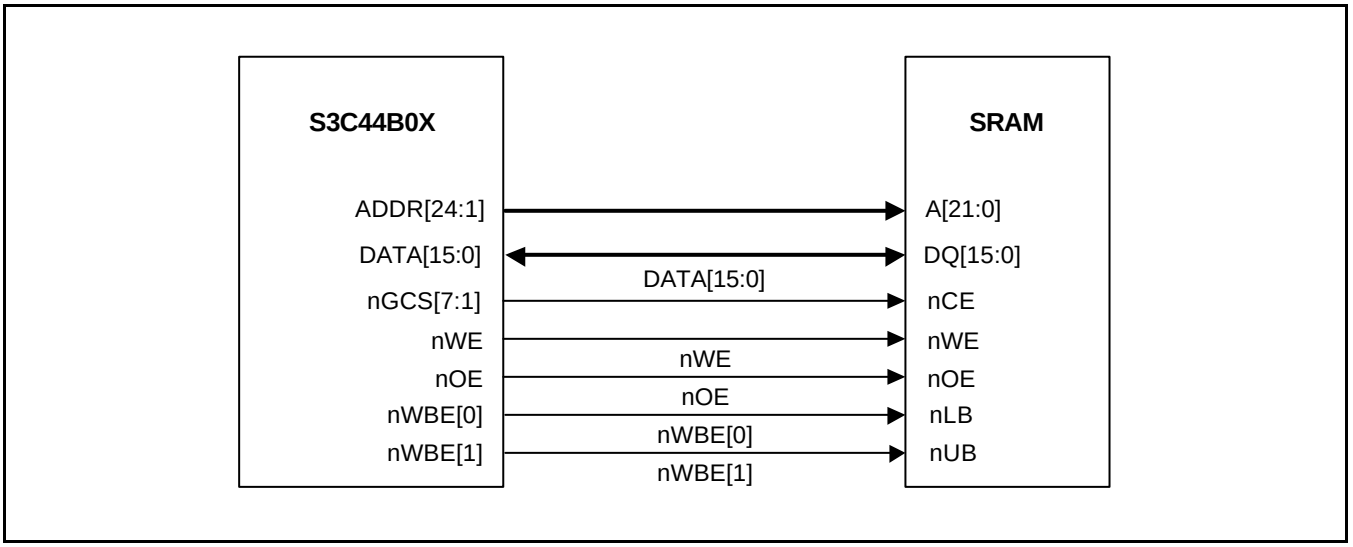


Figure 4-8. Half-word SRAM Banks Design with Half-word SRAM

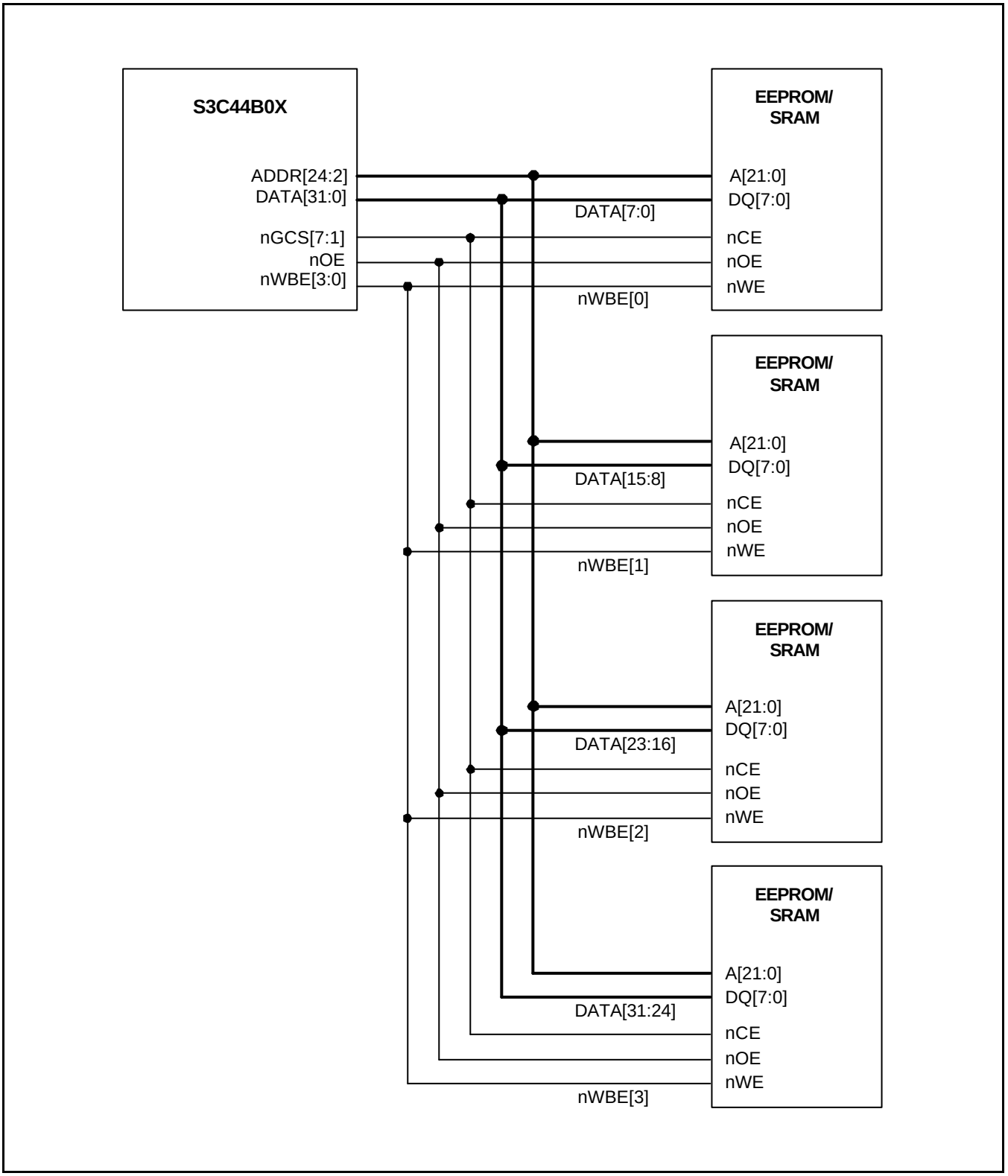


Figure 4-9. Word EEPROM/SRAM Banks Design

EDO DRAM BANKS DESIGN FOR S3C44B0X

The DRAM banks 6-7, can have a various width of data bus, and the bus width is controlled by S/W, A BWSCON special register set. A sample design for DRAM bank 6-7 is shown in Figure 4-10 and Figure 4-11.

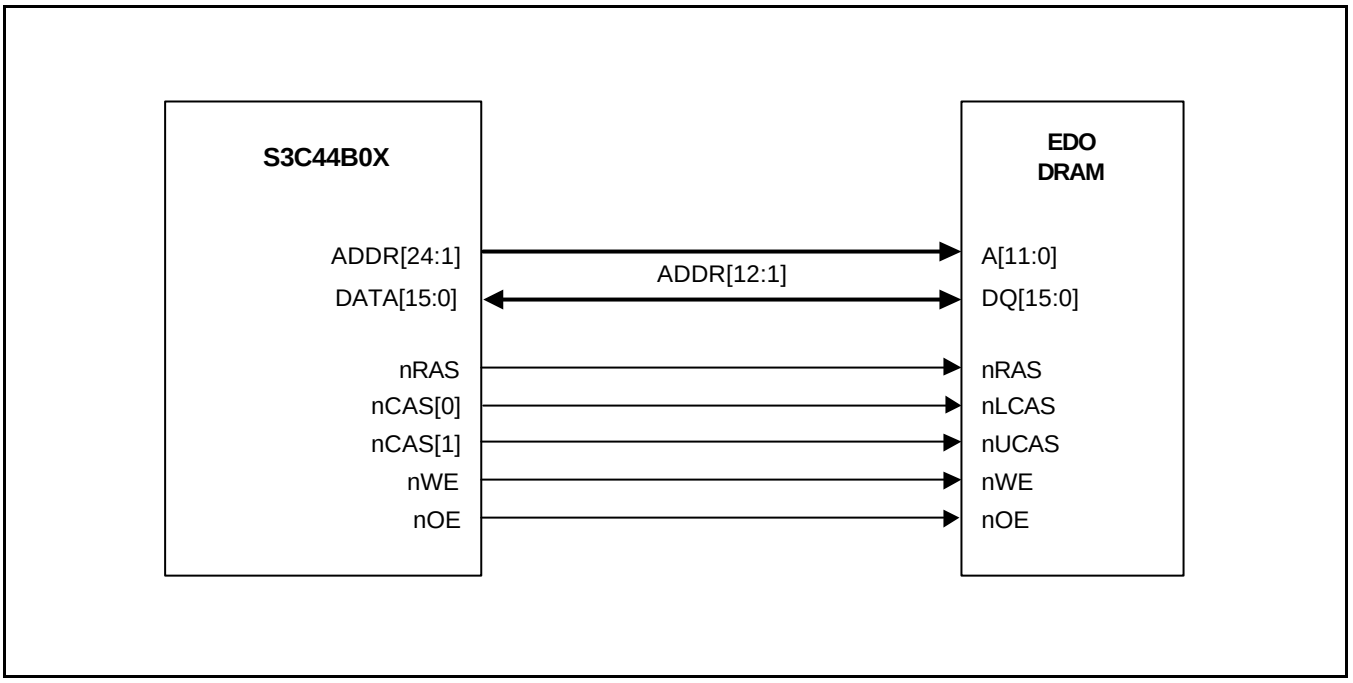


Figure 4-10. Half-Word EDO/Normal DRAM Banks Design

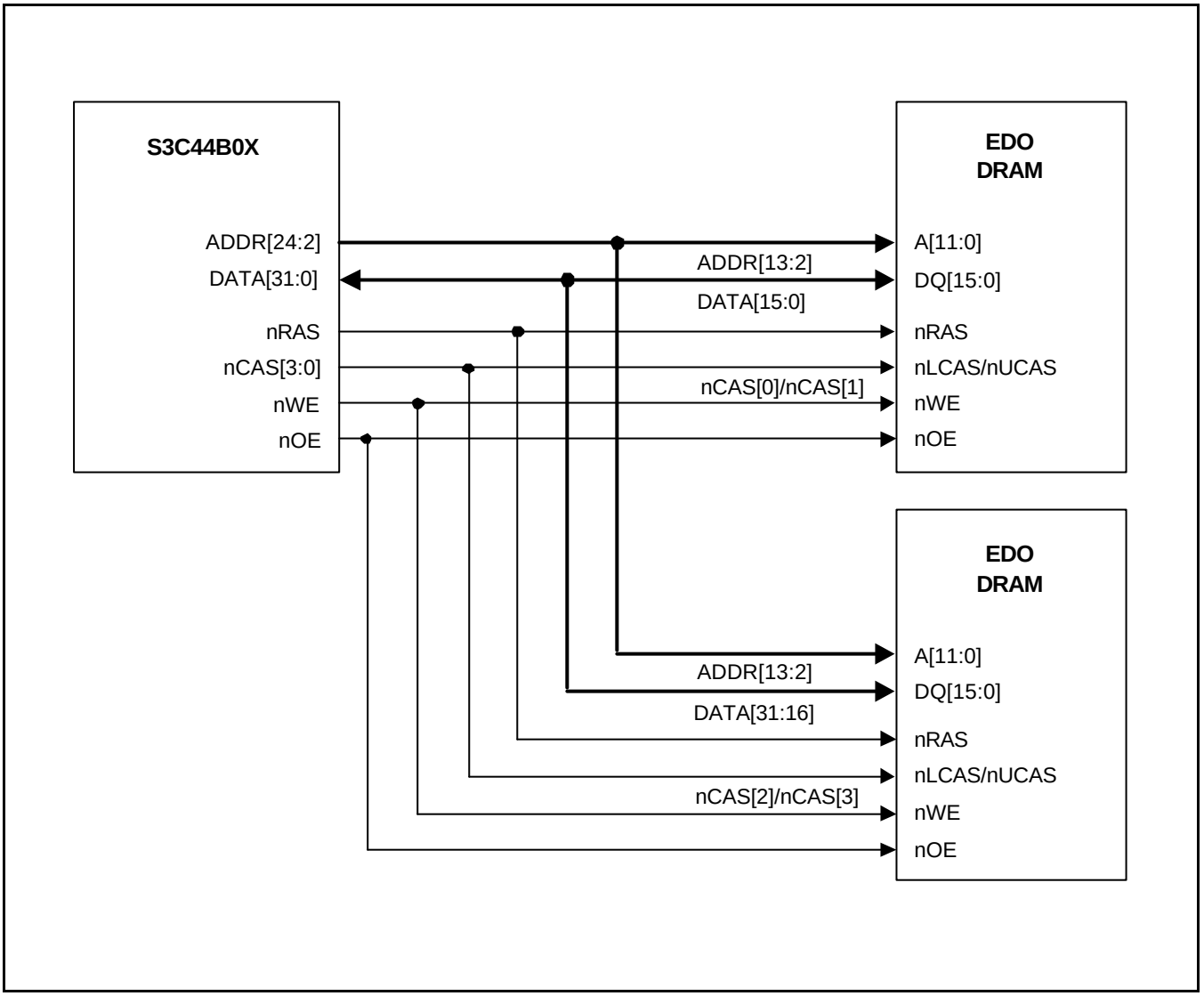


Figure 4-11. word EDO/Normal DRAM Bank

SDRAM BANKS DESIGN FOR S3C44B0X

The S3C44B0X Synchronous DRAM interface features are as follows :

- Maximum column address of SDRAM: 10 bit
- CAS latency: 2/3 cycle

Table 4-3. SDRAM Bank Address configuration

Bank Size	Bus Width	Base Component	Memory Configuration	Bank Address
2MByte	x8	16Mbit	(1M x 8 x 2Bank) x 1	A20
	x16		(512K x 16 x 2B) x 1	
4MB	x8	16Mb	(2M x 4 x 2B) x 2	A21
	x16		(1M x 8 x 2B) x 2	
	x32		(512K x 16 x 2B) x 2	
8MB	x16	16Mb	(2M x 4 x 2B) x 4	A22
	x32		(1M x 8x 2B) x 4	
	x8	64Mb	(4M x 8 x 2B) x 1	
	x8		(2M x 8 x 4B) x 1	A[22:21]
	x16		(2M x 16 x 2B) x 1	A22
	x16		(1M x 16 x 4B) x 1	A[22:21]
	x32		(512K x 32 x 4B) x 1	
16MB	x32	16Mb	(2M x 4 x 2B) x 8	A23
	x8	64Mb	(8M x 4 x 2B) x 2	
	x8		(4M x 4 x 4B) x 2	A[23:22]
	x16		(4M x 8 x 2B) x 2	A23
	x16		(2M x 8 x 4B) x 2	A[23:22]
	x32		(2M x 16 x 2B) x 2	A23
	x32		(1M x 16 x 4B) x 2	A[23:22]
	x8	128Mb	(4M x 8 x 4B) x 1	
	x16		(2M x 16 x 4B) x 1	
32MB	x16	64Mb	(8M x 4 x 2B) x 4	A24
	x16		(4M x 4 x 4B) x 4	A[24:23]
	x32		(4M x 8 x 2B) x 4	A24
	x32		(2M x 8 x 4B) x 4	A[24:23]
	x16	128Mb	(4M x 8 x 4B) x 2	
	x32		(2M x 16 x 4B) x 2	
	x8	256Mb	(8M x 8 x 4B) x 1	
	x16		(4M x 16 x 4B) x 1	

The required SDRAM interface pin is CKE, SCLK, nSCS[1:0], nSCAS, nSRAS, DQM[3:0], ADDR[12]/AP. The sample design with SDRAM is shown in Figure 4-12, and Figure 4-13.

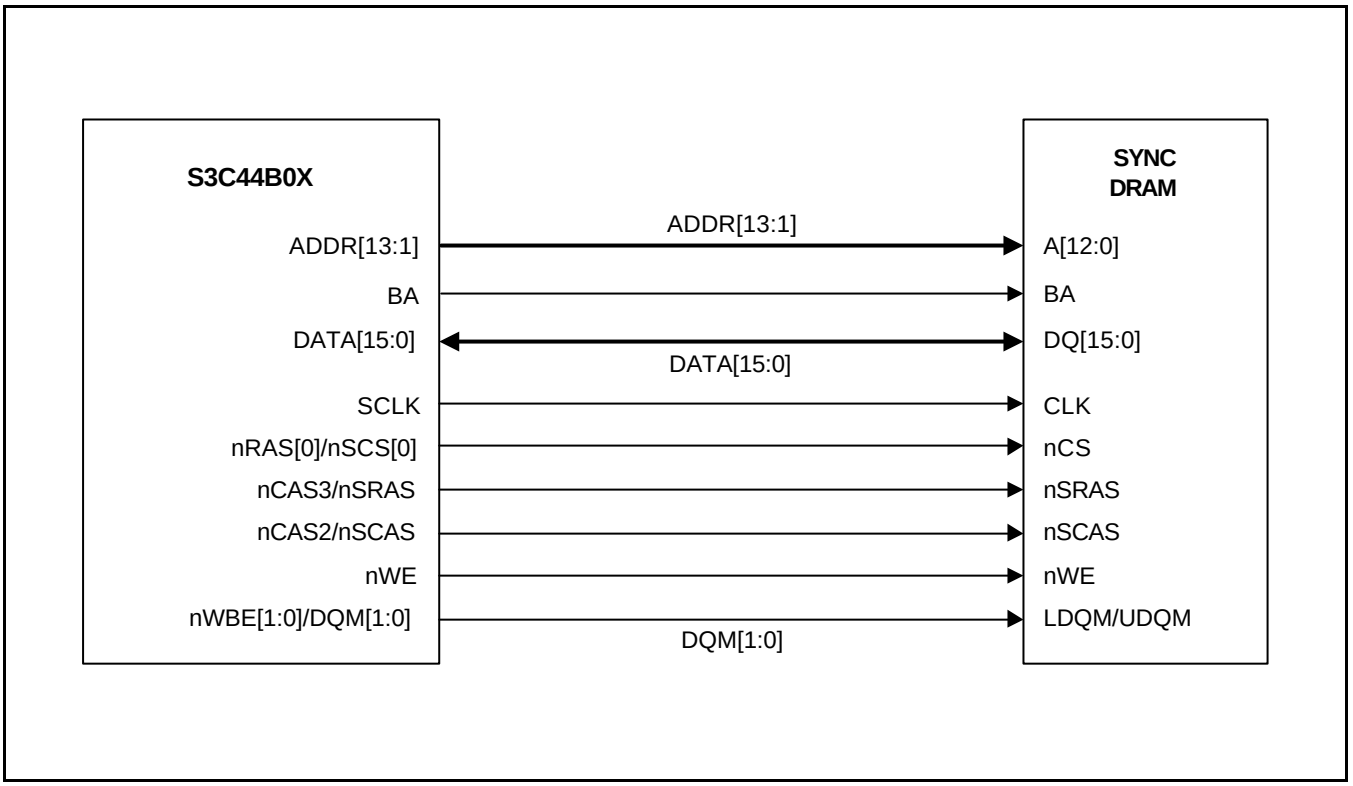


Figure 4-12. Half-word SDRAM Design with Half-word Component

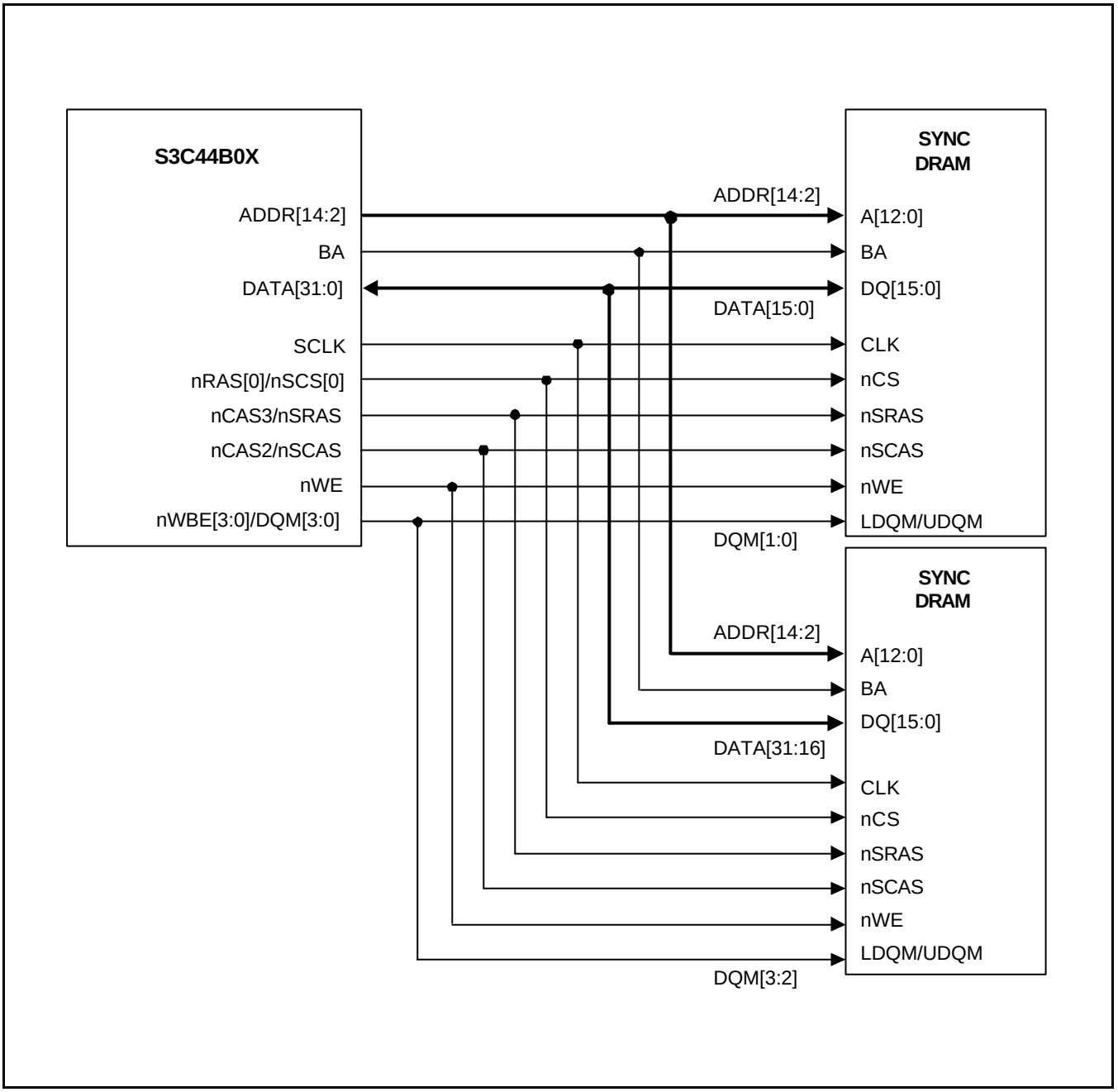


Figure 4-13. Word SDRAM Design with Half-word Component

I/O PORT CONFIGURATION

S3C44B0X has multiplexed input/output/function port pins. The SMDK41100 demo board uses only some functions, then some pins are float or some pins have selectable function port with 0 ohm resistor, therefore users must define the pin's configuration and attach the 0 ohm resistor on the proper place before running the main program.

For reducing the power consumption in SMDK41100, the port state and usage of internal pull-up resistor are decided very carefully. The followings are the sample port configuration for SMDK41100.

Port A : Memory address pins

Port	H/W connection	@Normal	@Stop	@Idle
PA0	OPEN	ADDR0		
PA1	ADDR16	ADDR16		
PA2	ADDR17	ADDR17		
PA3	ADDR18	ADDR18		
PA4	ADDR19	ADDR19		
PA5	ADDR20	ADDR20		
PA6	BA0	ADDR21		
PA7	BA1	ADDR22		
PA8	OPEN	ADDR23		
PA9	OPEN	ADDR24		
PDATA	**–**	**–**		
PCONA	**–**	**0x3ff**		

Port B : Memory control and output(LED) pins

Port	H/W Connection	@Normal	@Stop(data)	@Idle
PB0	SCKE	SCKE		
PB1	SCLK	SCLK		
PB2	nSCAS	nSCAS		
PB3	nSRAS	nSRAS		
PB4	DQM0	nWBE2/nBE2/DQM2		
PB5	DQM1	nWBE3/nBE3/DQM3		
PB6	OPEN	nGCS1		
PB7	OPEN	nGCS2		
PB8	OPEN	nGCS3		
PB9	LED	Output	Output(High)	Output
PB10	LED	Output	Output(High)	Output
PDATB	**–**	**–**	**0x600**	**–**
PCONB	**–**	**0x1ff**		

Port C : IIS, LCD data and UART control pins.

Not used function pins are defined input port with pull-up resistor enabled to reduce the power consumption. If the UART function is not used the pins should be disabled pull-up resistor, output signal(Tx, nRTS) is defined output port and input signal(Rx, nCTS) is defined input port avoid conflicting the signals of MAX3232.

Port	H/W Connection	@Normal(data, pull-up)	@Stop(data, pull-up)	@Idle(data, pull-up)
PC0	IISLRCK	Input(pull-up enable)		
PC1	IISDO	Input(pull-up enable)		
PC2	IISDI	Input(pull-up enable)		
PC3	IISCLK	Input(pull-up enable)		
PC4	VD7	Input(pull-up enable)		
PC5	VD6	Input(pull-up enable)		
PC6	VD5	Input(pull-up enable)		
PC7	VD4	Input(pull-up enable)		
PC8	OPEN	Input(pull-up enable)		
PC9	OPEN	Input(pull-up enable)		
PC10	nRTS1	Output(High, pull-up disable)		
PC11	nCTS1	Input(pull-up disable)		
PC12	TxD1	Output(High, pull-up disable)		
PC13	RxD1	Input(pull-up disable)		
PC14	nRTS0	Output(High, pull-up disable)		
PC15	nCTS0	Input(pull-up disable)		
PDATC	**–**	**0x5400**		
PUPC	**–**	**0xfc00**		
PCONC	**–**	**0x11100000**		

Port D : LCD data and LCD control pins.

Port	H/W Connection	@Normal(pull-up)	@Stop(data, pull-up)	@Idle(pull-up)
PD0	VD0	VD0(pull-up disable)	Output(High,pull-up disable)	VD0(pull-up disable)
PD1	VD1	VD1(pull-up disable)	Output(High,pull-up disable)	VD1(pull-up disable)
PD2	VD2	VD2(pull-up disable)	Output(High,pull-up disable)	VD2(pull-up disable)
PD3	VD3	VD3(pull-up disable)	Output(High,pull-up disable)	VD3(pull-up disable)
PD4	VCLK	VCLK(pull-up disable)	Output(High,pull-up disable)	VCLK(pull-up disable)
PD5	VLINE	VLINE(pull-up disable)	Output(High,pull-up disable)	VLINE(pull-up disable)
PD6	VM	VM(pull-up disable)	Output(High,pull-up disable)	VM(pull-up disable)
PD7	VFRAME	VFRAME(pull-up disable)	Output(High,pull-up disable)	VFRAME(pull-up disable)
PDATD	**–**	**–**	**0xff**	**–**
PUPD	**–**	**0xff**	**0xff**	**0xff**
PCOND	**–**	**0xaaaa**	**0x5555**	**0xaaaa**

Port E : UART control pins and ENDIAN pin.

PE8 port should be defined input port with disabled pull-up resistor, because the PE8 pin is connected to GND for little endian mode in SMDK41100 demo board.

Port	H/W Connection	@Normal(pull-up)	@Stop(pull-up)	@Idle(pull-up)
PE0	OPEN	Input(pull-up enable)		
PE1	TxD0	TxD0(pull-up disable)		
PE2	RxD0	RxD0(pull-up disable)		
PE3	OPEN	Input(pull-up enable)		
PE4	OPEN	Input(pull-up enable)		
PE5	OPEN	Input(pull-up enable)		
PE6	OPEN	Input(pull-up enable)		
PE7	OPEN	Input(pull-up enable)		
PE8	ENDIAN	ENDIAN(pull-up disable)		
PDATE	**–**	**–**		
PUPE	**–**	**0x106**		
PCONE	**–**	**0x28**		

Port F : IIS and SIO control pins.

PF0 and PF1 pins should be defined disabled pull-up resistor input port, because these pins are connected pull-up resistors by hardware.

Port	H/W Connection	@Normal(pull-up)	@Stop(pull-up)	@Idle(pull-up)
PF0	IICSCL	Input(pull-up disable)		
PF1	IICSDA	Input(pull-up disable)		
PF2	OPEN	Input(pull-up enable)		
PF3	OPEN	Input(pull-up enable)		
PF4	OPEN	Input(pull-up enable)		
PF5	SIOTxD	Input(pull-up enable)		
PF6	SIORDY	Input(pull-up enable)		
PF7	SIORxD	Input(pull-up enable)		
PF8	SIOCLK	Input(pull-up enable)		
PDATF	**–**	**–**		
PUPF	**–**	**0x3**		
PCONF	**–**	**0x0**		

Port G : External interrupt pins.

The SMDK41100 demo board is using PG4 and PG5 ports as external interrupt ports with button.

Port	H/W Connection	@Normal(pull-up)	@Stop(pull-up)	@Idle(pull-up)
PG0	EINT0	Input(pull-up enable)		
PG1	EINT1	Input(pull-up enable)		
PG2	EINT2	Input(pull-up enable)		
PG3	EINT3	Input(pull-up enable)		
PG4	EINT4	EINT4(pull-up disable)		
PG5	EINT5	EINT5(pull-up disable)		
PG6	OPEN	Input(pull-up enable)		
PG7	OPEN	Input(pull-up enable)		
PDATG	**–**	**–**		
PUPG	**–**	**0x30**		
PCONG	**–**	**0xf00**		

Special pull-up resistor control register

In normal operation, the pull-up resistor should be disabled but in stop mode it is enabled for power consumption.

Connection	@Normal	@Stop	@Idle
SPUCR0	Pull-up disable	Pull-up enable	Pull-up disable
SPUCR1	Pull-up disable	Pull-up enable	Pull-up disable
Hz@STOP	Previsous state	High-impedance	Previsous state
SPUCR	**0x7**	**0x0**	**0x7**

NOTE: To reduce power consumption users shall consider the state of pins and refer to the table below.

Usage of Pins	Pull-up resistor + Data
Unused input port pins	Pull-up enable
Normal output port pins	Pull-up disable + Data High
Function(address, data, control) pins	Pull-up disable

MMC CARD INTERFACE WITH S3C44B0X SIO

For MMC card interface, The S3C44B0X SPI interface is used as the following example circuit:

- The nCS signal is implemented by general I/O port(GPE5).
- On SIORXD pin, the 50K internal pull-up resister is turned on.
- Please refer to end of this chapter, there is the sample code of MMC-SPI application .

Filename	File Descriptions	Page
makefile	Makefile for 44btest	4-30
spi.h	Header file for spi.c	4-32
44btest.c	Main program	4-33
spi.c	SPI test program	4-37

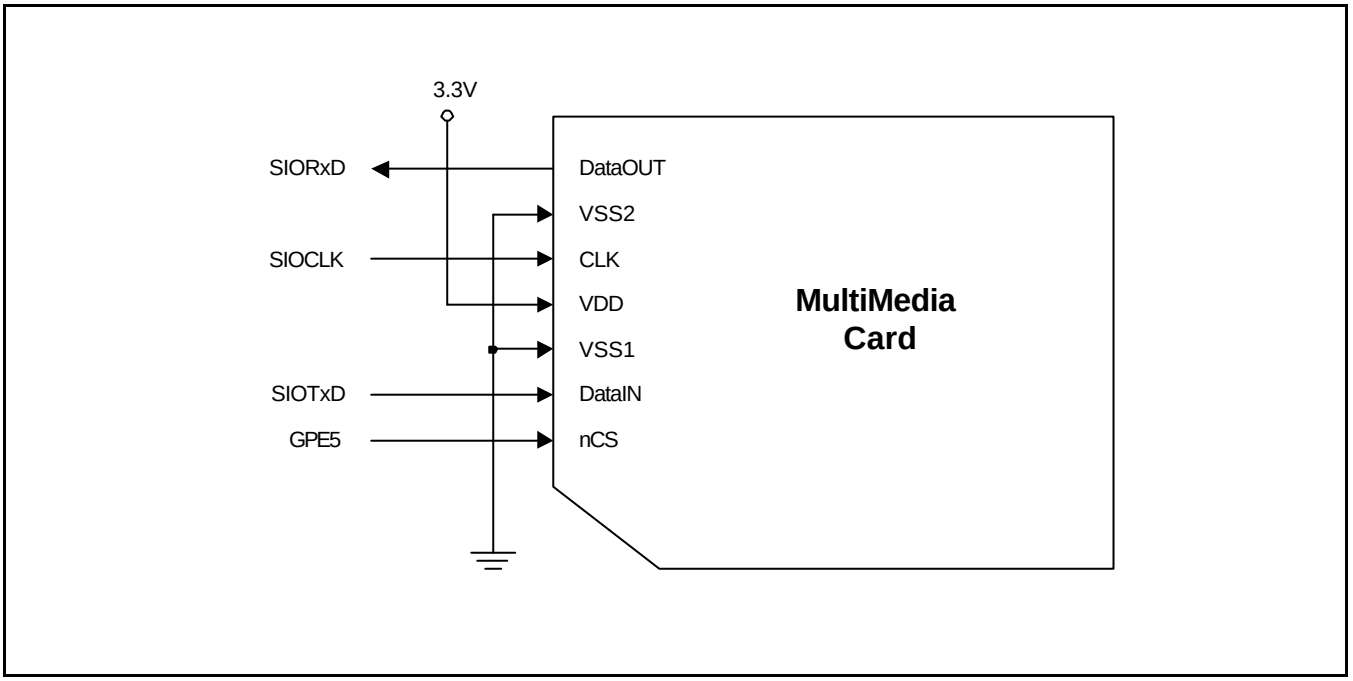


Figure 4-14. SPI connection with S3C44B0X

PC CARD(PCMCIA) INTERFACE APPLICATION USING CL-PD6710(CIRRUS LOGIC)

The PC card(PCMCIA card) can be interfaced with S3C44B0X using following components;

- CL-PD6710 from Cirrus logic
- TPS2211from Texas Instruments
- Please refer to end of this chapter, there is the sample code of PCMCIA application.

We had tested the PC card interface by accessing the CIS(card information structure) in the modem card as Figure 4-15. The following test code and schematics are attached in the end of this chapter.

Filename	File Descriptions	Page
pd6710.h	CL-PD6710 register definitions	4-49
pd6710.c	CL-PD6710 PC Card program	4-51

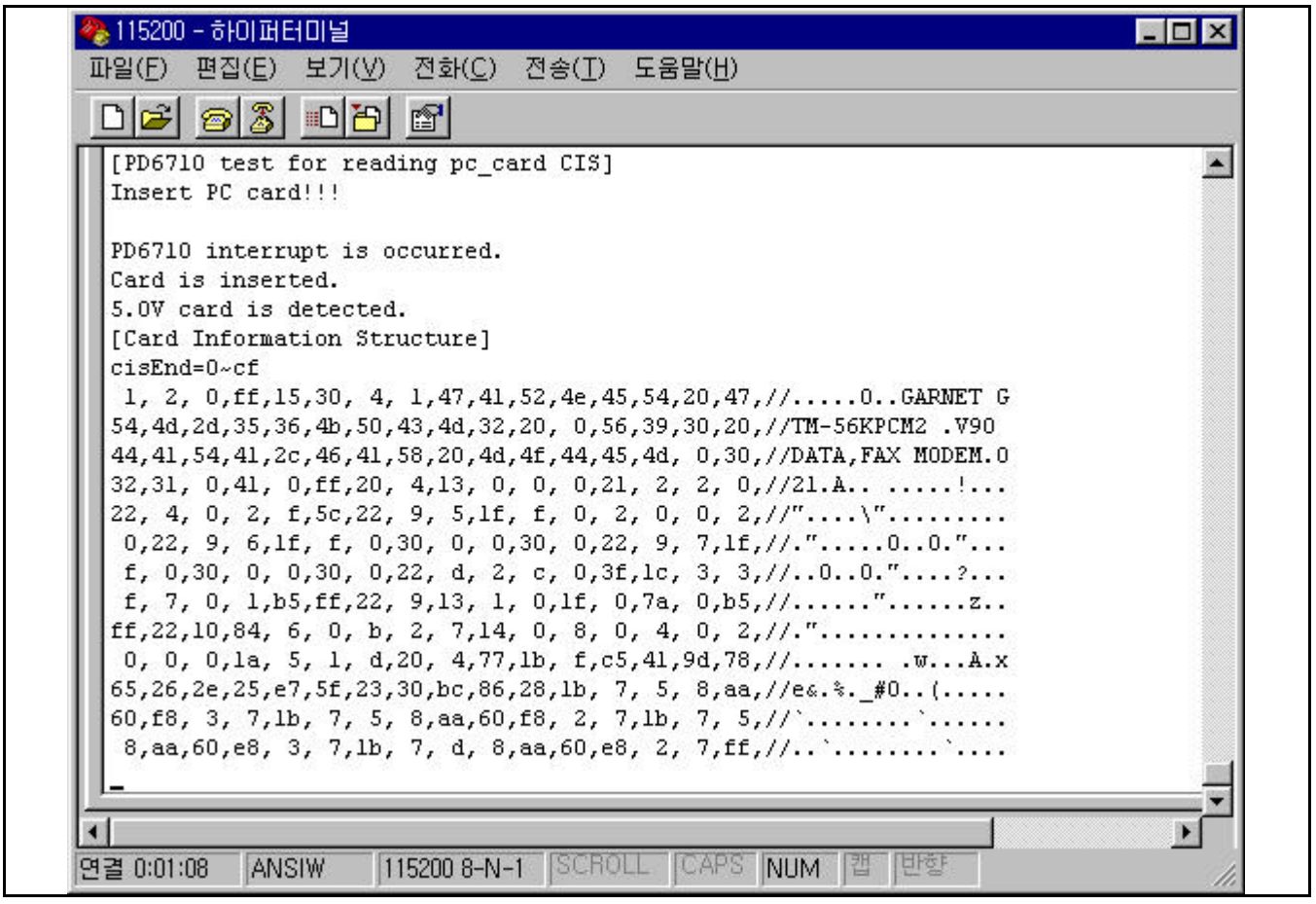


Figure 4-15. PC Card CIS Access Example on S3C44B0X

10BASE-T ETHERNET CONTROLLER(CS8900A) INTERFACE

The 10BASE-T Ethernet can be supported on S3C44B0X using following components.

- CS-8000A from Cirrus logic
- 821-M0542 transformer from E&E Magnetic products Ltd.(UNISTANDARD, +82-2-557-5355) or E2023 from Pulse (UNIQUEST, +82-2-562-8805)

We have not tested the CS8000A interface until this application note is published. So, only the schematic is attached in this application note. For source code for CS8900A, Please visit our web-site (http://www.samsungsemi.com) or send e-mail to purnnamu@sec.samsung.com.

AUDIO CODEC(UDA1341TS) CONNECTION WITH S3C44B0X

The S3C44B0X IIS interface example circuit is as follows :

- UDA1341TS from Philips Semiconductors.
- The L3 interface of Philips(L3MOD, L3CLOCK, L3DATA) is realized by general I/O port.
- Please refer to Chapter 3(page 3-111), the sample code of audio application which play PCM file with 44.1KHz sampling frequency.

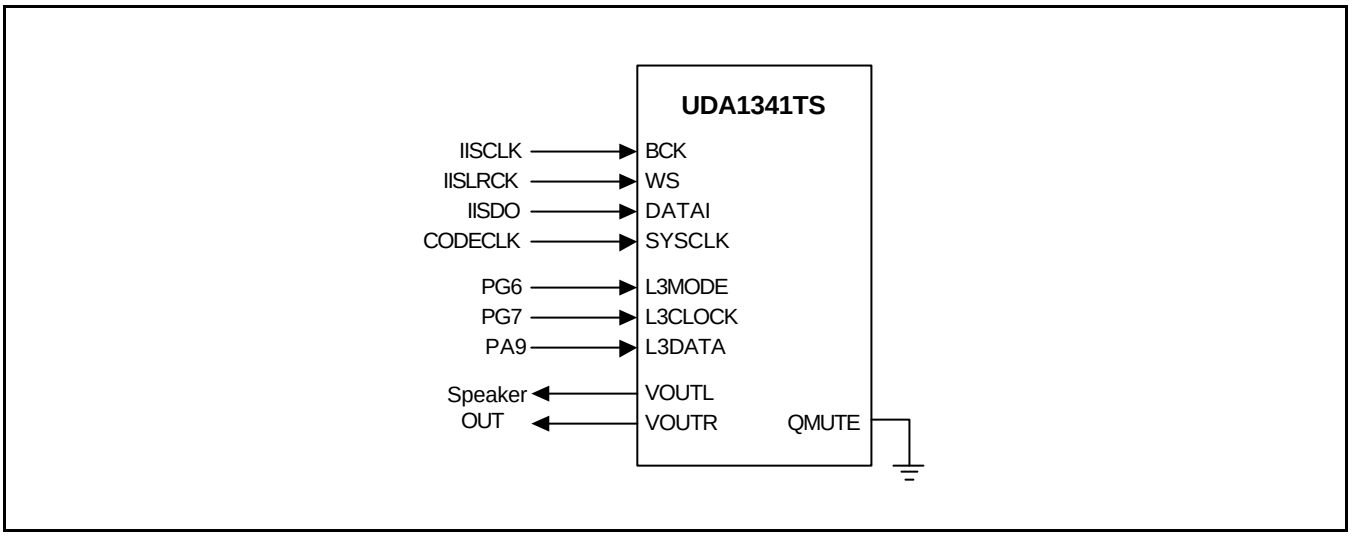


Figure 4-16. UDA1341TS connection with S3C44B0X

LCD CONNECTION WITH S3C44B0X

The S3C44B0X LCD interface example circuit is as follows :

- UG-32F04(320x240 mono STN LCD) from SAMSUNG DISPLAY DEVICES CO.,LTD (refer to Figure 4-17)
  - TL497CAN can be used to make VEE(-25V).

- UG-24U03A(320x240 mono STN LCD) from SAMSUNG DISPLAY DEVICES CO.,LTD (refer to Figure 4-18)
  - VEE is generated by the circuit on LCD module.
  - VL is 2.4V typically.
  - DISPON H : display on, L : display off
  - nEL_ON H : EL off L : EL on

- KHS038AA1AA-G24 (256 color STN LCD) from KYOCERA Co. (refer to Figure 4-19)
  - DISP signal can be made using I/O port, or power control circuit, or nRESET circuit.
  - V1-V5 can be made using the power circuit recommended by the LCD specification.

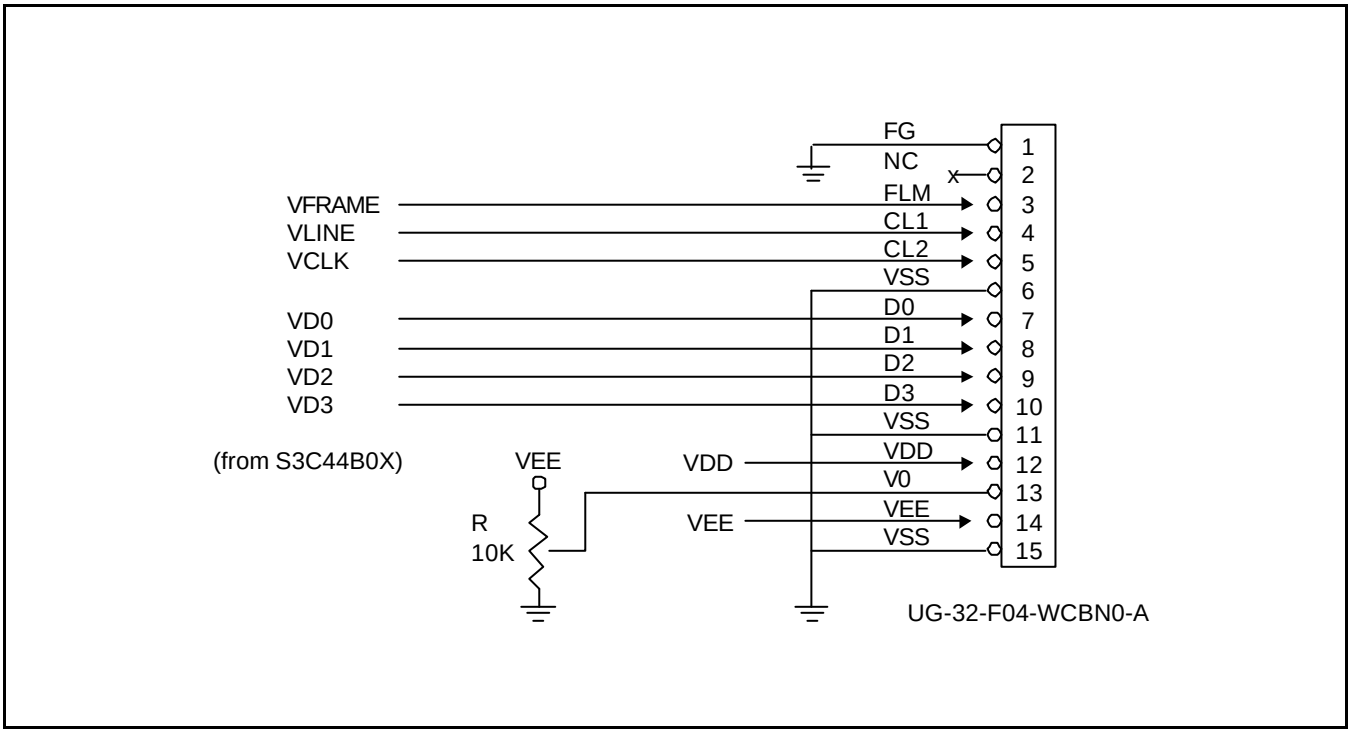


Figure 4-17. UG-32F04 connection with S3C44B0X(320x240 mono STN LCD)

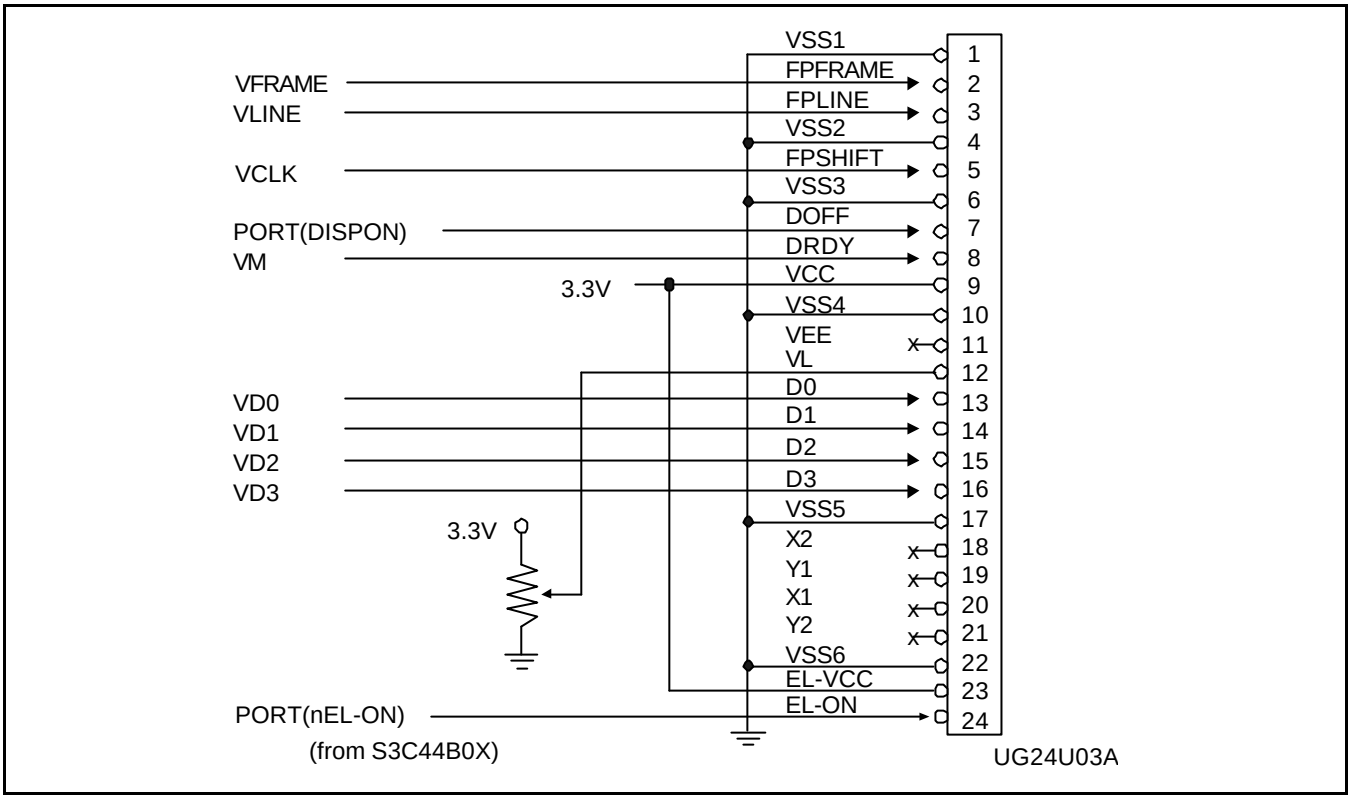


Figure 4-18. UG24U03A connection with S3C44B0X (320x240 mono STN LCD)

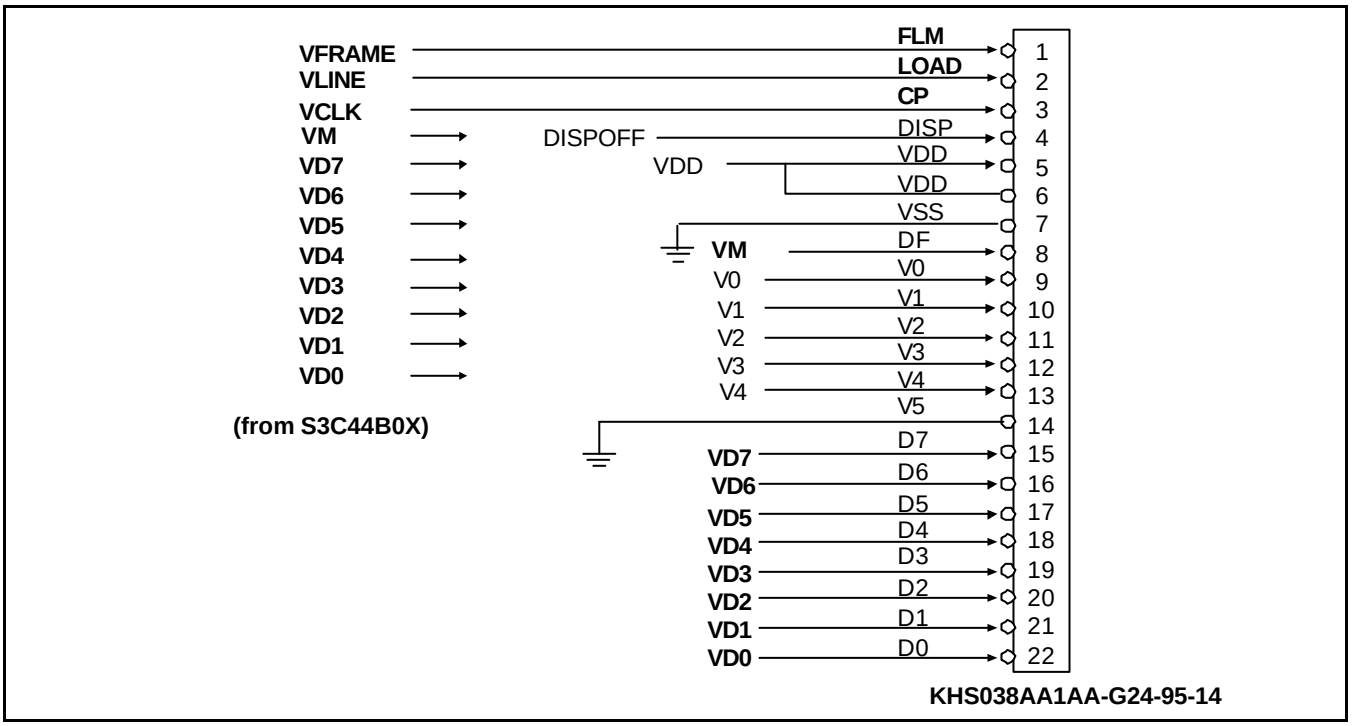


Figure 4-19. KHS038AA1AA-G24 connection with S3C44B0X (256 color STN LCD)

TSP(TOUCH SCREEN PANEL) INTERFACE CIRCUIT WITH S3C44B0X

Typically, TSP consists of 2 plane resisters (x-axis plane resister, y-axis plane resister). So, TSP device has 4 terminals. To read X coordinate, Q1, Q2 is turned on and Q3, Q4 is turned off. So, X coordinate value can be read out from AIN1 by ADC. To read Y coordinate, Q3, Q4 is turned on and Q1, Q2 is turned off. So, ADC can read out Y coordinate value from AIN0. The PEN_INT can be used to know whether or not the TSP is touched.

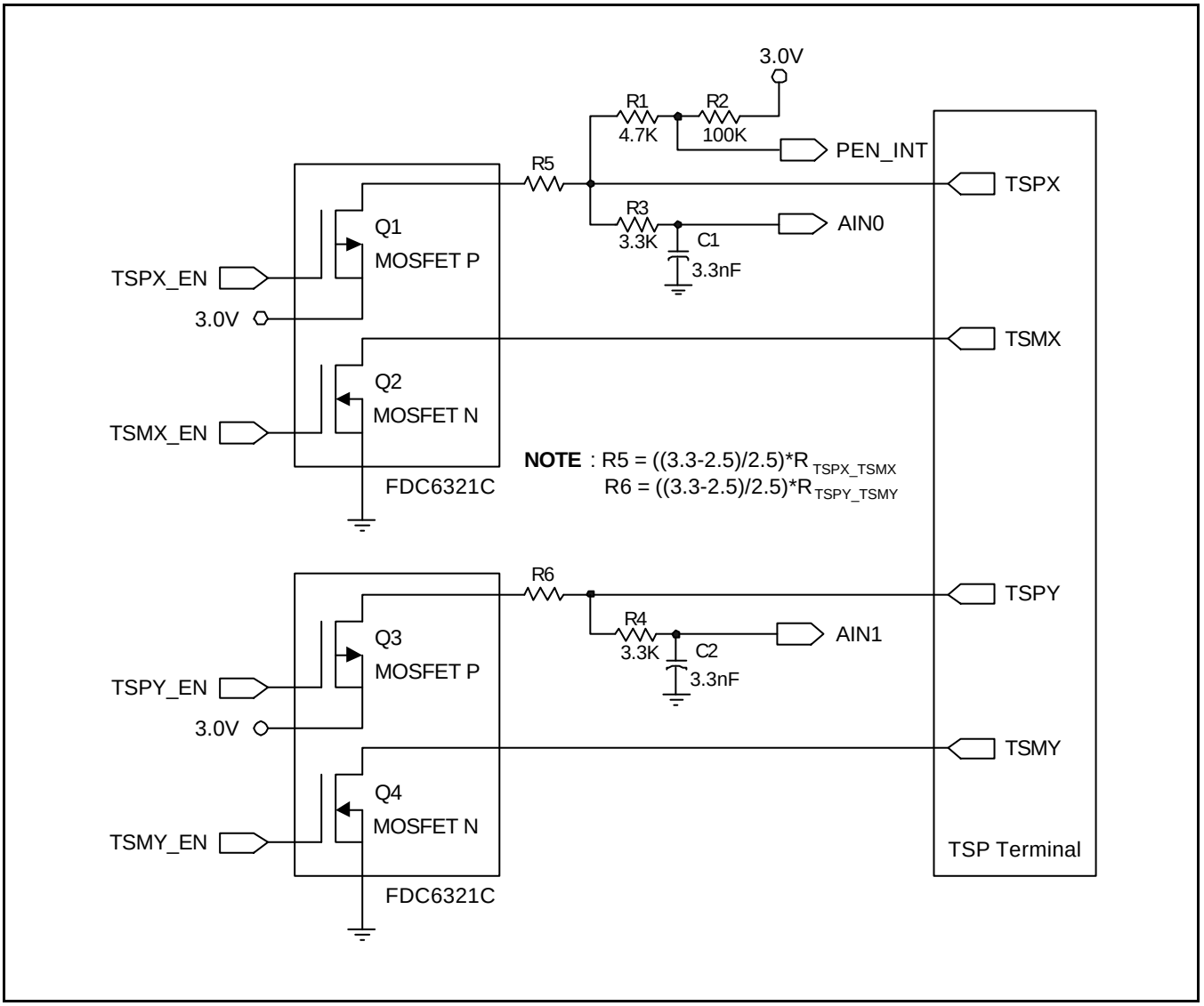


Figure 4-20. TSP Interface Circuit with S3C44B0X

SYSTEM DESIGN WITH DEBUGGER SUPPORT

EmbeddedICE Macrocell and EmbeddedICE Interface

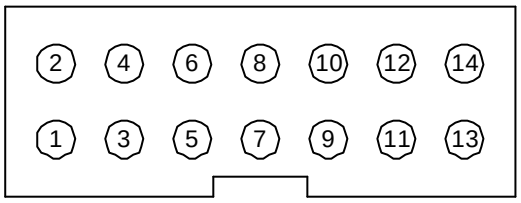
The S3C44B0X has an EmbeddedICE macrocell that provides debug support fro ARM cores. The EmbeddedICE macrocell is programmed in serial using the TAP(Test Access Port) controller on the S3C44B0X. The EmbeddedICE interface is a JTAG protocol conversion unit. It translates a debug protocol message generated by the debugger into a JTAG signal which is sent to the built-in serial and parallel ports.

JTAG port for EmbeddedICE Interface

When you build a system with the S3C44B0X EmbeddedICE interface, you should design a JTAG port for EmbeddedICE interface. Usually, the interface connector is a 14-way box header, and this plug is connected to the EmbeddedICE interface module using 14-way IDC cable.

The JTAG port signals, nTRST,TDI,TMS,TCK have to be connected pulled-up register(10K ohm) externally. **When you operate normal mode without EmdeddedICE, nRESET signal on S3C44B0X is connected nTRST via JP1 (jumper1). In debugger mode, nRESET signal on S3C44B0X is surely seperated nTRST via JP1 (jumper1).**

The pin configuration and a sample design are described in Figure 4-21, 4-22, respectively.



Pin	Name	Function
1,13	SPU	Connected to VDD through 33 or 0 ohm resister*
3	nTRST	Test reset, active low (connected pull-up reg.)
5	TDI	Test data in (connected pull-up reg.)
7	TMS	Test mode select (connected pull-up reg.)
9	TCK	Test clock (connected pull-up reg.)
11	TDO	Test data out
12	nICERST	Connected to VDD through 10K ohm resister
2, 4, 6, 8, 10, 14	VSS	System Ground

NOTE: 33 ohm for EmbeddedICE, 0 ohm for Multi-ICE, pullup Reg.:10K ohm

Figure 4-21. EmbeddedICE Interface JTAG Connector

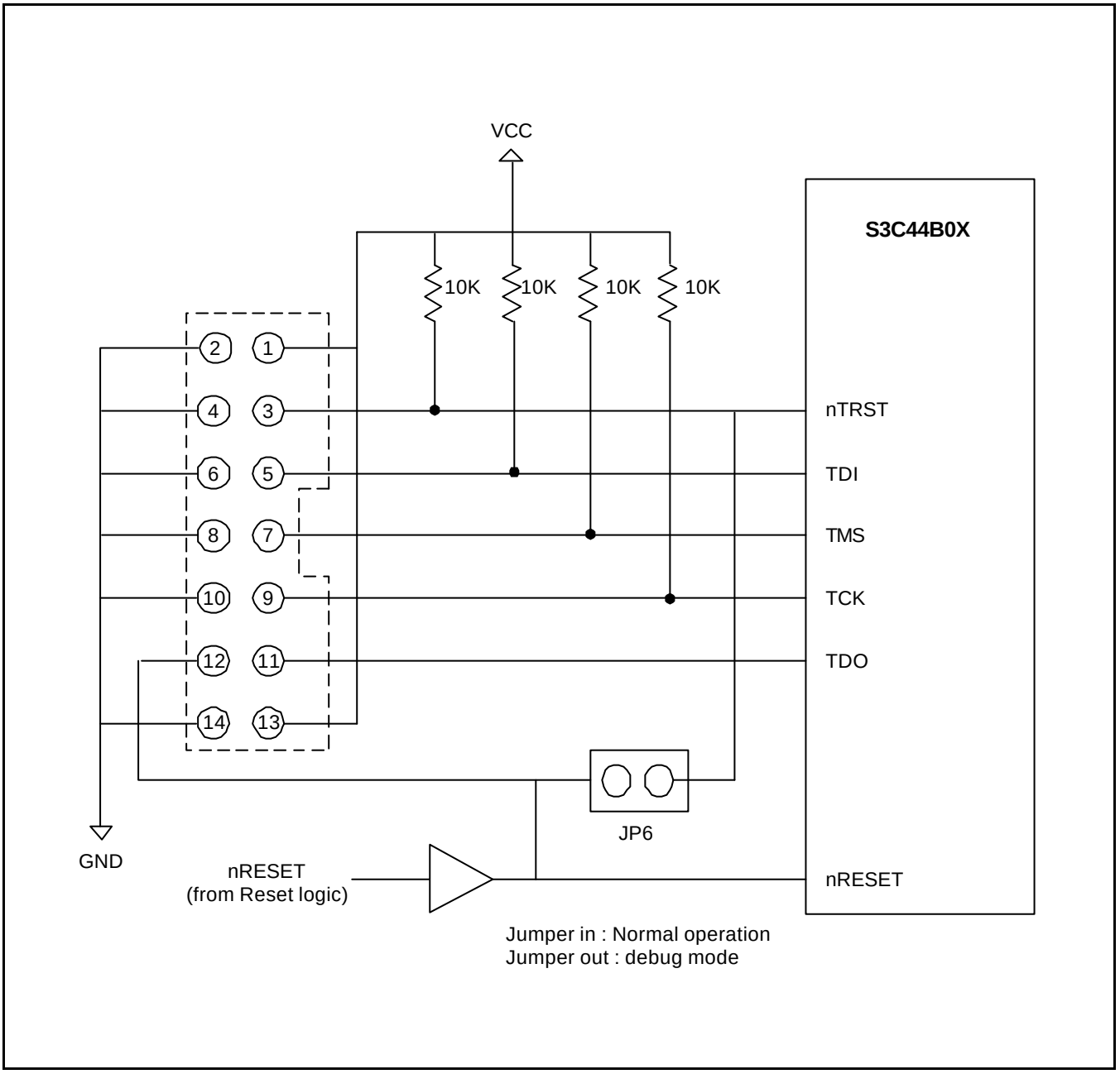


Figure 4-22. EmbeddedICE Interface Design Example

CHECK LIST FOR SYSTEM DESIGN WITH S3C44B0X

When you design a system with the S3C44B0X, you should check a number of items to build a good system. The check list is described below.

- The OM[3:0] and ENDIAN pin have to be configured.
- To run the CPU without using JTAG(ICE), connect nTRST and nRESET pin.
- If EXTCLK pin is used for MCLK, XTAL0 has to be connected to VDD. If XTAL0 pin is used for MCLK, EXTCLK has to be connected to VDD.
- If an input pin is unused, connect the pin to VDD or GND. If the pin is float, S3C44B0X may not operate.
- nGCS6,7 do not support DRAM & SDRAM combination,

  Please configure memory type to below combination in bank6 & 7:
  DRAM & DRAM, SDRAM & SDRAM , SRAM & SRAM, SRAM & DRAM, SRAM & SDRAM.

MMC SAMPLE CODE

MMC : Makefile

```
#### File Definition ####
PRJ = 44btest
INIT= 44binit
AM1 = 44blib_a
CM1 = 44blib
CM2 = spi

#### Destination path Definition ####
SRC=.
INC=..\inc
OBJ=..\obj
ERR=..\err
BIN=..\bin

#### ARM tool Definition ####
ARMLINK= armlink
ARMASM = armasm
ARMCC  = armcc

#### Option Definition ####
LFLAGS = -ro-base 0xc000000 -rw-base 0x0c040000 -elf -map -xref \
        -list $(BIN)\list.txt -first $(INIT).o(Init)
AFLAGS = -li -apcs /noswst -cpu ARM7TDMI
CFLAGS = -c -g+ -li -apcs /noswst -cpu ARM7TDMI

#If you doesn't debug,use following CFLAGS for more faster operation.
#CFLAGS = -c -g- -fc -apcs 3/32bit/noswst/nofp -li -processor ARM7TM -arch 4T

#If you doesn't debug,use following CFLAGS for more faster operation.
#CFLAGS = -c -g- -fc -apcs 3/32bit/noswst/nofp -li -processor ARM7TM -arch 4T

#### Object combine Definition ####
OBJS = $(OBJ)\$(PRJ).o      $(OBJ)\$(INIT).o $(OBJ)\$(AM1).o\
           $(OBJ)\$(CM1).o   $(OBJ)\$(CM2).o

all: $(PRJ).elf

clean:
       del $(OBJ)\*.o

$(PRJ).elf: $(OBJS)
       del $(BIN)\$(PRJ).bin
       del $(BIN)\$(PRJ).elf
       $(ARMLINK) $(LFLAGS) -o $(BIN)\$(PRJ).elf $(OBJS)
       fromelf -nodebug -nozeropad $(BIN)\$(PRJ).elf -bin $(BIN)\$(PRJ).bin
       copy ..\bin\$(PRJ).bin .
       copy ..\bin\$(PRJ).elf .
#For SDT2.5 fromelf -nodebug -nozeropad $(BIN)\$(PRJ).elf -bin $(BIN)\$(PRJ).bin
```

```
#For ADS1.0 fromelf -nodebug -bin -output $(BIN)\$(PRJ).bin $(BIN)\$(PRJ).elf

$(OBJ)\$(PRJ).o : $(SRC)\$(PRJ).c $(INC)\44b.h $(INC)\44blib.h $(INC)\option.h makefile
       del $(OBJ)\$(PRJ).o
       del $(ERR)\$(PRJ).err
       $(ARMCC) $(CFLAGS) $(SRC)\$(PRJ).c -o $(OBJ)\$(PRJ).o -Errors $(ERR)\$(PRJ).err

$(OBJ)\$(INIT).o: $(SRC)\$(INIT).s $(INC)\option.a $(INC)\memcfg.a makefile
       del $(OBJ)\$(INIT).o
       del $(ERR)\$(INIT).err
       $(ARMASM) $(AFLAGS) $(SRC)\$(INIT).s -o $(OBJ)\$(INIT).o -Errors $(ERR)\$(INIT).err

$(OBJ)\$(AM1).o: $(SRC)\$(AM1).s makefile
       del $(OBJ)\$(AM1).o
       del $(ERR)\$(AM1).err
       $(ARMASM) $(AFLAGS) $(SRC)\$(AM1).s -o $(OBJ)\$(AM1).o -Errors $(ERR)\$(AM1).err

$(OBJ)\$(CM1).o: $(SRC)\$(CM1).c $(INC)\$(CM1).h $(INC)\44b.h $(INC)\44blib.h $(INC)\option.h
makefile
       del $(OBJ)\$(CM1).o
       del $(ERR)\$(CM1).err
       $(ARMCC) $(CFLAGS) $(SRC)\$(CM1).c -o $(OBJ)\$(CM1).o -Errors $(ERR)\$(CM1).err

$(OBJ)\$(CM2).o: $(SRC)\$(CM2).c $(INC)\$(CM2).h $(INC)\44b.h $(INC)\44blib.h $(INC)\option.h
makefile
       del $(OBJ)\$(CM2).o
       del $(ERR)\$(CM2).err
       $(ARMCC) $(CFLAGS) $(SRC)\$(CM2).c -o $(OBJ)\$(CM2).o -Errors $(ERR)\$(CM2).err
```

MMC : option.a

MMC : memcfg.a

MMC : 44binit.s

MMC : 44blib_a.s

MMC : 44b.h

MMC : 44blib.h

MMC : def.h

MMC : option.h

MMC : 44blib.c

These codes are exactly same with that in chapter3.

MMC : spi.h

```
#ifndef __SPI_H__
#define __SPI_H__

void _WtCMD(char cmd, int addr);

void __irq Sio_Int(void);

short int _RdRSP(char type);
char _WtDAT(char pattern);
void _RdDAT(void);
void TR_Buf_init(void);

void Card_sel(void);

#define Card_desel() rPDATE=0x20;

//-- Read register functions
void Rd_CSD(void);
void Rd_CID(void);
void Chk_Status(void);

//-- Read data functions
void Rd_Slg_blk(void);

//-- Write data functions
void Wt_Slg_blk(void);

//-- Set protection functions
void Set_Prt(void);
void Clr_Prt(void);
void Chk_Prt(void);

//-- Erase functions
void Er_Sector(void);
void Er_Group(void);

//-- Rx buffer functions
void Flush_Rx_buf(void);
void View_Rx_buf(void);
void View_Tx_buf(void);


#endif /*__SPI_H__*/
```

MMC : 44btest.c

```
#include <string.h>
#include "..\inc\option.h"
#include "..\inc\44b.h"
#include "..\inc\44blib.h"
#include "..\inc\def.h"
#include "..\inc\spi.h"

void AutoTest(void);
void Isr_Init(void);
void HaltUndef(void);
void HaltSwi(void);
void HaltPabort(void);
void HaltDabort(void);


void * function[][2]=
{
/*    //-- Read register test
    (void *)Rd_CSD,         "Rd CSD reg    ",
    (void *)Rd_CID,         "Rd CID reg    ",
    (void *)Chk_Status,     "Rd Status reg ",
*/
    //-- Write data test
    (void *)Wt_Slg_blk,     "Wt sgl blk    ",

    //-- Read data test
    (void *)Rd_Slg_blk,     "Rd sgl blk    ",

/*    //-- Set protection test
    (void *)Set_Prt,        "Set wt prt    ",
    (void *)Clr_Prt,        "Clr wt prt    ",
    (void *)Chk_Prt,        "Chk wt prt    ",

    //-- Erase test
    (void *)Er_Sector,      "Ers sector    ",
    (void *)Er_Group,       "Ers group     ",
*/
    //-- Rx buffer control
    (void *)Flush_Rx_buf,   "Flush Rx buf  ",
    (void *)View_Rx_buf,    "View Rx buf   ",
    (void *)View_Tx_buf,    "View Tx buf   ",

    0,0
};

void Main(void)
{
    int i;

    rSYSCFG=CACHECFG;

    //_ctype_init(); //to use ctype.h
    Port_Init();
```

```
    Isr_Init();
    Uart_Init(0,115200);
    Uart_Select(0);
    Delay(0);  //calibrate Delay()
    spi_card_init();
```

```
    while(1)
    {
       i=0;
       Uart_Printf("\n\nS3C44B0X SPI Test Program Ver 0.00 MCLK=5MHz(SLOW)\n\n");

       while(1)
       {   //display menu
           Uart_Printf("%2d:%s",i,function[i][1]);
           i++;
           if((int)(function[i][0])==0)
           {
              Uart_Printf("\n");
              break;
           }
           if((i%4)==0)
              Uart_Printf("\n");
       }
       Uart_Printf("\nSelect the function to test?");

       i=Uart_GetIntNum();
       Uart_Printf("\n");
       if(i>=0 && (i<(sizeof(function)/8)) )
           ( (void (*)(void)) (function[i][0]) )();
    }
}

void Isr_Init(void)
{
    pISR_UNDEF=(unsigned)HaltUndef;
    pISR_SWI  =(unsigned)HaltSwi;
    pISR_PABORT=(unsigned)HaltPabort;
    pISR_DABORT=(unsigned)HaltDabort;

    rINTCON=0x5;       // Non-vectored,IRQ enable,FIQ disable
    rINTMOD=0x0;       // All=IRQ mode
    rINTMSK=BIT_GLOBAL;       // All interrupt is masked.
}

void HaltUndef(void)
{
    Uart_Printf("Undefined instruction exception!!!\n");
    while(1);
}

void HaltSwi(void)
{
    Uart_Printf("SWI exception!!!\n");
    while(1);
}

void HaltPabort(void)
{
    Uart_Printf("Pabort exception!!!\n");
    while(1);
}

void HaltDabort(void)
```

```
{
    Uart_Printf("Dabort exception!!!\n");
    while(1);
}
```

MMC : spi.c

```
#include <string.h>
#include "..\inc\44b.h"
#include "..\inc\44blib.h"
#include "..\inc\spi.h"
#include "..\inc\def.h"

#define BLOCK_LEN   (512)
//-- Command array
char CMD0[6] ={0x40,0x0,0x0,0x0,0x0,0x95};//GO_IDLE_STATE
char CMD1[6] ={0x41,0x0,0x0,0x0,0x0,0xff};//SEND_OP_COND
char CMD9[6] ={0x49,0x0,0x0,0x0,0x0,0xff};//SEND_CSD
char CMD10[6]={0x4a,0x0,0x0,0x0,0x0,0xff};//SEND_CID
char CMD13[6]={0x4d,0x0,0x0,0x0,0x0,0xff};//SEND_STATUS
char CMD16[6]={0x50,0x0,0x0,0x0,0x0,0xff};//SET_BLOCKLEN
char CMD17[6]={0x51,0x0,0x0,0x0,0x0,0xff};//READ_SINGLE_BLOCK
char CMD24[6]={0x58,0x0,0x0,0x0,0x0,0xff};//WRITE_BLOCK
char CMD28[6]={0x5c,0x0,0x0,0x0,0x0,0xff};//SET_WRITE_PROT
char CMD29[6]={0x5d,0x0,0x0,0x0,0x0,0xff};//CLR_WRITE_PROT
char CMD30[6]={0x5e,0x0,0x0,0x0,0x0,0xff};//SEND_WRITE_PROT
char CMD32[6]={0x60,0x0,0x0,0x0,0x0,0xff};//TAG_SECTOR_STATRT
char CMD33[6]={0x61,0x0,0x0,0x0,0x0,0xff};//TAG_SECTOR_END
char CMD34[6]={0x62,0x0,0x0,0x0,0x0,0xff};//UNTAG_SECTOR
char CMD35[6]={0x63,0x0,0x0,0x0,0x0,0xff};//TAG_ERASE_GROUP_START
char CMD36[6]={0x64,0x0,0x0,0x0,0x0,0xff};//TAG_ERASE_GROUP_END
char CMD37[6]={0x65,0x0,0x0,0x0,0x0,0xff};//UNGAG_ERASE_GROUP
char CMD38[6]={0x66,0x0,0x0,0x0,0x0,0xff};//ERASE
char CMD59[6]={0x7b,0x0,0x0,0x0,0x0,0xff};//CRC_ON_OFF

volatile char endSioTR=0;
char Tx_buffer[BLOCK_LEN];   //512
char Rx_buffer[BLOCK_LEN];   //512

void __irq Sio_Int(void)
{
    rI_ISPC=BIT_SIO;
    endSioTR=1;
}

void TR_Buf_init(void)
{
    //-- Tx & Rx Buffer initialize
    int i;

    for(i=0;i<BLOCK_LEN;i++)
    {
        Tx_buffer[i]=0xa5;
        Rx_buffer[i]=0x0;
    }
}

void Rd_CSD(void)
{}
void Rd_CID(void)
{}
```

```
void Chk_Status(void)
{}
```

```
void Rd_Slg_blk(void)
{
    int addr;
    short int rsp=1;

    Uart_Printf("[SPI single block read test]\n");
    Uart_Printf("Input the start address of block to read\n");
    Uart_Printf("The address must be aligned 512bytes !!!\n");
    Uart_Printf("(Ex: 0x0, 0x200, 0x400, 0x600, 0x800, 0xa00....)\n");

    addr=Uart_GetIntNum();

    Card_sel();
    while(rsp)
    {
        _WtCMD(17,addr);
       rsp=_RdRSP(1);
       //for Nrc(respose to command, 8clk)
       rSIOCON|=(1<<3);
       while(!endSioTR);
       endSioTR=0;
    }
    //for Nac(response to data, 8clk)
    rSIOCON=(0x0<<7)|(0x0<<6)|(0x0<<5)|(0x0<<4)|(0x0<<3)|(0x0<<2)|0x1;
    rSIOCON|=(1<<3);
    while(!endSioTR);
    endSioTR=0;

    _RdDAT();

    Card_desel();

    View_Rx_buf();
}

void Wt_Slg_blk(void)
{
    int addr;
    char pattern, status=0;
    short int rsp=1;

    Uart_Printf("[SPI single block write test]\n");

    Uart_Printf("Input the start address of block to write\n");
    Uart_Printf("The address must be aligned 512bytes !!!\n");
    Uart_Printf("(Ex: 0x0, 0x200, 0x400, 0x600, 0x800, 0xa00....)\n");
    addr=Uart_GetIntNum();

    Uart_Printf("\nInput the char pattern to write..\n");
    pattern=(char)Uart_GetIntNum();


    Card_sel();
```

```
    while(rsp)
    {
        _WtCMD(24,addr);
       rsp=_RdRSP(1);
       //for Nrc(respose to command, 8clk)
       rSIOCON|=(1<<3);
       while(!endSioTR);
       endSioTR=0;
    }
    //for Nwr(response to data, 8clk)
    rSIOCON|=(1<<3);
    while(!endSioTR);
    endSioTR=0;

    Uart_Printf("Writing..\n");

    status= _WtDAT(pattern);
    if(status)
       Uart_Printf("Write O.K.\n");
    else
       Uart_Printf("Write Fail..\n");
    Card_desel();
}

void Set_Prt(void)
{}

void Clr_Prt(void)
{}

void Chk_Prt(void)
{}

void Er_Sector(void)
{}

void Er_Group(void)
{}

void Flush_Rx_buf(void)
{
    //-- Flushing Rx buffer
    int i;

    for(i=0;i<BLOCK_LEN;i++)
        Rx_buffer[i]=0;
    Uart_Printf("\n--End Rx buffer flush\n");
}

void View_Rx_buf(void)
{
    //-- Display Rx buffer
    int i, j;
```

```
    j=Rx_buffer[0];
    for(i=0;i<BLOCK_LEN;i++)
    {
       if(!(j==Rx_buffer[i]))
           Uart_Printf("Check data..\n");
    }
    Uart_Printf("RxBuf[  0]=0x%x ~ ",Rx_buffer[0]);
    Uart_Printf("RxBuf[511]=0x%x\n",Rx_buffer[511]);
}

void View_Tx_buf(void)
{
    //-- Display Tx buffer
    int i;

    for(i=0;i<BLOCK_LEN;i++)
        Uart_Printf("TB[%03x]=%x,",Tx_buffer[i]);
}

short int _RdRSP(char type)
{
    short int rsp, tmp;

    rSIOCON=(0x0<<7)|(0x0<<6)|(0x0<<5)|(0x0<<4)|(0x0<<3)|(0x0<<2)|0x1;
    //receive only
    if(type==1)     //R1 type
    {
       //for Ncr(command to respose, 8~16clk)
       rSIOCON|=(1<<3);
       while(!endSioTR);
       endSioTR=0;

       rSIOCON|=(1<<3);//after 8clk
       while(!endSioTR);
       rsp=(short int)rSIODAT;
       endSioTR=0;

       if(rsp&0x8)
       {
           rSIOCON|=(1<<3);//after 16clk
           while(!endSioTR);
           rsp=(short int)rSIODAT;
           endSioTR=0;
       }
    }
    else     //R2 type
    {
       //for Ncr(command to respose, 8~16clk)
       rSIOCON|=(1<<3);
       while(!endSioTR);
       endSioTR=0;
```

```
        rSIOCON|=(1<<3);//after 8clk
        while(!endSioTR);
        rsp=(short int)rSIODAT;
        tmp=rsp<<8;
        endSioTR=0;

        if(rsp&0x8)
        {
            rSIOCON|=(1<<3);//after 16clk
            while(!endSioTR);
            rsp=(short int)rSIODAT;
            tmp=rsp<<8;
            endSioTR=0;
        }

        rSIOCON|=(1<<3);
        while(!endSioTR);
        rsp=(short int)rSIODAT;
        rsp=tmp|rsp;
        endSioTR=0;
    }
    return rsp;
}

void _RdDAT(void)
{
    int i;
    char rsp=0;

    //--for Nac(until start byte)
    while(!(rsp==0xfe))
    {
        rSIOCON|=(1<<3);
        while(!endSioTR);
        rsp=(short int)rSIODAT;
        endSioTR=0;
    }
//    Uart_Printf("Start data=0x%x\n",rsp);

    //--Rx data
    for(i=0;i<BLOCK_LEN;i++)
    {
        rSIOCON|=(1<<3);
        while(!endSioTR);
        Rx_buffer[i]=rSIODAT;
        endSioTR=0;
    }
    //--for CRC16
    for(i=0;i<2;i++)
    {
        rSIOCON|=(1<<3);
        while(!endSioTR);
        endSioTR=0;
```

```
    }
    //--for Next command(Nrc)
    rSIOCON|=(1<<3);
    while(!endSioTR);
    endSioTR=0;
}

char _WtDAT(char pattern)
{
    int i;
    short int tmp=0, rsp=0, busy=0;

    rSIOCON |=(0x1<<5);
    //--for start
    rSIODAT=0xfe;
    rSIOCON|=(1<<3);
    while(!endSioTR);
    endSioTR=0;
    //--Tx data
    for(i=0;i<BLOCK_LEN;i++)
    {
       rSIODAT=pattern;
       rSIOCON|=(1<<3);
       while(!endSioTR);
       endSioTR=0;
    }
    //--for CRC16
    for(i=0;i<2;i++)
    {
       rSIODAT=0xff;
       rSIOCON|=(1<<3);
       while(!endSioTR);
       endSioTR=0;
    }

    //--for data response
    rSIOCON=(0x0<<7)|(0x0<<6)|(0x0<<5)|(0x0<<4)|(0x0<<3)|(0x0<<2)|0x1;
       //receive only

    while(!(rsp==0x1))
    {
       rSIOCON|=(1<<3);
       while(!endSioTR);
       rsp=(short int)rSIODAT;
       tmp=rsp;
       endSioTR=0;
       rsp &= 0x11;
    }
    //--busy check
    while(!(busy==0xff))
    {
       rSIOCON|=(1<<3);
       while(!endSioTR);
```

```
        busy=(short int)rSIODAT;
        //WrUTXH0('$');
    }

    tmp &= 0xe;
    if(tmp==4)
        return 1;
    else
    {
        Uart_Printf("Response=0x%x\n",tmp);
        return 0;
    }
}


void _WtCMD(char cmd, int addr)
{
    int i;

    rSIOCON |=(0x1<<5);

    switch(cmd)
    {
        case 0:
           for(i=0;i<6;i++)
           {
              rSIODAT=CMD0[i];
              rSIOCON|=(1<<3);
              while(!endSioTR);
              endSioTR=0;
           }
            break;
        case 1:
           for(i=0;i<6;i++)
           {
              rSIODAT=CMD1[i];
              rSIOCON|=(1<<3);
              while(!endSioTR);
              endSioTR=0;
           }
            break;
       case 9:
           for(i=0;i<6;i++)
           {
              rSIODAT=CMD9[i];
              rSIOCON|=(1<<3);
              while(!endSioTR);
              endSioTR=0;
           }
            break;
       case 10:
           for(i=0;i<6;i++)
           {
```

```
            rSIODAT=CMD10[i];
            rSIOCON|=(1<<3);
            while(!endSioTR);
            endSioTR=0;
        }
         break;
     case 13:
        for(i=0;i<6;i++)
        {
            rSIODAT=CMD13[i];
            rSIOCON|=(1<<3);
            while(!endSioTR);
            endSioTR=0;
        }
         break;
     case 16:
        for(i=0;i<4;i++)
            CMD16[i+1]=(char)(addr>>(24-8*i));
        for(i=0;i<6;i++)
        {
            rSIODAT=CMD16[i];
            rSIOCON|=(1<<3);
            while(!endSioTR);
            endSioTR=0;
        }
         break;
    case 17:
        for(i=0;i<4;i++)
            CMD17[i+1]=(char)(addr>>(24-8*i));
        for(i=0;i<6;i++)
        {
            rSIODAT=CMD17[i];
            rSIOCON|=(1<<3);
            while(!endSioTR);
            endSioTR=0;
        }
         break;
    case 24:
        for(i=0;i<4;i++)
            CMD24[i+1]=(char)(addr>>(24-8*i));
        for(i=0;i<6;i++)
        {
            rSIODAT=CMD24[i];
            rSIOCON|=(1<<3);
            while(!endSioTR);
            endSioTR=0;
        }
         break;
    case 28:
        for(i=0;i<4;i++)
            CMD28[i+1]=(char)(addr>>(24-8*i));
        for(i=0;i<6;i++)
        {
```

```
        rSIODAT=CMD28[i];
        rSIOCON|=(1<<3);
        while(!endSioTR);
        endSioTR=0;
    }
     break;
case 29:
    for(i=0;i<4;i++)
       CMD29[i+1]=(char)(addr>>(24-8*i));
    for(i=0;i<6;i++)
    {
       rSIODAT=CMD29[i];
       rSIOCON|=(1<<3);
       while(!endSioTR);
       endSioTR=0;
    }
     break;
case 30:
    for(i=0;i<4;i++)
       CMD30[i+1]=(char)(addr>>(24-8*i));
    for(i=0;i<6;i++)
    {
       rSIODAT=CMD30[i];
       rSIOCON|=(1<<3);
       while(!endSioTR);
       endSioTR=0;
    }
     break;
case 32:
    for(i=0;i<4;i++)
       CMD32[i+1]=(char)(addr>>(24-8*i));
    for(i=0;i<6;i++)
    {
       rSIODAT=CMD32[i];
       rSIOCON|=(1<<3);
       while(!endSioTR);
       endSioTR=0;
    }
     break;
case 33:
    for(i=0;i<4;i++)
       CMD33[i+1]=(char)(addr>>(24-8*i));
    for(i=0;i<6;i++)
    {
       rSIODAT=CMD33[i];
       rSIOCON|=(1<<3);
       while(!endSioTR);
       endSioTR=0;
    }
     break;
case 34:
    for(i=0;i<4;i++)
       CMD34[i+1]=(char)(addr>>(24-8*i));
```

```
    for(i=0;i<6;i++)
    {
       rSIODAT=CMD34[i];
       rSIOCON|=(1<<3);
       while(!endSioTR);
       endSioTR=0;
    }
     break;
case 35:
    for(i=0;i<4;i++)
       CMD35[i+1]=(char)(addr>>(24-8*i));
    for(i=0;i<6;i++)
    {
       rSIODAT=CMD35[i];
       rSIOCON|=(1<<3);
       while(!endSioTR);
       endSioTR=0;
    }
     break;
case 36:
    for(i=0;i<4;i++)
       CMD36[i+1]=(char)(addr>>(24-8*i));
    for(i=0;i<6;i++)
    {
       rSIODAT=CMD36[i];
       rSIOCON|=(1<<3);
       while(!endSioTR);
       endSioTR=0;
    }
     break;
case 37:
    for(i=0;i<4;i++)
       CMD37[i+1]=(char)(addr>>(24-8*i));
    for(i=0;i<6;i++)
    {
       rSIODAT=CMD37[i];
       rSIOCON|=(1<<3);
       while(!endSioTR);
       endSioTR=0;
    }
     break;
case 38:
    for(i=0;i<4;i++)
       CMD38[i+1]=(char)(addr>>(24-8*i));
    for(i=0;i<6;i++)
    {
       rSIODAT=CMD38[i];
       rSIOCON|=(1<<3);
       while(!endSioTR);
       endSioTR=0;
    }
     break;
case 59:
```

```
            for(i=0;i<4;i++)
               CMD59[i+1]=(char)(addr>>(24-8*i));
            for(i=0;i<6;i++)
            {
               rSIODAT=CMD59[i];
               rSIOCON|=(1<<3);
               while(!endSioTR);
               endSioTR=0;
            }
             break;
       default :
            break;
    }
}

void Card_sel(void)
{
    int i;
    //for 74clk
    for(i=0;i<10;i++)
    {
       rSIODAT=0xff;
       rSIOCON|=(1<<3);
       while(!endSioTR);
       endSioTR=0;
    }
    rPDATE=0x0;      //for CS
}

void spi_card_init(void)
{
    short int rsp=1;

    pISR_SIO=(unsigned)Sio_Int;
    rINTMSK=~(BIT_GLOBAL|BIT_SIO);

    TR_Buf_init();
    //-- SPI(SIO) port initialize
    rPCONF=(rPCONF&0x3ff)+(3<<19)+(3<<16)+(3<<13)+(3<<10);
    rPUPF |=0x160;//Rx must pullup

    //-- SPI(SIO) controller & card initialize
    rSIOCON=(0x0<<7)|(0x0<<6)|(0x0<<5)|(0x0<<4)|(0x0<<3)|(0x0<<2)|0x1;
    //inter clk, MSB 1st, Tx/Rx, falling, no action, auto run, SIO int
    rSBRDR=99;       //MCLK=60MHz,SIOCK=300KHz(in initialize time)
    rIVTCNT=0;

    //-- Makes card idle state
    Card_sel();
    while(rsp)
    {
       _WtCMD(0,0);
       rsp=_RdRSP(1);
```

```
        //for Nrc
        if(rsp==0x1)
            break;
        rSIOCON|=(1<<3);
        while(!endSioTR);
        endSioTR=0;
    }
    Card_desel();
    Uart_Printf("\nIn idle state\n");

    //-- Negotiate operating condition, it makes card ready state
    Card_sel();
    while(rsp)
    {
        _WtCMD(1,0);
        rsp=_RdRSP(1);
        //for Nrc
        rSIOCON|=(1<<3);
        while(!endSioTR);
        endSioTR=0;
    }
    Uart_Printf("\nIn ready state\n");

    rSIOCON=(0x0<<7)|(0x0<<6)|(0x1<<5)|(0x0<<4)|(0x0<<3)|(0x0<<2)|0x1;
    rSBRDR=5; //MCLK=60MHz,SIOCK=5MHz
    rIVTCNT=0;

    Card_desel();
}
```

PCMCIA SAMPLE CODE

PCMCIA : pd6710.h

```
#ifndef __PD6710_H__
#define __PD6710_H__

void TestPD6710(void);


#define PD6710_MEM_BASE_ADDRESS     (0x4000000) //nGCS2
#define PD6710_IO_BASE_ADDRESS      (0x5000000)

#define rPD6710_INDEX      (*(volatile unsigned char *)(PD6710_IO_BASE_ADDRESS+0x3e0))
#define rPD6710_DATA (*(volatile unsigned char *)(PD6710_IO_BASE_ADDRESS+0x3e1))


#define CHIP_REVISION                   (0x0)
#define INTERFACE_STATUS          (0x1)
#define POWER_CTRL                      (0x2)
#define INT_GENERAL_CTRL          (0x3)
#define CARD_STAT_CHANGE          (0x4)
#define MANAGEMENT_INT_CONFIG     (0x5)
#define MAPPING_ENABLE                  (0x6)
#define IO_WINDOW_CTRL                  (0x7)
#define SYS_IO_MAP0_START_L       (0x8)
#define SYS_IO_MAP0_START_H       (0x9)
#define SYS_IO_MAP0_END_L         (0xa)
#define SYS_IO_MAP0_END_H         (0xb)
#define SYS_IO_MAP1_START_L       (0xc)
#define SYS_IO_MAP1_START_H       (0xd)
#define SYS_IO_MAP1_END_L         (0xe)
#define SYS_IO_MAP1_END_H         (0xf)
#define SYS_MEM_MAP0_START_L      (0x10)
#define SYS_MEM_MAP0_START_H      (0x11)
#define SYS_MEM_MAP0_END_L        (0x12)
#define SYS_MEM_MAP0_END_H        (0x13)
#define CARD_MEM_MAP0_OFFSET_L (0x14)
#define CARD_MEM_MAP0_OFFSET_H (0x15)
#define MISC_CTRL1                      (0x16)
#define FIFO_CTRL                       (0x17)
#define SYS_MEM_MAP1_START_L      (0x18)
#define SYS_MEM_MAP1_START_H      (0x19)
#define SYS_MEM_MAP1_END_L        (0x1a)
#define SYS_MEM_MAP1_END_H        (0x1b)
#define CARD_MEM_MAP1_OFFSET_L (0x1c)
#define CARD_MEM_MAP1_OFFSET_H (0x1d)
#define MISC_CTRL2                      (0x1e)
#define CHIP_INFO                       (0x1f)
#define SYS_MEM_MAP2_START_L      (0x20)
#define SYS_MEM_MAP2_START_H      (0x21)
#define SYS_MEM_MAP2_END_L        (0x22)
#define SYS_MEM_MAP2_END_H        (0x23)
```

```
#define CARD_MEM_MAP2_OFFSET_L (0x24)
#define CARD_MEM_MAP2_OFFSET_H (0x25)
#define ATA_CTRL                        (0x26)
#define SCRATCHPAD                      (0x27)
#define SYS_MEM_MAP3_START_L      (0x28)
#define SYS_MEM_MAP3_START_H      (0x29)
#define SYS_MEM_MAP3_END_L        (0x2a)
#define SYS_MEM_MAP3_END_H        (0x2b)
#define CARD_MEM_MAP3_OFFSET_L (0x2c)
#define CARD_MEM_MAP3_OFFSET_H (0x2d)
#define EXTENDED_INDEX                  (0x2e)
#define EXTENDED_DATA                   (0x2f)
#define SYS_MEM_MAP4_START_L      (0x30)
#define SYS_MEM_MAP4_START_H      (0x31)
#define SYS_MEM_MAP4_END_L        (0x32)
#define SYS_MEM_MAP4_END_H        (0x33)
#define CARD_MEM_MAP4_OFFSET_L (0x34)
#define CARD_MEM_MAP4_OFFSET_H (0x35)
#define CARD_IO_MAP0_OFFSET_L     (0x36)
#define CARD_IO_MAP0_OFFSET_H     (0x37)
#define CARD_IO_MAP1_OFFSET_L     (0x38)
#define CARD_IO_MAP1_OFFSET_H     (0x39)
#define SETUP_TIMING0                   (0x3a)
#define CMD_TIMING0                     (0x3b)
#define RECOVERY_TIMING0          (0x3c)
#define SETUP_TIMING1                   (0x3d)
#define CMD_TIMING1                     (0x3e)
#define RECOVERY_TIMING1          (0x3f)


#endif /*__PD6710_H__*/
```

PCMCIA : pd6710.c

```
#include "..\inc\option.h"
#include "..\inc\def.h"
#include "..\inc\44b.h"
#include "..\inc\44blib.h"
#include "..\inc\pd6710.h"

/*
       DESCRIPTIONS OF THE BOARD

       nEINT1=nINT_P_CON=nINTR
       nEINT2=nINT_P_DEV=IRQ3

       A24=1, I/O area
       A24=0, mem area
       AEN=nGCS2,    So,the address area,A23=1, will not be used.

       absolute address
       0x4000000~0x4FFFFFF: memory area(memaddr=0x0~0x7FFFFF)
       0x5000000~0x5FFFFFF: I/O area(ioaddr=0x0~0x7FFFFF)
*/

int PD6710_Init(void);
void PD6710_InitBoard(void);
void PD6710_InitInterrupt(void);
void PD6710_CardEnable(void);
void PD6710_Wr(U8 index, U8 data);
U8 PD6710_Rd(U8 index);
void PD6710_Modify(U8 index,U8 mask,U8 data);
void PD6710_CommonMemAccess(void);
void PD6710_AttrMemAccess(void);

U8 Card_RdAttrMem(U32 memaddr);
U8 Card_RdIO(U32 ioaddr);
void Card_WrIO(U32 ioaddr,U8 data);

void PrintCIS(void);

void __irq IsrPD6710Card(void);
void __irq IsrPD6710Management(void);

volatile int isCardInsert=0;

void _dbg(int i)
{
       Uart_Printf("<%d>",i);
       //Uart_Printf("INT_GENERAL_CTRL=%x\n",PD6710_Rd(INT_GENERAL_CTRL));
}

void TestPD6710(void)
{
       Uart_Printf("[PD6710 test for reading pc_card CIS]\n");
```

```
        Uart_Printf("Insert PC card!!!\n");
        PD6710_InitBoard();
        PD6710_Init();
        PD6710_InitInterrupt();


        //Check if CD1,2 is L. If nCD1,2 is L, the card has been inserted.
        //In this case, the card management interrupt isn't occurred.
        if((PD6710_Rd(INTERFACE_STATUS)&0xc)==0xc)
        {
                Uart_Printf("Card is inserted.\n");
            Delay(2000);
                        //For card contact stablization. This delay is needed for stable operation.
                        //If this delay isn't here, CF card will not be identified.
                PD6710_CardEnable();
                PrintCIS();
        }

        while(1)
        {
                Uart_Printf(".");
                Delay(10000);
        }
}


//IRQ stack size is enlarged to 4096 byte.
void PrintCIS(void)
{
        int i,j;
        int cisEnd=0;
        static U8 str[16];
        U8 c;
        Uart_Printf("[Card Information Structure]\n");

        PD6710_AttrMemAccess();

        //search the end of CIS
        while(1)
        {
                if(Card_RdAttrMem((cisEnd)*2)==0xff)     //0xff= termination tuple
                        break;
                cisEnd++;
                cisEnd+=Card_RdAttrMem((cisEnd)*2)+1;
        }
        Uart_Printf("cisEnd=0~%x\n",cisEnd);

        for(i=0;i<=cisEnd*2;i+=2)
        {
                c=Card_RdAttrMem(i);
                str[(i%0x20)/2]=c;
                Uart_Printf("%2x,",c);
                if((i%0x20)>=0x1e)
```

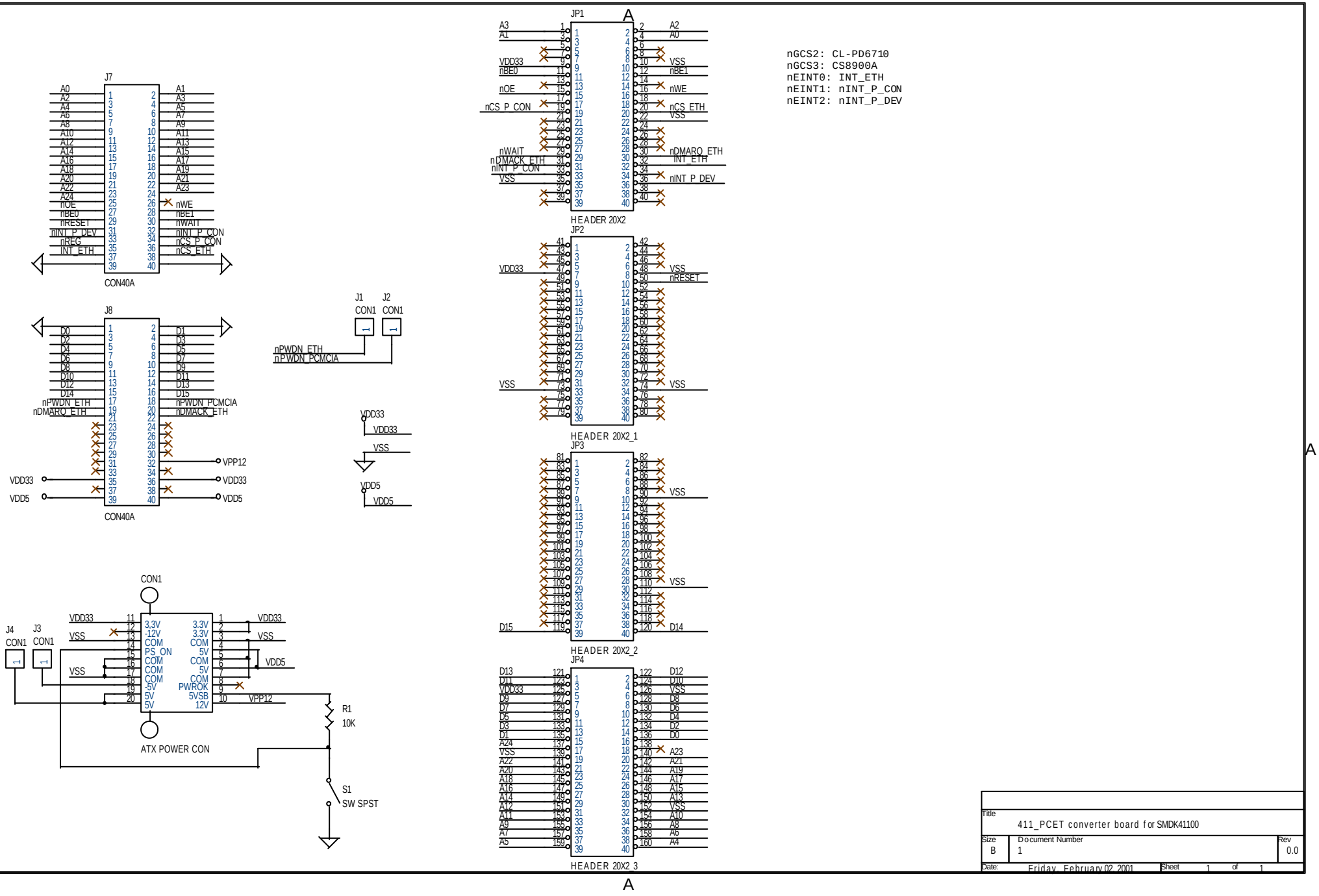
```
            {
                Uart_Printf("//");
                for(j=0;j<0x10;j++)
                    if(str[j]>=' ' && str[j]<=127)Uart_Printf("%c",str[j]);
                    else Uart_Printf(".");
                Uart_Printf("\n");
            }
        }
        Uart_Printf("\n");
}



void PD6710_InitBoard(void)
{
//initialize S3C44B0X for PD6710
        rNCACHBE0=( (((PD6710_MEM_BASE_ADDRESS+0x1ffffff)>>12)+1)<<16 )|
            (PD6710_MEM_BASE_ADDRESS>>12);

    rPCONG=(rPCONG&~(0x3c))|0x3c; //EINT1,2 is enabled.

    rPCONF=rPCONF&~(0x3<<4)|(0x2<<4); //PF2=nWAIT

    rBWSCON=rBWSCON&~(0xf<<8)|(0xd<<8);    //nGCS2=nUB/nLB(nSBHE),nWAIT,16-bit
}


void PD6710_InitInterrupt(void)
{
        //nEINT1=nINT_P_CON=nINTR
        //nEINT2=nINT_P_DEV=IRQ3
    rEXTINT=(rEXTINT&~(0xff<<4))|(0x2<<4)|(0x4<<8); //EINT1=falling,EINT2=rising.
    //rEXTINT=(rEXTINT&~(0xff<<4))|(0x0<<4)|(0x1<<8); //EINT1=L leavel,EINT2=H level.

    pISR_EINT1=(U32)IsrPD6710Management; //nINT_P_CON
        pISR_EINT2=(U32)IsrPD6710Card;    //nINT_P_DEV
        rI_ISPC=BIT_EINT1;
        rI_ISPC=BIT_EINT2;
        //rINTMSK=~(BIT_GLOBAL|BIT_EINT1|BIT_EINT2);
        rINTMSK=~(BIT_GLOBAL|BIT_EINT1);
            //The card interrupt is disabled temporarily
}


int PD6710_Init(void)
{
//Initialize PD-6710
        PD6710_Wr(POWER_CTRL,(0<<7)|(1<<5)|(0<<4)|(0<<0));
            //output2card_disable,AUTO_POWER,VCC_POWER_OFF,Vpp1=0V

        PD6710_Wr(INT_GENERAL_CTRL,(1<<7)|(0<<5)|(1<<4)|(3<<0));
            //nSTSCHG=status change(not RI input),mem_card_interface,
            //manage_int=-INTR pin, nINT_P_DEV=IRQ3
```

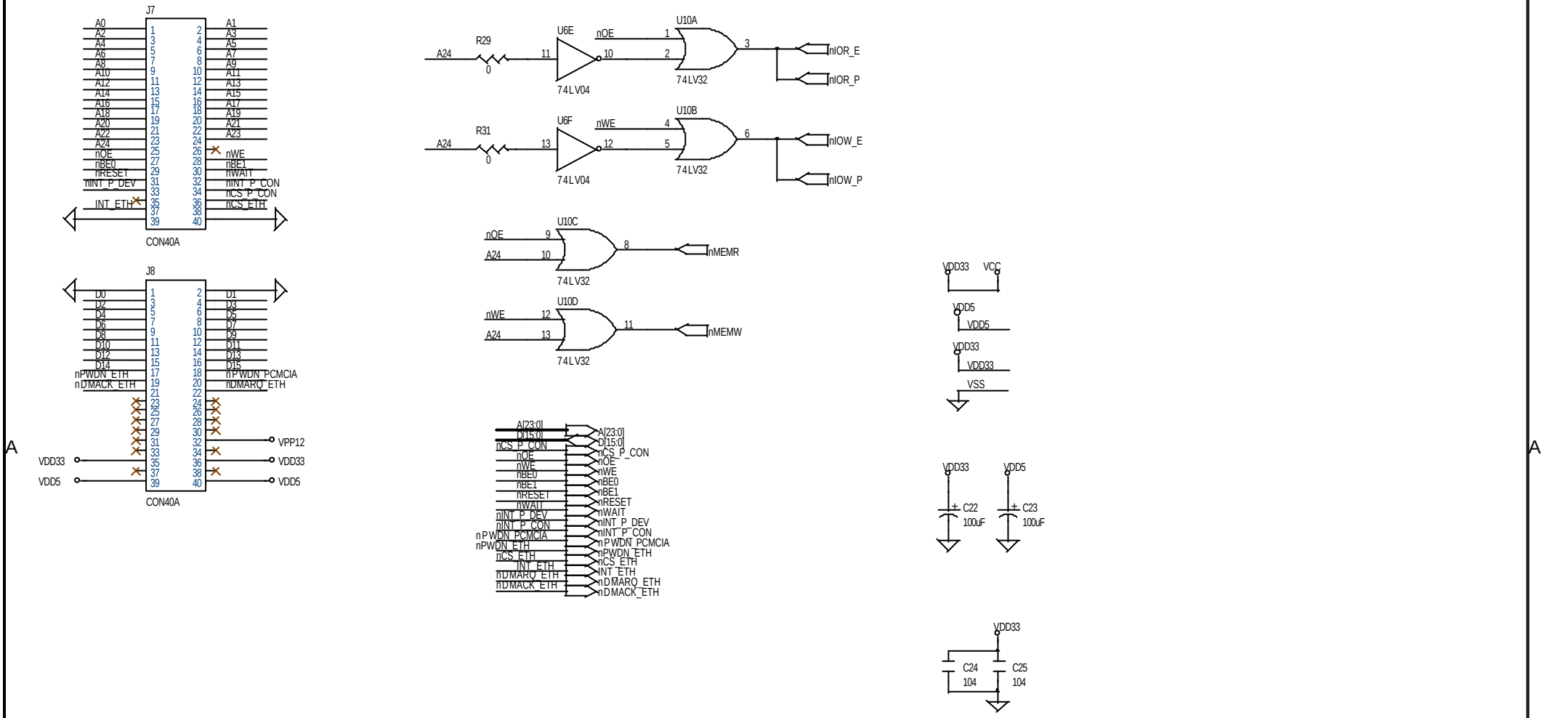
```
PD6710_Wr(MANAGEMENT_INT_CONFIG,(1<<0)|(1<<3)|(3<<4));
          //status_change_int_enable, card_detect_int_enable, IRQ disabled(IRQ_3)
          //Is nINTR pin used???
   PD6710_Wr(IO_WINDOW_CTRL,0|(0<<3));
          //IO0=8bit,timing_set_0

   // I/O windows must never include 3e0h and 3e1h
   // IO AREA=0x3f8~0x3ff(COM1) ->0x3f8~0x3ff
   PD6710_Wr(SYS_IO_MAP0_START_L,0xf8);
   PD6710_Wr(SYS_IO_MAP0_START_H,0x3);
   PD6710_Wr(SYS_IO_MAP0_END_L,0xff);
   PD6710_Wr(SYS_IO_MAP0_END_H,0x3);
   PD6710_Wr(CARD_IO_MAP0_OFFSET_L,0x0);
   PD6710_Wr(CARD_IO_MAP0_OFFSET_H,0x0);


   //PD6710_Wr(SYS_MEM_MAP0_START_L,0x10);
          //If this is memory window, the lowest 64KB should be reserved.
   PD6710_Wr(SYS_MEM_MAP0_START_L,0x0); //MEM0=8bit data width
          //To access CIS, I have known that memory window bus width should be
          //  8-bit by experiment.
          //Originally, CIS uses 8-bit bus. But, why not in 16-bit bus width.
   PD6710_Wr(SYS_MEM_MAP0_START_H,0x0);
   PD6710_Wr(SYS_MEM_MAP0_END_L,0x0f);              //0x0 ~ 0xffff
   PD6710_Wr(SYS_MEM_MAP0_END_H,0x0|(0<<6)); //timing_set_0
   PD6710_Wr(CARD_MEM_MAP0_OFFSET_L,0x0);
   PD6710_Wr(CARD_MEM_MAP0_OFFSET_H,0x0|(1<<6)); //nREG=active
          //MEM AREA=0x0~0xFFFF ->0x0~0xFFFFFF

   PD6710_Wr(MAPPING_ENABLE,1|(1<<6));
          //memory map 0 enabled, I/O map 0 enabled

   PD6710_Wr(MISC_CTRL1,(0<<7)|(1<<4)|(1<<3)|(1<<2)|(1<<1));

   //INPACK_ignored,speak_enable,edge_irq_intr,edge_management_intr,nVCC_3_enabled(temp)
   PD6710_Wr(MISC_CTRL2,1|(1<<1)|(1<<4));
          //25Mhz_bypass,low_power_dynamic,IRQ12=drive_LED

   PD6710_Wr(FIFO_CTRL,0x80);  //Flush FIFO
   //before configuring timing register, FIFO should be cleared.

   //default access time is 300ns
   PD6710_Wr(SETUP_TIMING0,0x2);     //80ns(no spec)
   PD6710_Wr(CMD_TIMING0,0x8);       //320ns(by spec,25Mhz clock)
   PD6710_Wr(RECOVERY_TIMING0,0x2);  //80ns(no spec)

   PD6710_Wr(CHIP_INFO,0x0);
   if((PD6710_Rd(CHIP_INFO)&0xc0)!=0xc0 || (PD6710_Rd(CHIP_INFO)&0xc0)!=0x00 )
   {
          Uart_Printf("PD6710 hardware identification error!!!\n");
          return 0;
   }
```

```
    return 1;
}


void PD6710_CardEnable(void) //after insertion interrupt
{
       if(PD6710_Rd(MISC_CTRL1)&0x1)  //MISC_CTRL1[0] 0=3.3V  1=5.0V
       {
              PD6710_Modify(MISC_CTRL1,0x2,0x0); //nVCC_5_enabled
              Uart_Printf("5.0V card is detected.\n");
       }
       else
       {
              PD6710_Modify(MISC_CTRL1,0x2,0x2); //nVCC_3_enabled
              Uart_Printf("3.3V card is detected.\n");
       }
       PD6710_Modify(POWER_CTRL,(1<<4)|3,(1<<4)|1);
              //VCC_POWER_on,VPP=Vcc(3.3V or 5.0V)
       Delay(100);
              //Delay 10ms should be here for power stabilization.
              //If this time is not here, the program may halt
              // in case that the pc card has been inserted before power_on
        //But, Why?

       Delay(10); //RESET should be Hi-Z for minimum 1ms
       PD6710_Modify(INT_GENERAL_CTRL,(1<<6),0); //RESET=active(H)
       PD6710_Modify(POWER_CTRL,(1<<7),(1<<7)); //output2card_enable(RESET=Hi-Z -> output)
       PD6710_Modify(INT_GENERAL_CTRL,(1<<6),0); //RESET=active(H)
       Delay(1); //wait for minimum 10us
       PD6710_Modify(INT_GENERAL_CTRL,(1<<6),(1<<6)); //RESET=inactive(L)
       Delay(200);   //wait for 20ms
       //READY pin isn't available in mem_io_card mode.
       //So, don't check READY pin on I/O card.
       //while(!(PD6710_Rd(INTERFACE_STATUS)&0x20)); //INTERFACE_STATUS[5]=READY_PIN_STATUS
       PD6710_Modify(INT_GENERAL_CTRL,(1<<5),(1<<5));  //mem_card -> mem_io_card

       Delay(5000);
              //If this delay isn't here, some CF card will not be identified.
              //In oder to remove this delay, I think, we have to check READY signal.
}



void PD6710_Wr(U8 index, U8 data)
{
       //nREG=L
       rPD6710_INDEX=index;
       rPD6710_DATA=data;
}


U8 PD6710_Rd(U8 index)
{
```

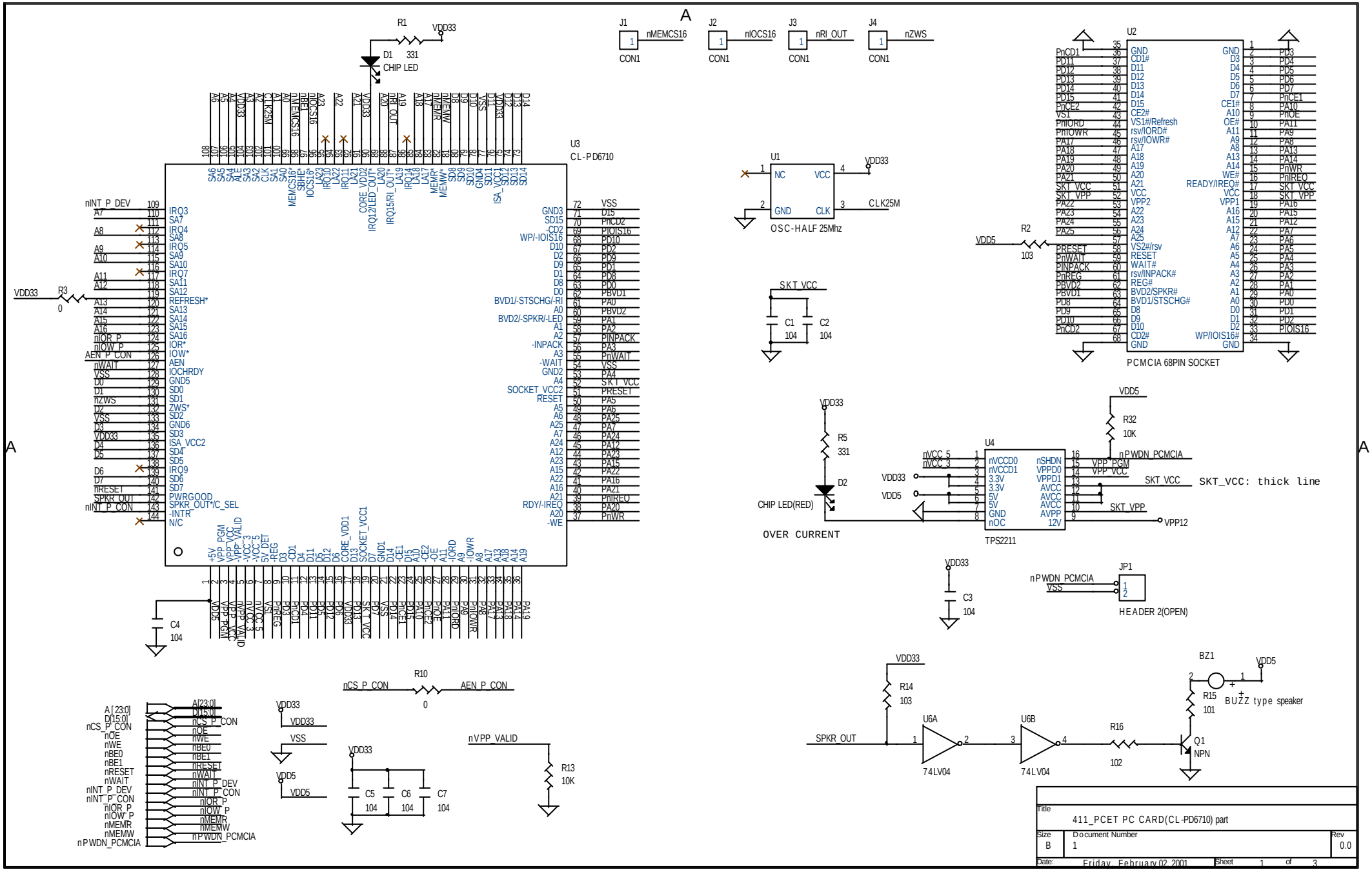
```
	//nREG=L
	rPD6710_INDEX=index;
	return rPD6710_DATA;
}


void PD6710_Modify(U8 index,U8 mask,U8 data)
{
	//nREG=L
	rPD6710_INDEX=index;
	rPD6710_DATA=(rPD6710_DATA)&~mask|data;
}


U8 Card_RdAttrMem(U32 memaddr)
{
	//nREG=L
	return *((volatile U8 *)(PD6710_MEM_BASE_ADDRESS+memaddr));
}


U8 Card_RdIO(U32 ioaddr)
{
	return *((volatile U8 *)(PD6710_IO_BASE_ADDRESS+ioaddr));
}


void Card_WrIO(U32 ioaddr,U8 data)
{
	*((volatile U8 *)(PD6710_IO_BASE_ADDRESS+ioaddr))=data;
}


/*
1)	I/O access: nREG=L(automatic)
	attribute memory access: nREG=L
	common memory access: nREG=H
2)	Before accessing Common memory area,
	PD6710_BeginCommonMemAccess() should be called to make nREG 'H'.
*/
void PD6710_CommonMemAccess(void)
{
	PD6710_Modify(CARD_MEM_MAP0_OFFSET_H,(1<<6),(0<<6)); //nREG=inactive, H
}


void PD6710_AttrMemAccess(void)
{
	PD6710_Modify(CARD_MEM_MAP0_OFFSET_H,(1<<6),(1<<6)); //nREG=active, L
}


void __irq IsrPD6710Management(void)    //nINT_P_CON
```

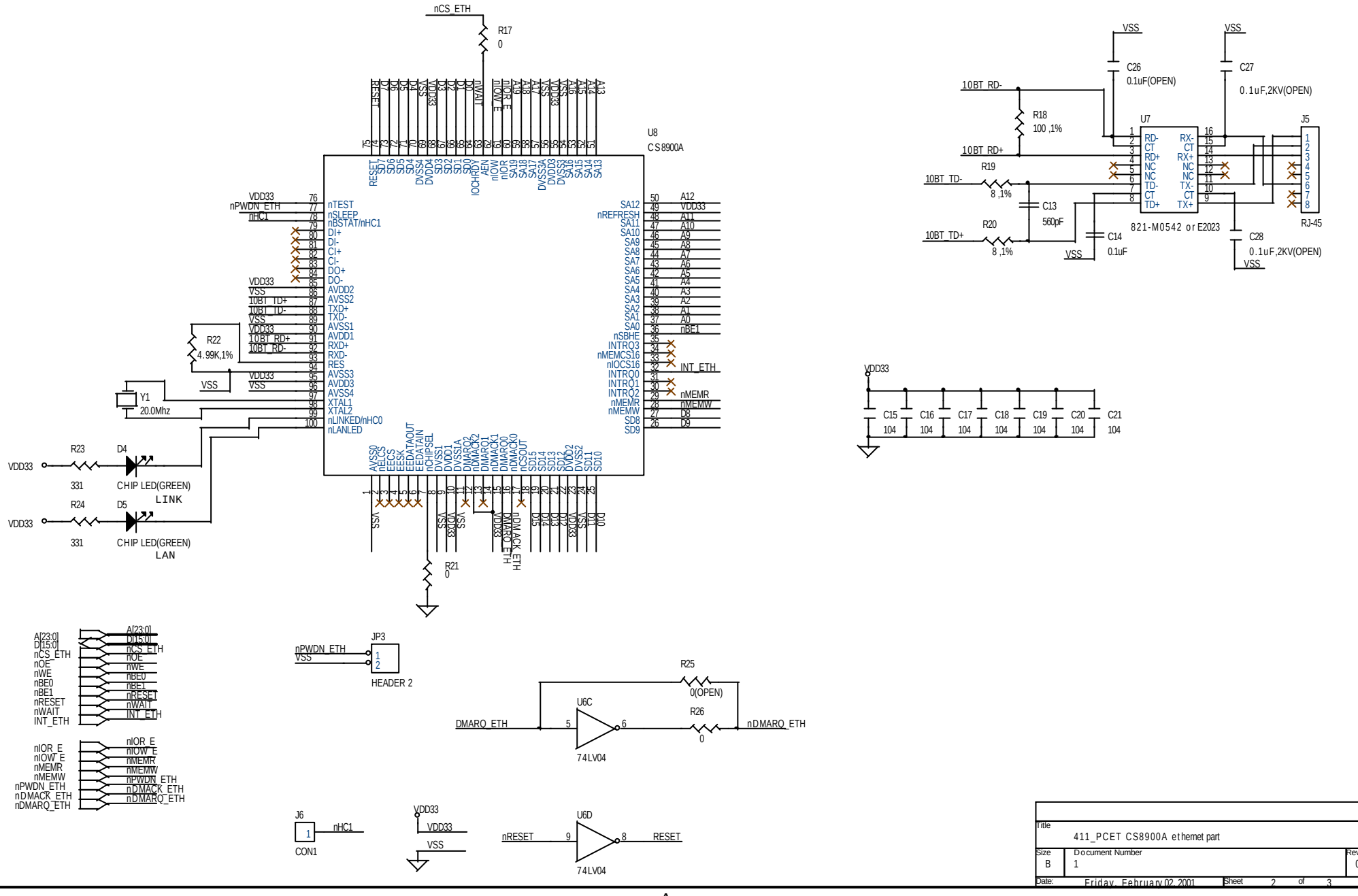
```
{
       U8 cardStat;
    //rI_ISPC=BIT_EINT1;
    Uart_Printf("\nPD6710 interrupt is occurred.\n");

    Delay(2000);
       //For card contact stablization. This delay is needed for operation stability
              //If this delay isn't here, some CF card may not be identified.

       cardStat=PD6710_Rd(CARD_STAT_CHANGE);
    //Card detect changed?
       if(cardStat&0x8)
       {
              //check if CD1,2 is L.
              if((PD6710_Rd(INTERFACE_STATUS)&0xc)==0xc)
              {
                     Uart_Printf("Card is inserted.\n");
                     PD6710_CardEnable();
                     PrintCIS();
              }
              else
              {
                     Uart_Printf("Card is ejected.\n");
                     PD6710_Init(); //can be removed.
              }
       }

       rI_ISPC=BIT_EINT1;
              //For level interrupt, the int pending should be cleared at the end of ISR.

}


void __irq IsrPD6710Card(void)    //nINT_P_DEV
{
    rI_ISPC=BIT_EINT2;
    Uart_Printf("PC card interrupt is occurred.\n");
}
```


J7
CON40A
J8
CON40A
J1 CON1
J2 CON1
nPWDN_ETH
nPWDN_PCMCIA
VDD33
VSS
VDD5
CON1
J4 CON1
J3 CON1
ATX POWER CON
R1
10K
S1
SW SPST
JP1
HEADER 20X2
JP2
HEADER 20X2_1
JP3
HEADER 20X2_2
JP4
HEADER 20X2_3
nGCS2: CL-PD6710
nGCS3: CS8900A
nEINT0: INT_ETH
nEINT1: nINT_P_CON
nEINT2: nINT_P_DEV
Title
411_PCET converter board for SMDK41100
Size B
Document Number 1
Rev 0.0
Date: Friday, February 02, 2001
Sheet 1 of 1



Title	411_PCET interface part	
Size: B	Document Number: 1	Rev: 0.0
Date: Friday, February 02, 2001	Sheet 3 of 3	


R1
VDD33
D1
331
CHIP LED
J1
nMEMCS16
J2
nIOCS16
J3
nRI_OUT
J4
nZWS
CON1
U3
CL-PD6710
U1
OSC-HALF 25Mhz
CLK25M
U2
PCMCIA 68PIN SOCKET
R2
103
VDD5
R3
0
C1
C2
104
SKT_VCC
C4
C3
R5
331
D2
CHIP LED(RED)
OVER CURRENT
U4
TPS2211
R32
10K
nPWDN_PCMCIA
VPP_PGM
VPP_VCC
SKT_VCC
SKT_VCC: thick line
SKT_VPP
VPP12
JP1
HEADER 2(OPEN)
R10
0
nCS_P_CON
AEN_P_CON
R13
10K
nVPP_VALID
R14
103
U6A
U6B
74LV04
SPKR_OUT
R16
102
Q1
NPN
BZ1
BUZZ type speaker
R15
101
C5
C6
C7
A[23:0]
D[15:0]
nOE
nWE
nBE0
nBE1
nRESET
nWAIT
nINT_P_DEV
nINT_P_CON
nIOR_P
nIOW_P
nMEMR
nMEMW
nPWDN_PCMCIA
Title
411_PCET PC CARD(CL-PD6710) part
Size
B
Document Number
1
Rev
0.0
Date:
Friday, February 02, 2001
Sheet
1
of
3



Title	411_PCET CS8900A ethernet part	
Size: B	Document Number: 1	Rev: 0.0
Date: Friday, February 02, 2001	Sheet 2 of 3	

NOTES