

SMIL (SmartMedia™ Interface Library)

Hardware Edition

Version 1.00

TOSHIBA Corporation

01, July, 2000

The information contained herein is subject to change without notice.

The information contained herein is presented only as a guide for the application of our products.

No responsibility is assumed by TOSHIBA for any infringement of patents or other rights of third parties, which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of TOSHIBA or others.

Copyright (C) 2000 TOSHIBA Corporation. All rights reserved.

Contents

1. Introduction	3
2. Hardware required Specification.....	4
2.1. Block Diagram.....	5
2.2. SmartMedia™ Interface	6
2.3. Voltage Control	9
3. Register Specification	11
3.1. Register Map	11
3.2. Detailed Specification.....	12
3.2.1. Data Register (read/write)	12
3.2.2. Mode Register (write).....	13
3.2.3. Status Register (read).....	15
3.2.4. Interrupt Status Register (read/write) [Optional]	16
3.2.5. Interrupt Mask Register (read/write) [Optional]	17
4. ECC Circuit Control	18
4.1. ECC circuit control Algorithm.....	18
4.2. Controller Identify Data Read.....	20
5. Reference Circuit	21
5.1. Block Diagram.....	21
5.2. Voltage control circuit.....	22
5.3. SmartMedia™ Control Circuit.....	22
5.4. VHDL source code.....	25

Revision history

Ver 1.00 2000-07-01 Formal release

<Related Documents>

- “SmartMedia™ Standard 2000”
- “RN5RG Series application manual”

SSFDC Forum
Ricoh Corporation

- SmartMedia™ is a trademark of Toshiba Corporation.

1.Introduction

SMIL (SmartMedia™ Interface Library) is the common design resource for all hardware and software compatible with SmartMedia™. The interface of SmartMedia™ is rather simple with less than 100 gates for hardware and approximately 30KB size driver. However, the implementation method was different for each device or make. For this reason, each manufacturer used individual hardware and software with the original design.

Under such circumstances, the main purpose of SMIL is to develop a common driver with simplified implementation by providing the common interface of SmartMedia™.

By this way, any built-in devices can make them compatible with others relatively easily and which will lead to a greater market size with seamless compatibility among devices using SmartMedia™.

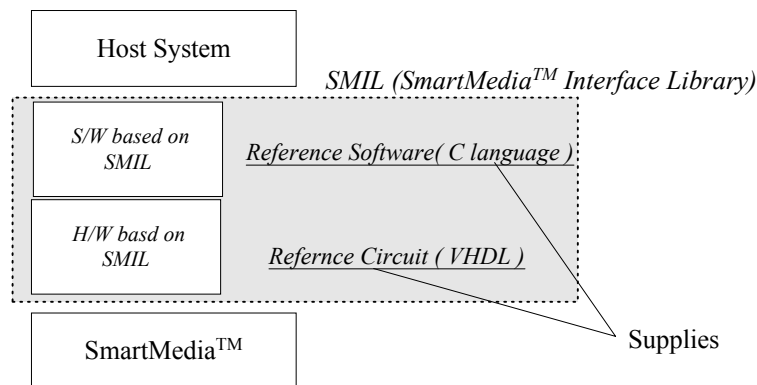


Fig. 1-1 SMIL concept

As a reference, logical description of SmartMedia™ Control Circuit is included in this document. It was made as a reference to develop a system that conform to SmartMedia™ control circuit based on SMIL and it can be used to understand the fundamental circuit structural image and to grasp the rough circuit size.

For an actual application, sufficient amount of simulation is recommended. Accordingly, see the important notices within this document.

2. Hardware required Specification

The SMIL enables you to control the SmartMedia™ with minimum of 2 byte I/O register. Additionally, it also has multiple functions to enhance the selectable functions.

This chapter explains about these specifications of SMIL hardware. As described in table 2-1, the device with the function required for SMIL hardware will be considered as the minimal setup.

Table 2-1 SMIL Hardware required specification, basic function

Function	Minimum construction
Data Register	O
Mode Register	O
Status Register	O
Interrupt Status Register	×
Interrupt Mask Register	×
Controller Identify Data Read	×
Hardware ECC Logic	×
8bit Data Transfer Mode	O
16bit Data Transfer Mode	×
32bit Data Transfer Mode	×
Data Access Lamp	Δ
Media Lock / Media Unlock	×
Media Eject	×
Status Change (Media Eject)	O
Status Change (Media Insert)	×
Interrupt	×

- O : indispensable function
- ×
- Δ : optional function

2.1. Block Diagram

SmartMedia™ Control Circuit conformed to SMIL is described below.

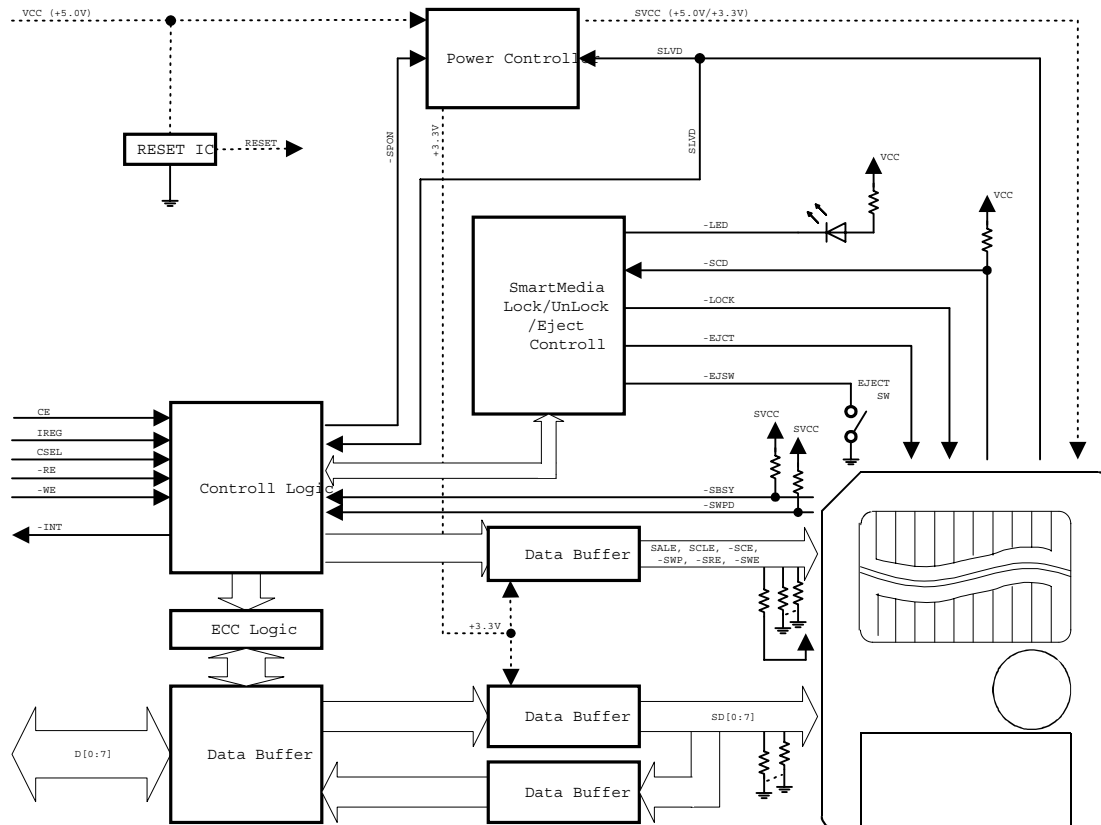


Fig. 2-1 block diagram

2.2. SmartMedia™ Interface

Followings are about the signal input/output level and the control method of SmartMedia™. Among them, control method to minimize the malfunction of SmartMedia™ during insertion or ejection and at the time of ON/OFF of the system power supply are included.

< SALE, SCLE, -SCE, -SWP, -SRE, -SWE, SD[0:7] >

In order to cope with both +5.0V and +3.3V model SmartMedia™, you must not have a circuit that will provide +5.0V to the +3.3V model by an error.

As a solution for such case, you can use the +3.3V voltage buffer for the output of SmartMedia™ Control Circuit. And for the dual port buffer, you can use the +3.3V voltage buffer with +5.0V voltage resistance.

In another case, there is a way to use SmartMedia™ voltage supply as voltage supply of the output buffer. However, in this case the electricity will be kept turned on regardless to the presence of SmartMedia™ and the output signal is required to set at GND level or Hi-Z even without SmartMedia™. Accordingly voltage supply can not be used without SmartMedia™, special caution is required in this case.

Also, the signal of Hi-Z should be pulled up with the power supply of SmartMedia™ SVCC (SVCC is at GND level when electricity is not turned on) and we recommend to secure stable signal level of input buffer.

< -SCD, -SBSY, -SWPD >

The signal of -SCD and -SBSY will be L: GND / H: Hi-Z. Accordingly, pull-up resistor will be required.

-SBSY will be pulled-up with SVCC, and -SCD will be pulled-up by voltage VCC which is turned on all the time.

Also, for -SCD, measures against chattering is required (such as inserting condenser between input terminal and GND).

For the cut-in detection of card insertion or ejection, we recommend to have a circuit against chattering on hardware side. (multiple checking from hardware)

The input signal of -SWPD, write-protect detection signal, will be pulled-up with VCC or SVCC. (The another end of write-protect detection should be connected to GND to form the complete circuit.)

< SLVD >

Pull-down is required for SLVD. If you regulate the power voltage for SLVD signal as a part of voltage circuit, pull-down will not be necessary because it will be executed within the control circuit.

< LED, LOCK, EJECT, -EJSW >

SMIL supports the SmartMedia™ connector with eject function.

Eject will be executed with the eject switch or with eject command from the system.

Followings are the specifications for hardware:

1. While the system making access to SmartMedia™, media-lock is in function.
2. While the media-lock is functioning, eject switch will not be functional. The event that eject switch was pushed will only be sent to the system.
3. When the system receives the event that eject switch was pushed, it will complete processing and release the media-lock, then execute the eject command.
4. If the media-lock is not in function, SmartMedia™ will be ejected by pushing the eject switch.
5. If there is no SmartMedia™ in the device, the command from eject switch will be neglected.

-EJSW is the input signal of eject switch. (Normally, VCC pull-up is required.)

-LED is the output signal of access lamp for SmartMedia™.

-LOCK is the output signal to indicate media-lock is in function.

-EJCT is the output signal for eject.

If LED to be used, the required specification should be open drain signal. To operate eject function, external Tr, such as solenoid, should be set.

Interface signals of SmartMedia™ are described in table 2-2:

The recommended DC characteristics are described in table 2-3. AC characteristics need to satisfy the specification required in “SmartMedia™ Electrical Specifications”.

For an actual application to the device, we recommend you to decide structural factors such as connectors, based on “SmartMedia™ Electrical Specifications”.

Table 2-2 SmartMedia™ Interface signals

Signal name	Sym.	IN/OUT	Remarks
SmartMedia™ Card Detect (*)	[-SCD]	IN	10k VCC (3.3V) pull up (schmitt trigger)
SmartMedia™ Busy (*)	[-SBSY]	IN	10k SVCC pull up (schmitt trigger)
SmartMedia™ Read Enable (*)	[-SRE]	OUT	100k SVCC pull up
SmartMedia™ Write Enable (*)	[-SWE]	OUT	
SmartMedia™ Card Enable (*)	[-SCE]	OUT	
SmartMedia™ Command Latch Enable	[SCLE]	OUT	100k pull down
SmartMedia™ Address Latch Enable	[SALE]	OUT	
SmartMedia™ Write Protected	[-SWP]	OUT	
SmartMedia™ Data 0	[SD0]	IN/OUT	100k pull down
SmartMedia™ Data 1	[SD1]	IN/OUT	
SmartMedia™ Data 2	[SD2]	IN/OUT	
SmartMedia™ Data 3	[SD3]	IN/OUT	
SmartMedia™ Data 4	[SD4]	IN/OUT	
SmartMedia™ Data 5	[SD5]	IN/OUT	
SmartMedia™ Data 6	[SD6]	IN/OUT	
SmartMedia™ Data 7	[SD7]	IN/OUT	
SmartMedia™ Low Voltage Detect	[SLVD]	IN	(schmitt trigger)
SmartMedia™ Write Protect Seal Detect	[-SWPD]	IN	10k SVCC pull up (schmitt trigger)
SmartMedia™ LED	[-LED]	OUT	O.D.
SmartMedia™ Locked	[-LOCK]	OUT	O.D.
SmartMedia™ EJECT Out	[-EJCT]	OUT	O.D.
SmartMedia™ EJECT In	[-EJSW]	IN	10k SVCC pull up (schmitt trigger)

Table 2-3 Recommended DC Characteristics

Characteristics	Sym.	Min	Typ	Max	Unit	Condition
H Input level *1	V _{IH}	2.0	-	5.5	V	
L Input level *1	V _{IL}	0	-	0.8	V	
H Output level *2	V _{OH}	2.8	-	SVCC	V	I _{OH} = -8mA
L Output level *2	V _{OL}	0	-	0.4	V	I _{OL} = 8mA
H Output level *3	V _{OH}	-	-	VCC	V	(O.D.)
L Output level *3	V _{OL}	0	-	0.4	V	I _{OL} = 24mA

*1) applicable input terminals are SD[0:7], -SCD, -SBSY, -SWPD, -EJSW

*2) applicable output terminals are SALE, SCLE, -SCE, -SWP, -SRE, -SWE, SD[0:7]

*3) applicable output terminals are (OD) is LED, LOCK, EJCT

2.3. Voltage Control

There are two models in SmartMedia™, one is +5.0V and the other is +3.3V model. And for each model, an appropriate voltage has to be provided. The appropriate power voltage can be detected from the input signal (SLVD). <Table 2-4>

If there is no power is supplied to SmartMedia™, SLVD can not detect the difference. Accordingly, special caution is required. Following is the description of voltage control circuit type. The table describes voltage specification required for SmartMedia™.

Table 2-4 FLVD signals

SmartMedia™ voltage	SLVD signal
+3.3V	SVCC
+5.0V	NC

1. SLVD Power Voltage Control

The insertion of SmartMedia™ turns on the power of SmartMedia™ with SLVD signal, the power voltage control will be executed as a part of voltage circuit and an appropriate voltage will be provided to each model of SmartMedia™. The circuit structure is as specified in the figure 2-2. RA & RB are feedback resistor and R1 & R2 are protection resistor for Tr. If transistor is to be mounted externally, 2SA1213 or equivalent type need to be used.

• Operation Explanation

< +3.3v SmartMedia™ : SLVD signal is connected to SVCC >

Output voltage level of SVCC will be detected at feedback resistor RA, RB. With error range amplifier, the difference from the reference voltage will be compared. Then, by adjusting the base current of PNP transistor, the output voltage SVCC will be adjusted. Both reference voltage and feedback resistor will set the output voltage SVCC to be +3.3V.

< +5.0v SmartMedia™ : FLVD signal is not connected. >

Because the + input of error range amplifier is 0V, the PNP transistor will be on all the time. For this reason, (VOC – Vce) will be output to SVCC.

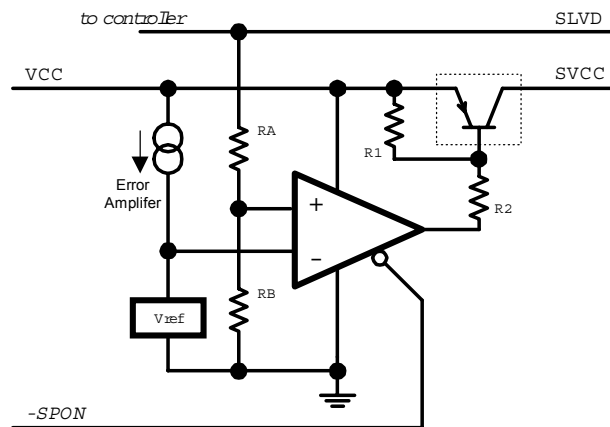


Fig. 2-2 Voltage control circuit

2. Two-step switching

After inserting the SmartMedia™, +3.3V will be supplied to SmartMedia™.

By checking the SLVD signal, when it detects the +5.0V model, the power source will be changed to +5.0V.

For power switching, “Vcc 3.3V to 5V” command will be used.

• Operation Explanation

< Power supply sequence >

Detect the insertion of SmartMedia™

+3.3V will be supplied to SmartMedia™

Detect the model of SmartMedia™

For the +5.0V model, +5.0V will be supplied.

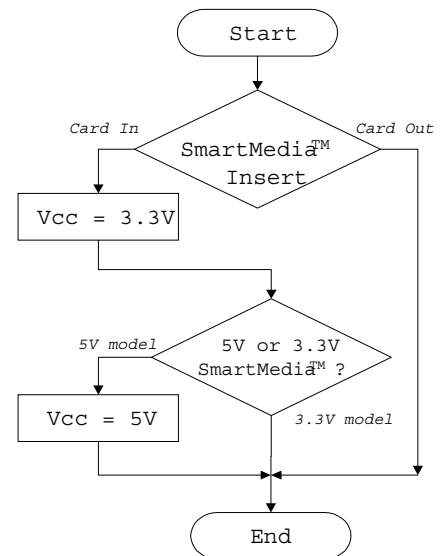


Fig.2-3 Power supply sequence

3. Power supply/Cut off only

If the device is for +3.3V model only, and intended to provide +3.3V power only, the voltage control should execute the function of power supply / cut off only.

When SmartMedia connectors, which support live-line connection / disconnection, are used for the above device, voltage control circuit is not necessary. However, even in such case, it is desirable to execute I/O pin output control of SmartMedia™, so we recommend to execute the voltage control from the software side.

When the power of devices is completely off during the insertion or ejection of SmartMedia™ (such device that unable to change SmartMedia™ without opening the battery cover), the voltage control is unnecessary.

Table 2-5 SmartMedia™ voltage required specification

Characteristics	Sym.	MIN	TYP	MAX	Unit	Condition
Output voltage	SVCC	3.15	3.30	3.45	V	+3.3V SmartMedia™ Only
		4.75	5.00	5.25	V	+5.0V SmartMedia™ Only
Output current	ISVCC	-	-	100	mA	

Note) These are strictly required specifications (about ±5%) for each SmartMedia™ specification.

For actual application to devices, we recommend to use connectors and other parts by conforming to “SmartMedia™ Electrical Specifications”.

3.Register Specification

3.1.Register Map

Table 3-1 shows the register of the control circuit based on SMIL.

SmartMedia™ can be controlled with three registers, “Data Register”, “Mode Register” and “Status Register”, and 2byte address spaces.

If it is necessary to have function such as insert/eject detection of SmartMedia™, interrupting control registers such as “Interrupt Status Register” or “Interrupt Mask Register” need to be added.

Each register can be placed at any address, however, for the enhancement of software compatibility, we recommend to place them as specified in table 3-2 ~ 3-4.

Table 3-1 SMIL Register

Function	Length (byte)	Access	note
Data Register	1	Read/Write	Optional
	2 / 4		
Mode Register	1	Write Only	
Status Register	1	Read Only	
Interrupt Status Register	1	Read/Write	Optional
Interrupt Mask Register	1	Read/Write	Optional

Table 3-2 Register Map (8bit Access)

Offset	Read	Write
+0	Data Register	
+1	(Reserved)	
+2	Status Register	Mode Register
+3	(Reserved)	
+4	Interrupt Status Register	
+5	(Reserved)	
+6	Interrupt Mask Register	
+7	(Reserved)	

Table 3-3 Register Map (16bit Access)

Offset	Read	Write
+0	Data Register (D7-0)	
+1	Data Register (D15-8)	
+2	Status Register	Mode Register
+3	(Reserved)	
+4	Interrupt Status Register	
+5	(Reserved)	
+6	Interrupt Mask Register	
+7	(Reserved)	

Note) “Data Register” is accessible for both 8 and 16byte.

Table 3-4 Register Map (32bit Access)

Offset	Read	Write
+0	Data Register (D7-0)	
+1	Data Register (D15-8)	
+2	Data Register (D23-16)	
+3	Data Register (D31-24)	
+4	(Mode Register)	Mode Register
+5	Status Register	-
+6	Interrupt Status Register	
+7	Interrupt Mask Register	

Note) “Data Register” is accessible for 8, 16 and 32byte.

“Mode Register” can be read-out. In this case the written value will be read-out.

3.2. Detailed Specification

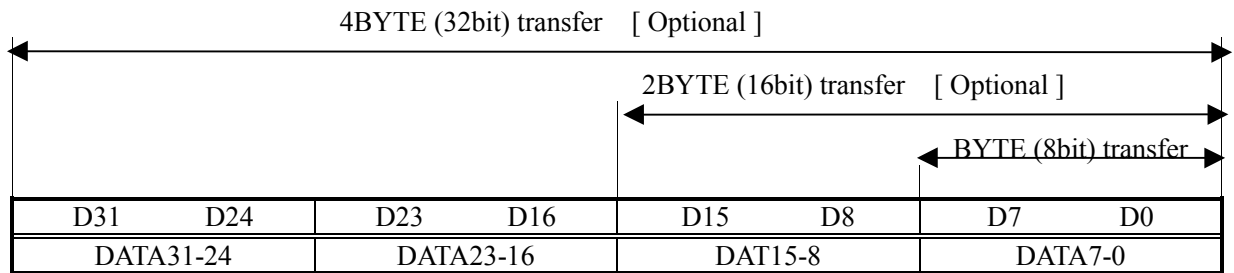
3.2.1.Data Register (read/write)

This is the register to execute data transfer between SmartMedia™ and host system.

The fundamental unit of data transfer is a byte access (8bit), however, 2byte(16bit) and 4byte(32bit) access are also possible depending upon the system.

(Default: xxxx xxxx / xxxx xxxx xxxx xxxx / xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx)

[“x” indicates unstable.]



- Write : Host System ---> SmartMedia™
- Read : SmartMedia™ ---> Host System

3.2.2.Mode Register (write)

This register will set the data transfer mode of the Data Register. It is prohibited to use other commands not specified in the table below.

D7	D6	D5	D4	D3	D2	D1	D0
<i>-WP</i>	ECC1	ECC0	<i>CE</i>	PCNT1	PCNT0/ <i>LED</i>	<i>ALE</i>	<i>CLE</i>

Register bits, written in italic type, will control the control signal of SmartMedia™ directly.

(Default: 0000 0000)

[-SCE] = - ([POWER] and CE)

[-SWP] = ([POWER] and -WP)

[SALE] = ([POWER] and ALE)

[SCLE] = ([POWER] and CLE)

[-LED] = - ([POWER] and LED)

([POWER] is the voltage supply of SmartMedia™.)

Two registers, PCNT0 & PCNT1 are used for voltage control. It is prohibited to set PCNT1 at '1' other than for voltage control.

Table 3-5 Mode setting command

Command	Code	Function
Read Data	0001 0100 (14h)	SmartMedia™ Data Read Mode
Write Command	0001 0101 (15h)	SmartMedia™ Command Write Mode (SmartMedia™ Data Read)
Write Address	0001 0110 (16h)	SmartMedia™ Address Write Mode (SmartMedia™ Data Read)
Write Data	1001 0100 (94h)	SmartMedia™ Data Write Mode
Write Command	1001 0101 (95h)	SmartMedia™ Command Write Mode (SmartMedia™ Data Write/Erase)
Write Address	1001 0110 (96h)	SmartMedia™ Address Write Mode (SmartMedia™ Data Write/Erase)
Standby	0000 0000 (00h)	SmartMedia™ Standby Mode
Vcc Power Off	0000 1000 (08h)	Vcc Power Off for SmartMedia™
Vcc Power On	0000 1100 (0Ch)	Vcc Power On for SmartMedia™
Vcc 3.3V to 5V	0000 1100 (0Ch)	Vcc Power +3.3V to +5.0V for SmartMedia™ (*3) [Optional]
LED Off	xxxx x0xx (00h)	LED turn Off
LED On	xxxx x1xx (04h)	LED turn On
SmartMedia™ Eject On	0110 1000 (68h)	SmartMedia™ Eject On [Optional]
SmartMedia™ Eject Off	0000 1000 (08h)	SmartMedia™ Eject Off [Optional]
SmartMedia™ Lock	0110 1100 (6Ch)	SmartMedia™ Locked [Optional]
SmartMedia™ UnLock	0000 1100 (0Ch)	SmartMedia™ UnLocked [Optional]

Command	Code	Function (ECC Controll [Optional])
Reset ECC Logic	x11x xxxx (-)	SmartMedia™ ECC Data Reset (*1)(*2)
R/W with ECC	x011 xxxx (-)	SmartMedia™ Data Read with ECC data count up
Read ECC Data	x101 xxxx (-)	SmartMedia™ ECC Data Read (*1)
Read Controller ID	x100 xxxx (40h)	Controller Identify Data Read (*1)

Note) *1) In this mode, -RE/-WE signal of SmartMedia™ will not be output during the Data Register access.

*2) To reset the ECC Logic, dummy read of Data Register is required.

*3) When the bit 6 (Model) of Status Register is '1'(+5.0V SmartMedia™) and bit 4 (PwrOn) is '1'(PowerOn), the voltage of SmartMedia™ will be set at +5.0V.

(It is required when the voltage control circuit is "Dual voltage Switch". Please refer "2.3 Power Supply Control" for the control method.)

Table 4-4 The contents of data transferring at Mode setting

• Normal

CE	IREG	CSEL	-WE	-RE	-WP	ECC1	ECC0	CE	PCNT1	PCNT0 /LED	ALE	CLE	Mode Reg.	Data	Data Bus
H	L	L		H	0	0	0	1	0	1	0	1	15h	Output	Write Command Data
					0	0	0	1	0	1	1	0	16h	Output	Write Address Data
					0	0	0	1	0	1	0	0	14h	Output	Write Data
H	L	L	H		0	0	0	1	0	1	0	0	14h	Input	Read Data
					0	0	1	1	0	1	0	0	34h	Input	Read Data with ECC data count up
					0	1	0	1	0	1	0	0	54h	Input	Read ECC data (*)
					0	1	0	0	0	0	0	0	40h	Input	Controller Identify Data Read (*)
					0	1	1	1	0	1	0	0	74h	Input	Reset ECC data (*) [Dummy Read]

• In case of SmartMedia™ Block Erase / SmartMedia™ Page Data Write

CE	IREG	CSEL	-WE	-RE	-WP	ECC1	ECC0	CE	PCNT1	PCNT0 /LED	ALE	CLE	Mode Reg.	Data	Data Bus
H	L	L		H	1	0	0	1	0	1	0	1	95h	Output	Write Command Data
					1	0	0	1	0	1	1	0	96h	Output	Write Address Data
					1	0	0	1	0	1	0	0	94h	Output	Write Data
					1	0	1	1	0	1	0	0	B4h	Output	Write Data with ECC data count up
H	L	L	H		1	0	0	1	0	1	0	0	94h	Input	Read Data
					1	1	0	1	0	1	0	0	D4h	Input	Read ECC data (*)

Note) In (*) mode, -RE/-WE signal of SmartMedia™ will not be output while the Data Register accessing.
 Other than Block Erase / Page Data Write time, '-WP' signal will be enabled.
 This is for the purpose to prevent erroneous write to SmartMedia™.

3.2.3. Status Register (read)

Followings are the present status information of controller. (Default: xx00 0x0x)

D7	D6	D5	D4	D3	D2	D1	D0
Busy	Model	'0'	PwrON	STCHG	CENB	EJREQ	WPD

Busy (bit 7) : *Busy / -Ready*

Shows the present status of SmartMedia™

- 1: Busy
- 0: Ready

Model (bit 6) : *SmartMedia™ Model*

Shows the model of SmartMedia™

- 1: 5V Vcc SmartMedia™
- 0: 3.3V Vcc SmartMedia™

PwrON (bit 4) : *SmartMedia™ Power ON*

If voltage is supplied to SmartMedia™, this bit will be set.

Only when SmartMedia™ is inserted, power can be turned on.

If SmartMedia™ is ejected during power supplying, the power will be cut off.
(While reset, the power is cut off.)

- 1: Power ON
- 0: Power OFF

STCHG (bit 3) : *Card Status Change*

Shows changing to different status.

Actually this means the cases of card inserting and card-ejecting.

This bit will be reset by writing on Mode Register.

There is no influence by voltage status.

This bit can't be reset by writing Interrupt Status Register.

- 1: SmartMedia™ Card Status Change
- 0: none

CENB (bit 2) : *Card Enable*

Shows inserting status of SmartMedia™

- 1: SmartMedia™ ready
- 0: NO SmartMedia™

EJREQ (bit 1) : *Card EJECT Request*

Shows the interrupting by eject command (Ex. switch on/off)

This bit will be reset by writing "1" on PCNT1 of Mode Register.

This bit will not be reset by writing on Interrupt Status Register.

- 1: EJECT Request
- 0: None

WPD (bit 0) : *Write Protect Seal Detected*

Shows writing protected or not (Protection seals detected).

- 1: Write Protected
- 0: Write Enabled

3.2.4. Interrupt Status Register (read/write) [Optional]

Followings describe present the interrupting information of controller. (Default: 0000 0000)

D7	D6	D5	D4	D3	D2	D1	D0
'0'	'0'	'0'	'0'	CDIN	CDOUT	EJREQ	RDYREQ

CDIN (bit 3) : Card Insert Request

If SmartMedia™ card is inserted in connector, this bit is set.

When writing, "1", this bit is changed to "0".

When writing "0", this bits is no changed.

When bit3 of Status register is changed, this bit is not set "0".

1: Card Insert Request

0: None

CDOUT (bit 2) : Card Out Request

When SmartMedia™ card is ejected from connector, this bit is set.

When writing, "1", this bit is changed to "0".

When writing "0", this bits is no changed.

When bit3 of Status register is changed, this bit is not set "0".

1: Card Out Request

0: None

EJREQ (bit 1) : Card EJECT Request

Shows there is interrupting of EJECT request by switch or something. (In case of LOCK)

When writing, "1", this bit is changed to "0".

When writing "0", this bits is no changed.

When bit1 of Status register is changed, this bit is not set "0".

1: Card EJECT Request

0: None

RDYREQ (bit 0) : SmartMedia™ Ready Request

When R/-B signal of SmartMedia™ is changed "Low" to "High", This bit is set.

When writing, "1", this bit is changed to "0".

When writing "0", this bits is no changed.

1: (Busy --> Ready) Request

0: none

3.2.5. Interrupt Mask Register (read/write) [Optional]

Followings are the present interrupting prohibition information of controller. (Default: 0000 0000)

D7	D6	D5	D4	D3	D2	D1	D0
INTEN	'0'	'0'	'0'	MCDIN	MCDOUT	MEJREQ	MRDYREQ

INTEN (bits 7) : *Interrupt Enable*

The setting of permission / prohibition of -INT signal.

- 1: Interrupt Enable
- 0: Interrupt Disable

[-INT] = -([INTEN and ((CDIN and MCDIN) or (CDOUT and MCDOUT)
Or (EJREQ and MEJREQ) or (RDYREQ and MRDYREQ))])

MCDIN (bit 3) : *Card Insert Request Mask*

Generation of interrupt depends on Interrupt Status Register

"CDIN" bit. "1": interrupt "0": prohibit to interrupt

- 1: Enable Card Insert Request
- 0: None

MCDOUT (bit 2): *Card Out Request Mask*

Generation of interrupt depends on Interrupt Status Register

"CDOUT" bit. "1": interrupt "0": prohibit to interrupt

- 1: Enable Card out Request
- 0: None

MEJREQ (bit 1) : *Card EJECT Request Mask*

Generation of interrupt depends on Interrupt Status Register

"EJREQ" bit. "1": interrupt "0": prohibit to interrupt

- 1: Enable Card EJECT Request
- 0: None

MRDYREQ (bit 0): *SmartMedia™ Ready Request Mask*

Generation of interrupt depends on Interrupt Status Register

"RDYREQ" bit. "1": interrupt "0": prohibit to interrupt

- 1: Enable (Busy --> Ready)
- 0: none

4.ECC Circuit Control

4.1. ECC circuit control Algorithm

When ECC of SmartMedia™ physical format is realized by the control circuit based on SMIL, operating procedure is shown as follows. For the detailed specification of ECC, please see “ SmartMedia™ Physical Format Specifications “.

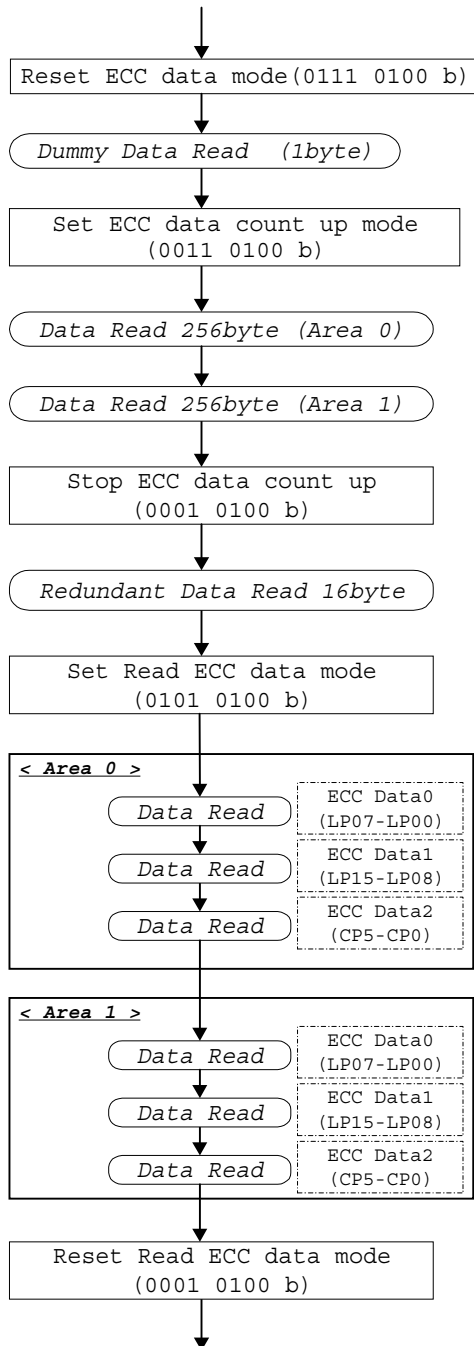


Fig.4-1 Data Read (512byte / page)

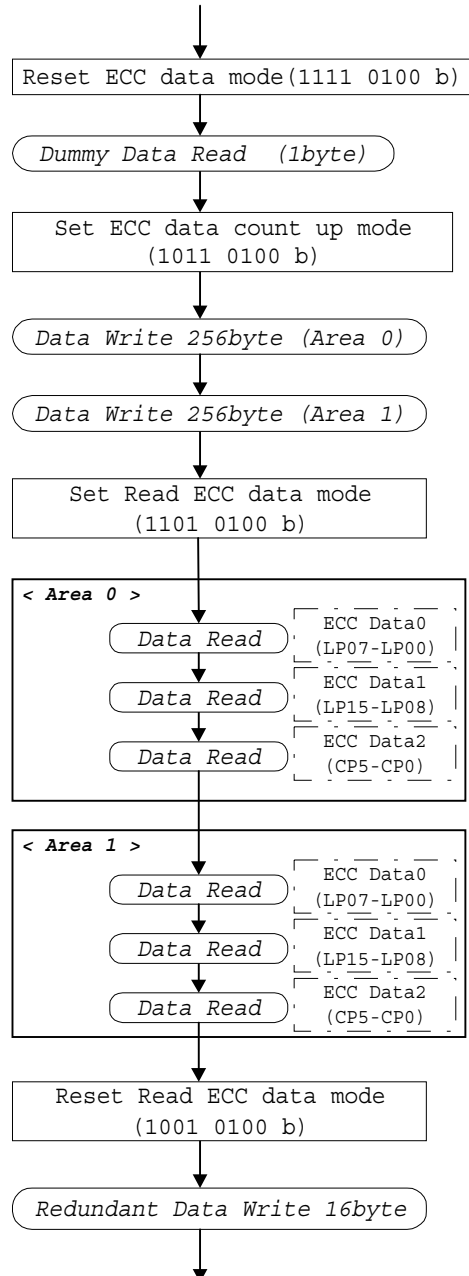


Fig.4-2 Data Write (512byte / page)

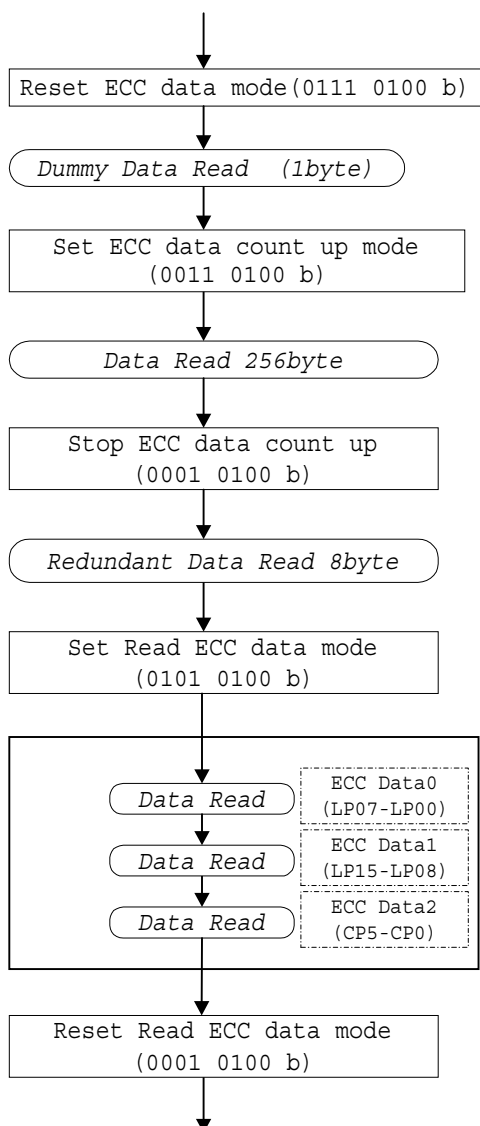


Fig.4-3 Data Read (256byte / page)

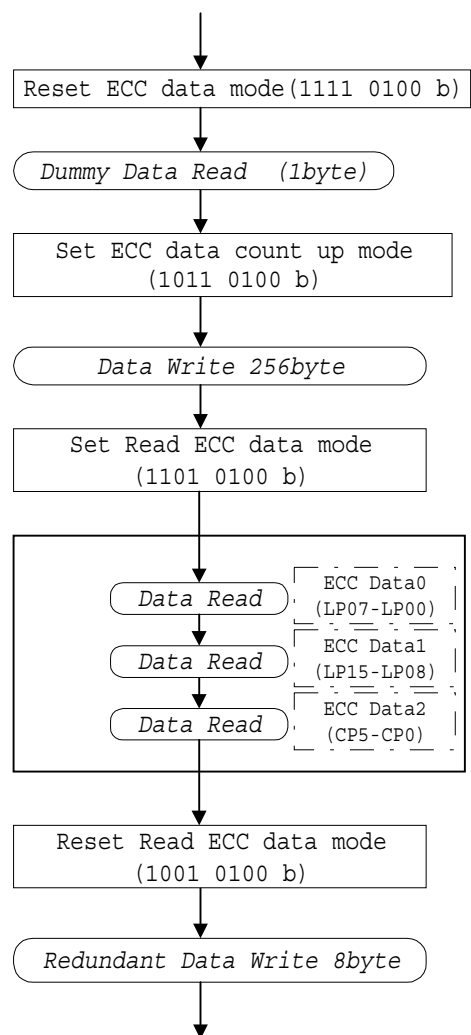


Fig.4-4 Data Write (256byte / page)

Note) Data register for ECC Data Read shall be accessed by the byte.

4.2. Controller Identify Data Read

In order to confirm function correspond to the discrimination of controller, 'Controller Identify Data' can be read. This procedure is shown as below. Data Read is done by byte access.

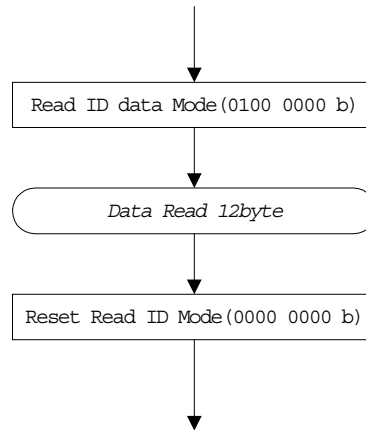


Fig. 4-5 Identify Data Read

Table 4-1 Identify Data

BYTE	DESCRIPTIONS	VALUE
1	Signature and Version "SMIL 1.00"	53h
2		4Dh
3		49h
4		4Ch
5		20h
6		31h
7		2Eh
8		30h
9		30h
10	Reserved	00h
11	HW option	X0h
		Bit 7 1: Support HW ECC, 0: Not Support
		Bit 6 1: Support Interrupt Reg., 0: Not Support
		Bit 5-0 Reserved
12	Reserved	00h

5.Reference Circuit

Followings are the practical examples of SmartMedia™ control circuit conformed to SMIL. These references support only byte (8bit) transfer.

5.1.Block Diagram

Fig.5-1 shows the block diagram of the whole control circuit.

The circuit consists of the voltage control circuit and SmartMedia™ control circuit.

The shaded area indicates SmartMedia™ control circuit. The circuit must meet the required specifications provided in section 2.2. “SmartMedia™ Interface”.

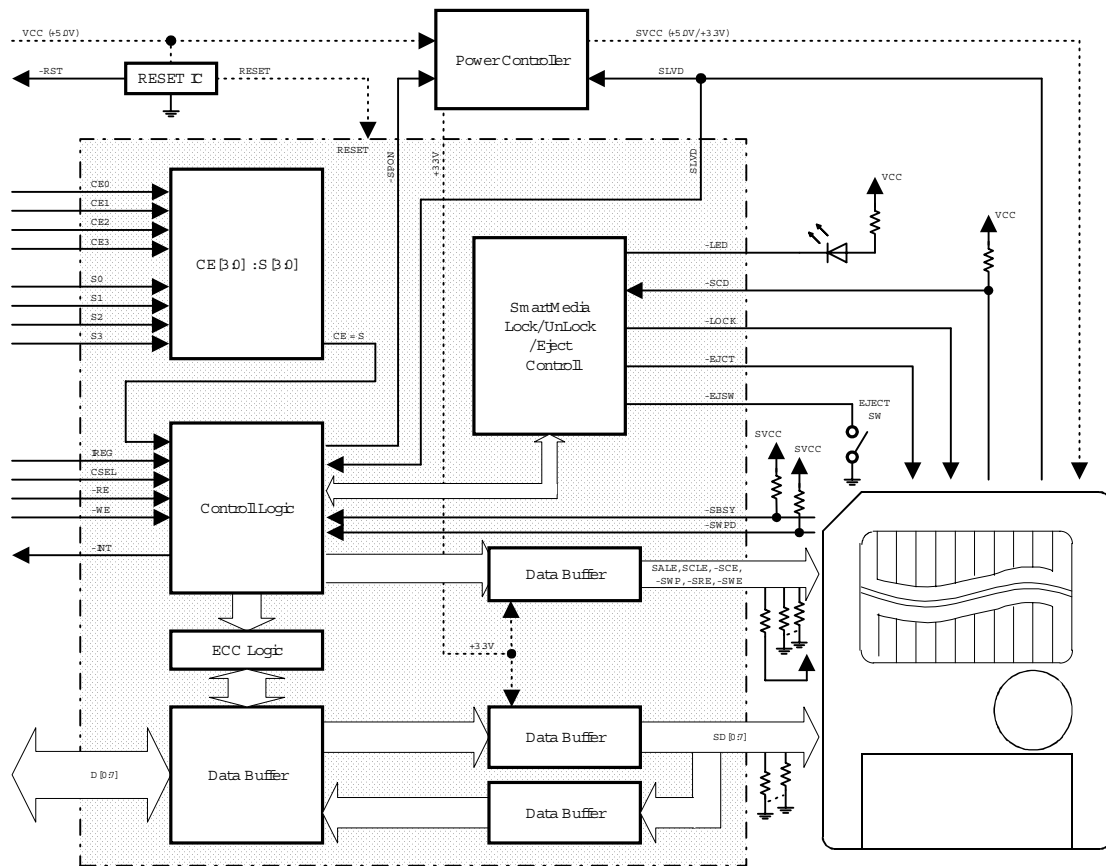


Fig. 5-1 Reference circuit block diagram

5.2. Voltage control circuit

As you know, there are two models of SmartMedia™. One is +5.0V and the other is +3.3V model. Naturally, each model has to be provided with the appropriate voltage.

There are two different methods to control voltage in Voltage control circuit.

- | | |
|-----------|---|
| Method 1) | Firstly, provide +3.3V to the SmartMedia™ and then identify the model of SmartMedia™. If the inserted SmartMedia™ is +5.0V model, +5.0V power will be supplied. |
| Method 2) | The method to control the power source of SmartMedia™ only. |

The following referential circuit (Figure 5-2) describes the Method 2)

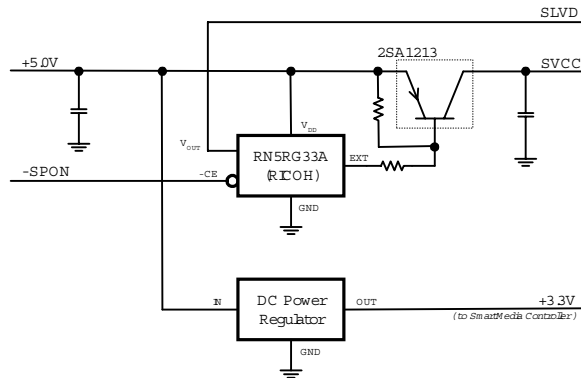


Fig.5-2 voltage control circuit

5.3. SmartMedia™ Control Circuit

The control circuit should use the tolerable +5.0V cells and a voltage of +3.3V.

As a substitute to the referential circuit, VHDL codes are attached.

Sample codes consist of unsynchronized circuit without clock input. Accordingly, it is necessary to pay special caution to the circuit delay in parity code generator of the ECC circuit. And hold time of CE Signal is used as a reset condition of Interrupt Status Register, so this is also taken care.

When you apply to system LSI, we recommend the change to synchronized circuit.

• Input Output terminal

Table 5-1 Input Output terminals

Signal name	Sym.	IN/OUT	Remarks
Controller Enable Select 0	[S0]	IN	
Controller Enable Select 1	[S1]	IN	
Controller Enable Select 2	[S2]	IN	
Controller Enable Select 3	[S3]	IN	
Controller Enable 0	[CE0]	IN	
Controller Enable 1	[CE1]	IN	
Controller Enable 2	[CE2]	IN	
Controller Enable 3	[CE3]	IN	
Controller Read Enable (*)	[-RE]	IN	
Controller Write Enable (*)	[-WE]	IN	
Controller Command Select	[CSEL]	IN	
Controller Interrupt Reg Select	[IREG]	IN	
Controller Interrupt Output	[-INT]	OUT	O.D.
Controller Data0	[D0]	IN/OUT	
Controller Data1	[D1]	IN/OUT	
Controller Data2	[D2]	IN/OUT	
Controller Data3	[D3]	IN/OUT	
Controller Data4	[D4]	IN/OUT	
Controller Data5	[D5]	IN/OUT	
Controller Data6	[D6]	IN/OUT	
Controller Data7	[D7]	IN/OUT	
SmartMedia™ Card Detect (*)	[-SCD]	IN	10k VCC(3.3V) pull up
SmartMedia™ Busy (*)	[-SBSY]	IN	10k SVCC pull up
SmartMedia™ Read Enable (*)	[-SRE]	OUT	100k SVCC pull up
SmartMedia™ Write Enable (*)	[-SWE]	OUT	
SmartMedia™ Card Enable (*)	[-SCE]	OUT	
SmartMedia™ Command Latch Enable	[SCLE]	OUT	100k GND pull down
SmartMedia™ Address Latch Enable	[SALE]	OUT	
SmartMedia™ Write Protected	[-SWP]	OUT	
SmartMedia™ Data0	[SD0]	IN/OUT	100k GND pull down
SmartMedia™ Data1	[SD1]	IN/OUT	
SmartMedia™ Data2	[SD2]	IN/OUT	
SmartMedia™ Data3	[SD3]	IN/OUT	
SmartMedia™ Data4	[SD4]	IN/OUT	
SmartMedia™ Data5	[SD5]	IN/OUT	
SmartMedia™ Data6	[SD6]	IN/OUT	
SmartMedia™ Data7	[SD7]	IN/OUT	
SmartMedia™ Power ON	[SPON]	OUT	
SmartMedia™ Low Voltage Detect	[SLVD]	IN	
SmartMedia™ Write Protect Seal Detect	[-SWPD]	IN	10k SVCC pull up
SmartMedia™ LED	[-LED]	OUT	O.D.
SmartMedia™ EJECT Out	[-EJCT]	OUT	O.D.
SmartMedia™ EJECT In	[-EJSW]	IN	10k SVCC pull up
SmartMedia™ Locked	[-LOCK]	OUT	O.D.
VCC	[VCC]	-	
GND	[GND]	-	

• CHIP SELECT

Chip selection will be executed by the input condition of S0 – S3 and CE0 – CE3.

S0 – S3 are connected to either GND or VCC, and can be set with 4 chip select signal of CE0 – CE3. When the conditions specified in table 5-2 are satisfied, chips are enabled. The select condition is as specified in the equation below.

$$CE = (S0==CE0) \text{ and } (S1==CE1) \text{ and } (S2==CE2) \text{ and } (S3==CE3)$$

Table 5-2 CHIP SELECT condition

S0	S1	S2	S3	CE0	CE1	CE2	CE3
GND	GND	GND	GND	Low	Low	Low	Low
VCC	GND	GND	GND	High	Low	Low	Low
GND	VCC	GND	GND	Low	High	Low	Low
GND	GND	VCC	GND	Low	Low	High	Low
GND	GND	GND	VCC	Low	Low	Low	High
VCC	VCC	GND	GND	High	High	Low	Low
VCC	GND	VCC	GND	High	Low	High	Low
VCC	GND	GND	VCC	High	Low	Low	High
GND	VCC	VCC	GND	Low	High	High	Low
GND	VCC	GND	VCC	Low	High	Low	High
GND	GND	VCC	VCC	Low	Low	High	High
VCC	VCC	VCC	GND	High	High	High	Low
VCC	VCC	GND	VCC	High	High	Low	High
VCC	GND	VCC	VCC	High	Low	High	High
GND	VCC	VCC	VCC	Low	High	High	High
VCC	VCC	VCC	VCC	High	High	High	High

• Register Map

Table 5-3 indicates how to access each register. Data transfer supports the byte access only.

Table 5-3 SMIL CHIP Register

CE	IREG	CSEL	-WR	-RD	Data	Function
L	*	*	*	*	High-Z	none
H	*	*	H	H	High-Z	none
H	L	L	L	H	Input	Data Register
H	L	L	H	L	Output	
H	L	H	L	H	Input	Mode Register
H	L	H	H	L	Output	Status Register
H	H	L	L	H	Input	Interrupt Status Register
H	H	L	H	L	Output	
H	H	H	L	H	Input	Interrupt Mask Register
H	H	H	H	L	Output	

5.4. VHDL source code

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity SMIL_CHIP is
port (
-----
RST          :in      std_logic;
-----
S            :in      std_logic_vector(3 downto 0);
CE           :in      std_logic_vector(3 downto 0);
-----
IREG         :in      std_logic;
CSEL         :in      std_logic;
WE           :in      std_logic;
RE           :in      std_logic;
DATA         :inout   std_logic_vector(7 downto 0);
-----
INT          :out      std_logic;
-----
SBSY         :in      std_logic;
SWPD         :in      std_logic;
SCD          :in      std_logic;
SCE          :out      std_logic;
SWP          :out      std_logic;
SCLE         :out      std_logic;
SALE         :out      std_logic;
SWE          :out      std_logic;
SRE          :out      std_logic;
SDATA        :inout   std_logic_vector(7 downto 0);
-----
SLVD         :in      std_logic;
SLED         :out      std_logic;
SPON         :out      std_logic;
-----
EJSW         :in      std_logic;
EJCT         :out      std_logic;
LOCK         :out      std_logic );
-----
end SMIL_CHIP;
```

architecture RTL of SMIL_CHIP is

```

signal Reset          :std_logic;          --- Controller Reset
signal ChipEnable     :std_logic;          --- Controller Chip Enable
signal CntWrite       :std_logic;          --- Controller Data Write
signal CntRead        :std_logic;          --- Controller Data Read
-----
signal CntDataInput   :std_logic_vector(7 downto 0); --- Controller Input Data
-----
signal CntModeReg     :std_logic_vector(7 downto 0); --- Controller Mode Input
signal CntStatusReg   :std_logic_vector(7 downto 0); --- Controller Status Output
signal CntIntReg      :std_logic_vector(7 downto 0); --- Interrupt Status Output/Reset
signal CntIntMaskReg  :std_logic_vector(7 downto 0); --- Interrupt Mask Data
signal CntIDdtReg     :std_logic_vector(7 downto 0); --- Controller ID Output
signal ECC_OutReg     :std_logic_vector(7 downto 0); --- ECC Data Output
-----
signal SMediaInput    :std_logic_vector(7 downto 0); --- SmartMedia(TM) Input Data
-----
signal SMediaReady    :std_logic;          --- SmartMedia(TM) Ready
signal SMediaCardIn   :std_logic;          --- SmartMedia(TM) Card In
signal SMediaEjctReq  :std_logic;          --- SmartMedia(TM) Eject Request
-----
signal SEL_DATA       :std_logic;          --- Select Data Register
signal SEL_MODE       :std_logic;          --- Select Mode Register
signal SEL_IntREG     :std_logic;          --- Select Interrupt Register
signal SEL_IntMASK    :std_logic;          --- Select Interrupt Register
-----
signal MODE_ReadECC   :std_logic;          --- Select Read ECC Data Mode
signal MODE_ReadID    :std_logic;          --- Select Read ID Data Mode
signal MODE_RstECC    :std_logic;          --- Select Reset ECC Data Mode
signal MODE_EnbECC    :std_logic;          --- Select ECC Calculate Eneble
-----
signal CntPower       :std_logic;          --- SmartMedia(TM) Power
signal CntCardIn      :std_logic;          --- Card In Event
signal CntCardOut     :std_logic;          --- Card Out Event
signal CntEjctReq     :std_logic;          --- Eject Switch Event
signal CntMediaLock   :std_logic;          --- Media Locked
signal CntMediaEjct   :std_logic;          --- Media Eject
-----
signal CntCounter     :std_logic_vector(3 downto 0); --- Controller Local Counter
signal CntRstInt      :std_logic;          --- Interrupt Reset
signal CntRstIntReg   :std_logic_vector(3 downto 0); --- Interrupt Register Reset
signal CntRstReq      :std_logic;          --- Eject Switch Request Reset
-----
signal ECC_Reset      :std_logic;          --- ECC Logic Reset
signal ECC_Clock      :std_logic;          --- ECC Logic Clock
signal ECC_Input      :std_logic_vector( 7 downto 0); --- ECC Logic Input Data
signal ECC_LineAddr   :std_logic_vector( 8 downto 0); --- ECC Logic Line Data Address
signal ECC_LineParity :std_logic_vector(31 downto 0); --- ECC Logic Line Parity
signal ECC_ColumnParity :std_logic_vector(11 downto 0); --- ECC Logic Column Parity

```

```

begin
-----
Reset      <= not RST;
CntRstInt  <= (not RST) or (not ChipEnable);
-----
SPON       <= not CntPower;
-----
CntWrite   <= WE;
CntRead    <= RE;
CntDataInput <= DATA;
-----
SMediaInput  <= SDATA;
SMediaReady  <= SBSY;
SMediaCardIn <= not SCD;
SMediaEjctReq <= not EJSW;
-----
CntStatusReg(0) <= (not SWPD) and CntPower;
CntStatusReg(1) <= CntEjctReq;
CntStatusReg(2) <= not SCD;
CntStatusReg(3) <= CntCardIN or CntCardOut;
CntStatusReg(4) <= CntPower;
CntStatusReg(5) <= '0';
CntStatusReg(6) <= not SLVD;
CntStatusReg(7) <= (not SBSY) and CntPower;
-----
CntIntReg(4) <= '0';
CntIntReg(5) <= '0';
CntIntReg(6) <= '0';
CntIntReg(7) <= '0';
-----
CntIntMaskReg(4) <= '0';
CntIntMaskReg(5) <= '0';
CntIntMaskReg(6) <= '0';
-----

-----
--- SMIL CHIP REGISTER SELECT
-----

process (CE,S)
begin
    if ((CE xor S)="0000") then
        ChipEnable <= '1';
    else
        ChipEnable <= '0';
    end if;
end process;

process (ChipEnable, IREG, CSEL)
begin
    SEL_DATA    <= '0';
    SEL_MODE    <= '0';
    SEL_IntREG  <= '0';
    SEL_IntMASK <= '0';

    if (ChipEnable='1') then
        if (IREG='0') then
            if (CSEL='0') then
                SEL_DATA <= '1';
            else

```

```

        SEL_MODE <= '1';
    end if;
    elsif (CSEL='0') then
        SEL_IntREG <= '1';
    else
        SEL_IntMASK <= '1';
    end if;
end if;
end process;

process (CntModeReg)
begin
    MODE_ReadID <= '0';
    MODE_ReadECC <= '0';
    MODE_RstECC <= '0';
    MODE_EnbECC <= '0';

    if (CntModeReg(6 downto 4)="100") then
        MODE_ReadID <= '1';
    elsif (CntModeReg(6 downto 4)="101") then
        MODE_ReadECC <= '1';
    elsif (CntModeReg(6 downto 5)="11") then
        MODE_RstECC <= '1';
    elsif (CntModeReg(6 downto 4)="011") then
        MODE_EnbECC <= '1';
    end if;
end process;

-----
--- Controller Data Read
-----

process (SEL_DATA,SEL_MODE,SEL_IntREG,SEL_IntMASK,CntRead,MODE_ReadID,MODE_ReadECC,
        CntIDdtReg,ECC_OutReg,SMediaInput,CntStatusReg,CntIntReg,CntIntMaskReg)
    variable OutDataReg :std_logic_vector(7 downto 0);
begin
    if (MODE_ReadID='1') then
        OutDataReg := CntIDdtReg;
    elsif (MODE_ReadECC='1') then
        OutDataReg := ECC_OutReg;
    else
        OutDataReg := SMediaInput;
    end if;

    if (CntRead='0') then
        if (SEL_DATA='1') then
            DATA <= OutDataReg;
        elsif (SEL_MODE='1') then
            DATA <= CntStatusReg;
        elsif (SEL_IntREG='1') then
            DATA <= CntIntReg;
        elsif (SEL_IntMASK='1') then
            DATA <= CntIntMaskReg;
        else
            DATA <= "ZZZZZZZZ";
        end if;
    else
        DATA <= "ZZZZZZZZ";
    end if;
end if;

```

```
end process;
```

```
-----  
--- Controller Data Write  
-----
```

```
process (Reset,SEL_MODE,CntWrite,CntDataInput)  
begin  
    if (Reset='1') then  
        CntModeReg <= (others => '0');  
    elsif (CntWrite'event and CntWrite='1') then  
        if (SEL_MODE='1') then  
            CntModeReg <= CntDataInput;  
        end if;  
    end if;  
end process;
```

```
-----  
process (Reset,SEL_IntMASK,CntWrite,CntDataInput)  
begin  
    if (Reset='1') then  
        CntIntMaskReg(0) <= '0';  
        CntIntMaskReg(1) <= '0';  
        CntIntMaskReg(2) <= '0';  
        CntIntMaskReg(3) <= '0';  
        CntIntMaskReg(7) <= '0';  
    elsif (CntWrite'event and CntWrite='1') then  
        if (SEL_IntMASK='1') then  
            CntIntMaskReg(0) <= CntDataInput(0);  
            CntIntMaskReg(1) <= CntDataInput(1);  
            CntIntMaskReg(2) <= CntDataInput(2);  
            CntIntMaskReg(3) <= CntDataInput(3);  
            CntIntMaskReg(7) <= CntDataInput(7);  
        end if;  
    end if;  
end process;
```

```
-----  
--- Controller Interrupt Logic  
-----
```

```
process (CntRstInt,SEL_IntREG,CntWrite,CntDataInput)  
begin  
    if (CntRstInt='1') then  
        CntRstIntReg <= "0000";  
    elsif (CntWrite'event and CntWrite='1') then  
        if (SEL_IntREG='1') then  
            CntRstIntReg <= CntDataInput(3 downto 0);  
        end if;  
    end if;  
end process;
```

```
-----  
process (Reset,SMediaReady,CntRstIntReg)  
begin  
    if (Reset='1' or CntRstIntReg(0)='1') then  
        CntIntReg(0) <= '0';  
    elsif (SMediaReady'event and SMediaReady='1') then  
        CntIntReg(0) <= '1';  
    end if;  
end process;
```

```

        end if;
    end process;

    process (Reset, SMediaEjctReq, CntRstIntReg)
    begin
        if (Reset='1' or CntRstIntReg(1)='1') then
            CntIntReg(1) <= '0';
        elsif (SMediaEjctReq'event and SMediaEjctReq='1') then
            if (CntMediaLock='1') then
                CntIntReg(1) <= '1';
            end if;
        end if;
    end process;

    process (Reset, SMediaCardIn, CntRstIntReg)
    begin
        if (Reset='1' or CntRstIntReg(2)='1') then
            CntIntReg(2) <= '0';
        elsif (SMediaCardIn'event and SMediaCardIn='0') then
            CntIntReg(2) <= '1';
        end if;
    end process;

    process (Reset, SMediaCardIn, CntRstIntReg)
    begin
        if (Reset='1' or CntRstIntReg(3)='1') then
            CntIntReg(3) <= '0';
        elsif (SMediaCardIn'event and SMediaCardIn='1') then
            CntIntReg(3) <= '1';
        end if;
    end process;

    -----
    process (CntIntReg, CntIntMaskReg)
    begin
        if (CntIntMaskReg(7)='1') then
            if ((CntIntReg and CntIntMaskReg)="00000000") then
                INT <= 'Z';
            else
                INT <= '0';
            end if;
        else
            INT <= 'Z';
        end if;
    end process;

    -----
    --- SmartMedia(TM) Control Logic
    -----
    process (SEL_DATA, CntRead, CntWrite, CntDataInput, CntPower, CntModeReg)
    begin
        if (CntPower='0') then
            SRE <= 'Z';
            SWE <= 'Z';
            SDATA <= (others => '0');
            -----
            SCE <= 'Z';
            SCLE <= '0';
        end if;
    end process;

```

```

        SALE <= '0';
        SWP  <= '0';
    else
        if (SEL_DATA='1' and CntModeReg(6)='0') then
            SRE  <= CntRead;
            SWE  <= CntWrite;
            if (CntRead='1') then
                SDATA <= CntDataInput;
            else
                SDATA <= (others => 'Z');
            end if;
        else
            SRE  <= '1';
            SWE  <= '1';
            SDATA <= (others => '0');
        end if;
        -----
        SCE  <= not CntModeReg(4);
        SCLE <= CntModeReg(0);
        SALE <= CntModeReg(1);
        SWP  <= CntModeReg(7);
    end if;
end process;

-----
--- SLED/EJCT/LOCK Control (for Open Drain)
-----

process (CntModeReg,CntPower)
begin
    if (CntModeReg(2)='1' and CntPower='1') then
        SLED <= '0';
    else
        SLED <= 'Z';
    end if;
end process;

process (SMediaEjctReq,CntMediaEjct,CntMediaLock)
begin
    if (CntMediaEjct='1') then
        EJCT <= '0';
    elsif (CntMediaLock='0' and SMediaEjctReq='1') then
        EJCT <= '0';
    else
        EJCT <= 'Z';
    end if;
end process;

process (CntMediaLock)
begin
    if (CntMediaLock='1') then
        LOCK <= '0';
    else
        LOCK <= 'Z';
    end if;
end process;

-----
--- SmartMedia(TM) Power Control Logic

```

```

-----
process (Reset,SEL_MODE,CntWrite,SMediaCardIn)
begin
    if (Reset='1' or SMediaCardIn='0') then
        CntPower <= '0';
    elsif (CntWrite'event and CntWrite='1') then
        if (SEL_MODE='1') then
            if (CntDataInput(3)='1') then
                if (CntDataInput(2)='1') then
                    CntPower <= '1';
                else
                    CntPower <= '0';
                end if;
            end if;
        end if;
    end if;
end process;

```

---- Media Eject/Lock Logic

```

process (Reset,SEL_MODE,CntWrite)
begin
    if (Reset='1') then
        CntMediaEjct <= '0';
        CntMediaLock <= '0';
    elsif (CntWrite'event and CntWrite='1') then
        if (SEL_MODE='1') then
            if (CntDataInput="01101000") then
                CntMediaEjct <= '1';
            elsif (CntDataInput="00001000") then
                CntMediaEjct <= '0';
            elsif (CntDataInput="01101100") then
                CntMediaLock <= '1';
            elsif (CntDataInput="00001100") then
                CntMediaLock <= '0';
            end if;
        end if;
    end if;
end process;

```

--- SmartMedia(TM) Card Status Change Check (Card In/Out)

```

process (Reset,SEL_MODE,CntWrite,SMediaCardIn)
begin
    if (Reset='1' or (SEL_MODE='1' and CntWrite='0')) then
        CntCardOut <= '0';
    elsif (SMediaCardIn'event and SMediaCardIn='0') then
        CntCardOut <= '1';
    end if;
end process;

```

```

process (Reset,SEL_MODE,CntWrite,SMediaCardIn)
begin
    if (Reset='1' or (SEL_MODE='1' and CntWrite='0')) then
        CntCardIn <= '0';
    elsif (SMediaCardIn'event and SMediaCardIn='1') then

```



```

        CntCardIn <= '1';
    end if;
end process;

-----
--- SmartMedia(TM) Eject Request (Eject Switch)
-----
process (CntRstInt,CntWrite)
begin
    if (CntRstInt='1') then
        CntRstReq <= '0';
    elsif (CntWrite'event and CntWrite='1') then
        if (SEL_MODE='1' and CntDataInput(3)='1') then
            CntRstReq <= '1';
        end if;
    end if;
end process;

process (Reset,CntRstReq,SEL_MODE,CntWrite,SMediaEjctReq)
begin
    if (Reset='1' or CntRstReq='1') then
        CntEjctReq <= '0';
    elsif (SMediaEjctReq'event and SMediaEjctReq='1') then
        if (CntMediaLock='1') then
            CntEjctReq <= '1';
        end if;
    end if;
end process;

-----
--- Controller ID & ECC Register Output Counter
-----
process (CntRead,MODE_ReadID,MODE_ReadECC)
begin
    if (MODE_ReadID='0' and MODE_ReadECC='0') then
        CntCounter <= (others => '1');
    elsif (CntRead'event and CntRead='0') then
        if (SEL_DATA='1') then
            CntCounter <= CntCounter + '1';
        end if;
    end if;
end process;

-----
--- Controller ID Data Output ( 12Byte )
-----
process (CntCounter)
begin
    case CntCounter is
        when "0000" => CntIDdtReg <= "01010011";    --- 'S': 0x53
        when "0001" => CntIDdtReg <= "01001101";    --- 'M': 0x4D
        when "0010" => CntIDdtReg <= "01001001";    --- 'I': 0x49
        when "0011" => CntIDdtReg <= "01001100";    --- 'L': 0x4C
        when "0100" => CntIDdtReg <= "00100000";    --- ' ': 0x20
        when "0101" => CntIDdtReg <= "00110001";    --- '1': 0x31
        when "0110" => CntIDdtReg <= "00101110";    --- ' ': 0x2E
        when "0111" => CntIDdtReg <= "00110000";    --- '0': 0x30
        when "1000" => CntIDdtReg <= "00110000";    --- '0': 0x30
    end case;
end process;

```

```

        when "1001" => CntIDdtReg <= "00000000";    ---      : 0x00
        when "1010" => CntIDdtReg <= "11000000";    ---      : 0xC0
        when "1011" => CntIDdtReg <= "00000000";    ---      : 0x00
        when others => CntIDdtReg <= "00000000";
    end case;
end process;

-----
--- ECC Logic Data Output
-----
process (CntCounter,ECC_LineParity,ECC_ColumnParity)
begin
    case CntCounter is
        when "0000" => ECC_OutReg <= ECC_LineParity(7 downto 0);
        when "0001" => ECC_OutReg <= ECC_LineParity(15 downto 8);
        when "0010" => ECC_OutReg <= ECC_ColumnParity(5 downto 0) & "11";
        when "0011" => ECC_OutReg <= ECC_LineParity(23 downto 16);
        when "0100" => ECC_OutReg <= ECC_LineParity(31 downto 24);
        when "0101" => ECC_OutReg <= ECC_LineParity(11 downto 6) & "11";
        when others => ECC_OutReg <= "00000000";
    end case;
end process;

-----
--- ECC Logic Input Data
-----
process (CntRead,CntDataInput,SMediaInput)
begin
    if (CntRead='1') then
        ECC_Input <= CntDataInput;
    else
        ECC_Input <= SMediaInput;
    end if;
end process;

process (Reset,SEL_DATA,CntWrite,CntRead,MODE_RstECC)
begin
    if (Reset='1') then
        ECC_Reset <= '1';
    elsif (SEL_DATA='1' and MODE_RstECC='1') then
        ECC_Reset <= not (CntWrite and CntRead);
    else
        ECC_Reset <= '0';
    end if;
end process;

process (SEL_DATA,CntWrite,CntRead,MODE_EnbECC)
begin
    if (SEL_DATA='1' and MODE_EnbECC='1') then
        ECC_Clock <= CntWrite and CntRead;
    else
        ECC_Clock <= '1';
    end if;
end process;

--- *****

```

```

--- *** ECC CALCULATE LOGIC *****
--- *****
process (ECC_Reset,ECC_Clock,ECC_LineAddr)
begin
    if (ECC_Reset='1') then
        ECC_LineAddr <= (others => '1');
    elsif (ECC_Clock'event and ECC_Clock='0') then
        ECC_LineAddr <= ECC_LineAddr +1;
    end if;
end process;

-----
--- Column Parity Generator
-----
process (ECC_Reset,ECC_Clock,ECC_Input)
    variable Parity :std_logic_vector(5 downto 0);
begin
    for I in 0 to 1 loop
        Parity(I)    := ECC_Input(I) xor ECC_Input(I+2)
                        xor ECC_Input(I+4) xor ECC_Input(I+6);
        Parity(I+2) := ECC_Input(I*2) xor ECC_Input(I*2+1)
                        xor ECC_Input(I*2+4) xor ECC_Input(I*2+5);
        Parity(I+4) := ECC_Input(I*4) xor ECC_Input(I*4+1)
                        xor ECC_Input(I*4+2) xor ECC_Input(I*4+3);
    end loop;

    if (ECC_Reset='1') then
        ECC_ColumnParity <= (others => '1');
    elsif (ECC_Clock'event and ECC_Clock='1') then
        if (ECC_LineAddr(8)='0') then
            for I in 0 to 5 loop
                ECC_ColumnParity(I) <= ECC_ColumnParity(I) xor Parity(I);
            end loop;
        else
            for I in 0 to 5 loop
                ECC_ColumnParity(I+6) <= ECC_ColumnParity(I+6) xor Parity(I);
            end loop;
        end if;
    end if;
end process;

-----
--- Line Parity Generator
-----
process (ECC_Reset,ECC_Clock,ECC_Input)
    variable Parity :std_logic;
begin
    Parity := '0';
    for I in 0 to 7 loop
        Parity := ECC_Input(I) xor Parity;
    end loop;

    if (ECC_Reset='1') then
        ECC_LineParity <= (others => '1');
    elsif (ECC_Clock'event and ECC_Clock='1') then
        if (ECC_LineAddr(8)='0') then
            for I in 0 to 7 loop

```

```

        if (ECC_LineAddr(I)='0') then
            ECC_LineParity(I*2) <= ECC_LineParity(I*2) xor Parity;
        else
            ECC_LineParity(I*2+1) <= ECC_LineParity(I*2+1) xor Parity;
        end if;
    end loop;
else
    for I in 0 to 7 loop
        if (ECC_LineAddr(I)='0') then
            ECC_LineParity(I*2+16) <= ECC_LineParity(I*2+16) xor Parity;
        else
            ECC_LineParity(I*2+17) <= ECC_LineParity(I*2+17) xor Parity;
        end if;
    end loop;
end if;
end if;
end process;

```

```

-----
end RTL;

```

End of this documentation