JZ4770

Mobile Application Processor

Peripherals Programming Manual

Release Date: Aug. 15, 2011

北京君正集成电路股份有限公司
Ingenic Semiconductor Co.,Ltd.

JZ4770 Mobile Application Processor

Peripherals Programming Manual

Disclaimer

This documentation is provided for use with Ingenic products. No license to Ingenic property rights is granted. Ingenic assumes no liability, provides no warranty either expressed or implied relating to the usage, or intellectual property right infringement except as provided for by Ingenic Terms and Conditions of Sale.

Ingenic products are not designed for and should not be used in any medical or life sustaining or supporting equipment.

All information in this document should be treated as preliminary. Ingenic may make changes to this document without notice. Anyone relying on this documentation should contact Ingenic for the current documentation and errata.

Ingenic Semiconductor Co., Ltd.

Room 108, Building A, Information Center, Zhongguancun Software Park
8 Dongbeiwang West Road, Haidian District, Beijing, China,
Tel: 86-10-82826661
Fax: 86-10-82825845
Http: //www.ingenic.cn

CONTENTS

TABLES

FIGURES

1 General-Purpose I/O Ports

1.1 Overview

General Purpose I/O Ports (GPIO) is used in generating and capturing application-specific input and output signals. Each port can be programmed as an output, an input or function port that serves certain peripheral. As input, pull up/down can be enabled/disabled for the port and the port also can be configured as level or edge tripped interrupt source.

Features:

- Each port can be configured as an input, an output or an alternate function port
- Each port can be configured as an interrupt source of low/high level or rising/falling edge triggering. Every interrupt source can be masked independently
- Each port has an internal pull-up or pull-down resistor connected. The pull-up/down resistor can be disabled
- GPIO output 6 interrupts, 1 for every group, to INTC

1.1.1 GPIO Port A Summary

Table 1-1 GPIO Port A summary

Bit N	PA N	Pull (U/D)	Shared Function Port Selected by				
			0	1	2	3	Note
0	00	U	sd0(io)	-	-	-	
1	01	U	sd1(io)	-	-	-	
2	02	U	sd2(io)	-	-	-	
3	03	U	sd3(io)	-	-	-	
4	04	U	sd4(io)	-	-	-	
5	05	U	sd5(io)	-	-	-	
6	06	U	sd6(io)	-	-	-	
7	07	U	sd7(io)	-	-	-	
8	08	U	sd8(io)	-	-	-	
9	09	U	sd9(io)	-	-	-	
10	10	U	sd10(io)	-	-	-	
11	11	U	sd11(io)	-	-	-	
12	12	U	sd12(io)	-	-	-	
13	13	U	sd13(io)	-	-	-	
14	14	U	sd14(io)	-	-	-	
15	15	U	sd15(io)	-	-	-	
16	16	U rst-pe	rd_(o)	-	-	-	
17	17	U rst-pe	we_(o)	-	-	-	
18	18	U rst-pe	fre_(o)	msc0_clk(o)	ssi0_clk(o)	-	
19	19	U rst-pe	fwe_(o)	msc0_cmd(io)	ssi0_ce0_(o)	-	
20	20	U	msc0_d0(io)	ssi0_dr(i)	-	-	1
21	21	U rst-pe	cs1_(o)	msc0_d1(io)	ssi0_dt(o)	-	
22	22	U rst-pe	cs2_(o)	msc0_d2(io)	-	-	
23	23	U rst-pe	cs3_(o)	msc0_d3(io)	-	-	
24	24	U rst-pe	cs4_(o)	-	-	-	
25	25	U rst-pe	cs5_(o)	-	-	-	
26	26	U	cs6_(o)	rdwr_(o)	-	-	

		rst-pe					
27	27	U	wait_(i)	-	-	-	
28	28	U	dreq0(i)	-	-	-	
29	29	U	dack0(o)	owi(io)	-	-	
30	30	-	-	-	-	-	6
31	31	-	-	-	-	-	7,11

1.1.2 GPIO Port B Summary

Table 1-2 GPIO Port B summary

Bit N	PB N	Pull (U/D)	Shared Function Port Selected by				
			0	1	2	3	Note
0	00	D rst-pe	sa0_cl(o)	-	-	-	8
1	01	D rst-pe	sa1_al(o)	-	-	-	9
2	02	U	sa2(o)	-	-	-	
3	03	U	sa3(o)	-	-	-	
4	04	U	sa4(o)	dreq1(i)	mii_crs(i)	-	
5	05	U	sa5(o)	dack1(o)	-	-	
6	06	U	cim_pclk(i)	-	-	-	
7	07	U	cim_hsyn(i)	-	-	-	
8	08	U rst-pe	cim_vsyn(i)	-	-	-	
9	09	U	cim_mclk(o)	-	-	epd_pwc(o)	
10	10	D	cim_d0(i)	-	-	epd_pwr0(o)	
11	11	D	cim_d1(i)	-	-	epd_pwr1(o)	
12	12	U	cim_d2(i)	-	-	epd_sce2_(o)	
13	13	U	cim_d3(i)	-	-	epd_sce3_(o)	
14	14	U	cim_d4(i)	-	-	epd_sce4_(o)	
15	15	U	cim_d5(i)	-	-	epd_sce5_(o)	
16	16	D	cim_d6(i)	-	-	epd_pwr2(o)	
17	17	D	cim_d7(i)	-	-	epd_pwr3(o)	
18	18	D	cim_d8(i)	dmic_clk(o)	-	epd_bd0(o)	
19	19	D	cim_d9(i)	dmic_in(i)	-	epd_bd1(o)	
20	20	U	msc2_d0(io)	ssi0_dr(i)	ssi1_dr(i)	tsd0(i)	
21	21	U	msc2_d1(io)	ssi0_dt(o)	ssi1_dt(o)	tsd1(i)	
22	22	U	tsd2(i)	-	-	-	
23	23	U	tsd3(i)	-	-	-	
24	24	U	tsd4(i)	-	-	-	

25	25	U	tsd5(i)	-	-	-	
26	26	U	tsd6(i)	-	-	-	
27	27	U	tsd7(i)	-	-	-	
28	28	U	msc2_clk(o)	ssi0_clk(o)	ssi1_clk(o)	tsclk(i)	
29	29	U rst-pe	msc2_cmd(io)	ssi0_ce0_(o)	ssi1_ce0_(o)	tsstr(i)	
30	30	U	msc2_d2(io)	ssi0_gpc(o)	ssi1_gpc(o)	tsfail(i)	
31	31	U rst-pe	msc2_d3(io)	ssi0_ce1_(o)	ssi1_ce1_(o)	tsfrm(i)	

1.1.3 GPIO Port C Summary

Table 1-3 GPIO Port C summary

Bit N	PC N	Pull (U/D)	Shared Function Port Selected by				
			0	1	2	3	Note
0	00	U	lcd_b0(o)	lcd_rev(o)	-		
1	01	U	lcd_b1(o)	lcd_ps(o)	-	-	
2	02	U	lcd_b2(o)	-	-	-	
3	03	U	lcd_b3(o)	-	-	-	
4	04	U	lcd_b4(o)	-	-	-	
5	05	U	lcd_b5(o)	-	-	-	
6	06	U	lcd_b6(o)	-	-	-	
7	07	U	lcd_b7(o)	-	-	-	
8	08	U	lcd_pclk(o)	-	-	-	
9	09	U	lcd_de(o)	-	-	-	
10	10	U rst-pe	lcd_g0(o)	lcd_spl(o)	-	-	
11	11	U	lcd_g1(o)	-	-	-	
12	12	U	lcd_g2(o)	-	-	-	
13	13	U	lcd_g3(o)	-	-	-	
14	14	U	lcd_g4(o)	-	-	-	
15	15	U	lcd_g5(o)	-	-	-	
16	16	U	lcd_g6(o)	-	-	-	
17	17	U	lcd_g7(o)	-	-	-	
18	18	U	lcd_hsyn(io)	-	-	-	
19	19	U	lcd_vsyn(io)	-	-	-	
20	20	U	lcd_r0(o)	lcd_cls(o)	-	-	
21	21	U	lcd_r1(o)	-	-	-	
22	22	U	lcd_r2(o)	-	-	-	
23	23	U	lcd_r3(o)	-	-	-	

24	24	U	lcd_r4(o)	-	-	-	
25	25	U	lcd_r5(o)	-	-	-	
26	26	U	lcd_r6(o)	-	-	-	
27	27	U	lcd_r7(o)	-	-	-	
28	28	U	uart2_rxd(i)	-	-	-	
29	29	U	uart2_cts_(i)	-	-	-	
30	30	U rst-pe	uart2_txd(o)	-	-	-	
31	31	U rst-pe	uart2_rts_(o)	-	-	-	

1.1.4 GPIO Port D Summary

Table 1-4 GPIO Port D summary

Bit N	PD N	Pull (U/D)	Shared Function Port Selected by				
			0	1	2	3	Note
0	00	U	pcm0_do(o)	-	-	-	
1	01	U	pcm0_clk(io)	-	-	-	
2	02	U	pcm0_syn(io)	-	-	-	
3	03	U	pcm0_di(i)	-	-	-	
4	04	U	ps2_mclk(io)	-	-	-	
5	05	U	ps2_mdata(io)	-	-	-	
6	06	U	ps2_kclk(io)	-	-	-	
7	07	U	ps2_kdata(io)	-	-	-	
8	08	U	scc_data(io)	-	-	-	
9	09	U	scc_clk(o)	clk48m(i)	-	-	
10	10	U	pwm6(io)	-	-	-	
11	11	U	pwm7(io)	-	-	-	
12	12	D	uart3_rxd(i)	bclk0(io)	-	epd_pwr4(o)	
13	13	D	lrclk0(io)	-	-	epd_pwr5(o)	
14	14	U	-	-	-	-	10
15	15	D rst-pe	exclk(o)	-	-	-	
16	16	U	-	-	-	-	
17	17	U	-	-	-	-	2,5
18	18	U	-	-	-	-	3,5
19	19	U	-	-	-	-	4,5
20	20	U	msc1_d0(io)	ssi0_dr(i)	ssi1_dr(i)	-	
21	21	U	msc1_d1(io)	ssi0_dt(o)	ssi1_dt(o)	-	
22	22	U	msc1_d2(io)	ssi0_gpc(o)	ssi1_gpc(o)	-	

23	23	U rst-pe	msc1_d3(io)	ssi0_ce1_(o)	ssi1_ce1_(o)	-	
24	24	U	msc1_clk(o)	ssi0_clk(o)	ssi1_clk(o)	-	
25	25	U rst-pe	msc1_cmd(io)	ssi0_ce0_(o)	ssi1_ce0_(o)	-	
26	26	U	uart1_rxd(i)	mii_rxd2(i)	-	-	
27	27	U	uart1_cts_(i)	mii_rxd3(i)	-	-	
28	28	U rst-pe	uart1_txd(o)	mii_txd2(o)	-	-	
29	29	U rst-pe	uart1_rts_(o)	mii_txd3(o)	-	-	
30	30	U	i2c0_sda(io)	-	-	-	
31	31	U	i2c0_sck(io)	-	-	-	

1.1.5 GPIO Port E Summary

Table 1-5 GPIO Port E summary

Bit N	PE N	Pull (U/D)	Shared Function Port Selected by				
			0	1	2	3	Note
0	00	D	pwm0(io)	-	-	-	
1	01	D	pwm1(o)	-	-	-	
2	02	U	pwm2(o)	-	-	-	
3	03	U	pwm3(io)	-	-	-	
4	04	U	pwm4(io)	-	-	-	
5	05	U rst-pe	pwm5(io)	uart3_txd(o)	sclk_rstn(o)	-	
6	06	U	aic0_sdati(i)	-	-	epd_pwr6(o)	
7	07	D	aic0_sdato(o)	-	-	epd_pwr7(o)	
8	08	U	uart3_cts_(i)	bclk0_ad(io)	-	-	
9	09	U rst-pe	uart3_rts_(o)	lrclk0_ad(io)	-	-	
10	10	D rst-pe	drvvbus(o)	-	-	-	
11	11	U	sdato1(o)	-	-	-	
12	12	U	sdato2(o)	bclk1_ad(io)	-	-	
13	13	U	sdato3(o)	lrclk1_ad(io)	-	-	
14	14	U	ssi0_dr(i)	ssi1_dr(i)	-	-	
15	15	U	ssi0_clk(o)	ssi1_clk(o)	-	-	
16	16	U rst-pe	ssi0_ce0_(o)	ssi1_ce0_(o)	-	-	

17	17	U	ssi0_dt(o)	ssi1_dt(o)	-	-	
18	18	U rst-pe	ssi0_ce1_(o)	ssi1_ce1_(o)	-	-	
19	19	U	ssi0_gpc(o)	ssi1_gpc(o)	-	-	
20	20	U	msc0_d0(io)	msc1_d0(io)	msc2_d0(io)	-	
21	21	U	msc0_d1(io)	msc1_d1(io)	msc2_d1(io)	-	
22	22	U	msc0_d2(io)	msc1_d2(io)	msc2_d2(io)	-	
23	23	U	msc0_d3(io)	msc1_d3(io)	msc2_d3(io)	-	
24	24	U	msc0_d4(io)	msc1_d4(io)	msc2_d4(io)	-	
25	25	U	msc0_d5(io)	msc1_d5(io)	msc2_d5(io)	-	
26	26	U	msc0_d6(io)	msc1_d6(io)	msc2_d6(io)	-	
27	27	U	msc0_d7(io)	msc1_d7(io)	msc2_d7(io)	-	
28	28	U	msc0_clk(o)	msc1_clk(o)	msc2_clk(o)	-	
29	29	U	msc0_cmd(io)	msc1_cmd(io)	msc2_cmd(io)	-	
30	30	U	i2c1_sda(io)	-	-	-	
31	31	U	i2c1_sck(io)	-	-	-	

1.1.6 GPIO Port F Summary

Table 1-6 GPIO Port F summary

Bit N	PF N	Pull (U/D)	Shared Function Port Selected by				
			0	1	2	3	Note
0	00	U	uart0_rxd(i)	gps_clk(i)	-	-	
1	01	U rst-pe	uart0_cts_(i)	gps_mag(i)	-	-	
2	02	U rst-pe	uart0_rts_(o)	gps_sig(i)	-	-	
3	03	U rst-pe	uart0_txd(o)	-	-	-	
4	04	D	mii_txd0(o)	-	-	-	
5	05	D	mii_txd1(o)	-	-	-	
6	06	D	mii_txclk(i)	-	-	-	
7	07	D	mii_rxclk(i)	-	-	-	
8	08	D	mii_rxer(i)	-	-	-	
9	09	D	mii_rxdv(i)	-	-	-	
10	10	D	mii_rxd0(i)	-	-	-	
11	11	D	mii_rxd1(i)	-	-	-	
12	12	U	mii_txen(o)	pcm1_do(o)	-	-	
13	13	U	mii_mdc(o)	pcm1_clk(io)	-	-	
14	14	U	mii_mdio(io)	pcm1_syn(io)	-	-	

15	15	U	mii_col(i)	pcm1_di(i)	-	-	
16	16	D	cim_d10(i)	-	i2c2_sda(io)	epd_bd2(o)	
17	17	D	cim_d11(i)	-	i2c2_sck(io)	epd_bd3(o)	
18	18	U	-	-	-		
19	19	D	-	-	-		
20	20	D	-	-	-		
21	21	U	-	-	-		
22	22	D	-	-	-		

NOTES:

1 If NAND flash is used, this pin must be used as NAND FRB.

2 PD17: GPIO group D bit 17 is used as BOOT_SEL0 input during boot.

3 PD18: GPIO group D bit 18 is used as BOOT_SEL1 input during boot.

4 PD19: GPIO group D bit 19 is used as BOOT_SEL2 input during boot.

5 BOOT_SEL2, BOOT_SEL1, BOOT_SEL0 are used to select boot source and function during the processor boot.

6 PA30: GPIO group A bit 30 can only be used as input and interrupt, no pull-up and pull-down. It is also used to select the function between PS2 function and JTAG function of JTAG/UART3/PS2 Pins(TCK_UART3_RTS_PS2_MCLK, TMS_UART3_CTS_PS2_MDATA, TDI_UART3_RxD_PS2_KCLK and TDO_UART3_TxD_PS2_KDATA), which share the same set of pins.
When PA30.function1 is false, select JTAG function.
When PA30.function1 is true, select PS2 function.

7 PA31: GPIO group A bit 31. No corresponding pin exists for this GPIO.
It is only used to select the function between UART and JTAG of JTAG/UART3/PS2 Pins (TCK_UART3_RTS_PS2_MCLK,TMS_UART3_CTS_PS2_MDATA,TDI_UART3_RxD_PS2_KCLK and TDO_UART3_TxD_PS2_KDATA), which share the same set of pins, by using register PASEL [31].
When PA31.function1 is false, select JTAG function.
When PA31.function1 is true, select UART function.

8 If NAND flash is used, this pin must be used as NAND CLE.

9 If NAND flash is used, this pin must be used as NAND ALE.

10 PD14 : this pin is generally used as RTCLK output. If this pin is intended to be used as GPIO or interrupt, CK32CTL(CKPCR[2:1] in RTC) should be configured to 2'b10.Please refer to RTC spec.

11 The pull enable of PA31 is used to control UART and JTAG of JTAG/UART3/PS2 Pins.

1.2 Registers Description

Table 1-7 summarized all memory-mapped registers, which can be programmed to operate GPIO port and alternate function port sharing configuration.

All registers are in 32-bits width. Usually, 1 bit in the register affects a corresponding GPIO port and every GPIO port can be operated independently.

Table 1-7 GPIO Registers

Name	Description	RW	Reset Value	Address	Size
GPIO PORT A					
PAPIN	PORT A PIN Level Register	R	0x????????	0x10010000	32
PAINT	PORT A Interrupt Register	RW	0x00000000	0x10010010	32
PAINTS	PORT A Interrupt Set Register	W	0x????????	0x10010014	32
PAINTC	PORT A Interrupt Clear Register	W	0x????????	0x10010018	32
PAMSK	PORT A Interrupt Mask Register	RW	0xFFFFFFFF	0x10010020	32
PAMSKS	PORT A Interrupt Mask Set Register	W	0x????????	0x10010024	32
PAMSKC	PORT A Interrupt Mask Clear Register	W	0x????????	0x10010028	32
PAPAT1	PORT A Pattern 1 Register	RW	0xFFFFFFFF	0x10010030	32
PAPAT1S	PORT A Pattern 1 Set Register	W	0x????????	0x10010034	32
PAPAT1C	PORT A Pattern 1 Clear Register	W	0x????????	0x10010038	32
PAPAT0	PORT A Pattern 0 Register	RW	0x00000000	0x10010040	32
PAPAT0S	PORT A Pattern 0 Set Register	W	0x????????	0x10010044	32
PAPAT0C	PORT A Pattern 0 Clear Register	W	0x????????	0x10010048	32
PAFLG	PORT A FLAG Register	R	0x00000000	0x10010050	32
PAFLGC	PORT A FLAG Clear Register	W	0x????????	0x10010058	32
PAPEN	PORT A PULL Disable Register	RW	0x00000000	0x10010070	32
PAPENS	PORT A PULL Disable Set Register	W	0x????????	0x10010074	32
PAPENC	PORT A PULL Disable Clear Register	W	0x????????	0x10010078	32
PADRVL	-	RW	0x00000000	0x10010080	32
PADRVLS	-	W	0x????????	0x10010084	32
PADRVLC	-	W	0x????????	0x10010088	32
PADIR	-	RW	0x00000000	0x10010090	32
PADIRS	-	W	0x????????	0x10010094	32
PADIRC	-	W	0x????????	0x10010098	32
PADRVH	-	RW	0x00000000	0x100100A0	32
PADRVHS	-	W	0x????????	0x100100A4	32
PADRVHC	-	W	0x????????	0x100100A8	32
GPIO PORT B					
PBPIN	PORT B PIN Level Register	R	0x????????	0x10010100	32
PBINT	PORT B Interrupt Register	RW	0x00000000	0x10010110	32

PBINTS	PORT B Interrupt Set Register	W	0x????????	0x10010114	32
PBINTC	PORT B Interrupt Clear Register	W	0x????????	0x10010118	32
PBMSK	PORT B Interrupt Mask Register	R	0xFFFFFFFF	0x10010120	32
PBMSKS	PORT B Interrupt Mask Set Register	W	0x????????	0x10010124	32
PBMSKC	PORT B Interrupt Mask Clear Register	W	0x????????	0x10010128	32
PBPAT1	PORT B Pattern 1 Register	R	0xFFFFFFFF	0x10010130	32
PBPAT1S	PORT B Pattern 1 Set Register	W	0x????????	0x10010134	32
PBPAT1C	PORT B Pattern 1 Clear Register	W	0x????????	0x10010138	32
PBPAT0	PORT B Pattern 0 Register	RW	0x00000000	0x10010140	32
PBPAT0S	PORT B Pattern 0 Set Register	W	0x????????	0x10010144	32
PBPAT0C	PORT B Pattern 0 Clear Register	W	0x????????	0x10010148	32
PBFLG	PORT B FLAG Register	R	0x00000000	0x10010150	32
PBFLGC	PORT B FLAG Clear Register	W	0x????????	0x10010158	32
PBPEN	PORT B PULL Disable Register	RW	0x00000000	0x10010170	32
PBPENS	PORT B PULL Disable Set Register	W	0x????????	0x10010174	32
PBPENC	PORT B PULL Disable Clear Register	W	0x????????	0x10010178	32
PBDRVL	-	RW	0x00000000	0x10010180	32
PBDRVLS	-	W	0x????????	0x10010184	32
PBDRVLC	-	W	0x????????	0x10010188	32
PBDIR	-	RW	0x00000000	0x10010190	32
PBDIRS	-	W	0x????????	0x10010194	32
PBDIRC	-	W	0x????????	0x10010198	32
PBDRVH	-	RW	0x00000000	0x100101A0	32
PBDRVHS	-	W	0x????????	0x100101A4	32
PBDRVHC	-	W	0x????????	0x100101A8	32
GPIO PORT C					
PCPIN	PORT C PIN Level Register	R	0x????????	0x10010200	32
PCINT	PORT C Interrupt Register	RW	0x00000000	0x10010210	32
PCINTS	PORT C Interrupt Set Register	W	0x????????	0x10010214	32
PCINTC	PORT C Interrupt Clear Register	W	0x????????	0x10010218	32
PCMSK	PORT C Interrupt Mask Register	R	0xFFFFFFFF	0x10010220	32
PCMSKS	PORT C Interrupt Mask Set Register	W	0x????????	0x10010224	32
PCMSKC	PORT C Interrupt Mask Clear Register	W	0x????????	0x10010228	32
PCPAT1	PORT C Pattern 1 Register	R	0xFFFFFFFF	0x10010230	32
PCPAT1S	PORT C Pattern 1 Set Register	W	0x????????	0x10010234	32
PCPAT1C	PORT C Pattern 1 Clear Register	W	0x????????	0x10010238	32
PCPAT0	PORT C Pattern 0 Register	RW	0x00000000	0x10010240	32
PCPAT0S	PORT C Pattern 0 Set Register	W	0x????????	0x10010244	32
PCPAT0C	PORT C Pattern 0 Clear Register	W	0x????????	0x10010248	32
PCFLG	PORT C FLAG Register	R	0x00000000	0x10010250	32
PCFLGC	PORT C FLAG Clear Register	W	0x????????	0x10010258	32

PCPEN	PORT C PULL Disable Register	RW	0x00000000	0x10010270	32
PCPENS	PORT C PULL Disable Set Register	W	0x????????	0x10010274	32
PCPENC	PORT C PULL Disable Clear Register	W	0x????????	0x10010278	32
PCDRVL	-	RW	0x00000000	0x10010280	32
PCDRVLS	-	W	0x????????	0x10010284	32
PCDRVLC	-	W	0x????????	0x10010288	32
PCDIR	-	RW	0x00000000	0x10010290	32
PCDIRS	-	W	0x????????	0x10010294	32
PCDIRC	-	W	0x????????	0x10010298	32
PCDRVH	-	RW	0x00000000	0x100102A0	32
PCDRVHS	-	W	0x????????	0x100102A4	32
PCDRVHC	-	W	0x????????	0x100102A8	32
GPIO PORT D					
PDPIN	PORT D PIN Level Register	R	0x????????	0x10010300	32
PDINT	PORT D Interrupt Register	RW	0x00000000	0x10010310	32
PDINTS	PORT D Interrupt Set Register	W	0x????????	0x10010314	32
PDINTC	PORT D Interrupt Clear Register	W	0x????????	0x10010318	32
PDMSK	PORT D Interrupt Mask Register	R	0xFFFFFFFF	0x10010320	32
PDMSKS	PORT D Interrupt Mask Set Register	W	0x????????	0x10010324	32
PDMSKC	PORT D Interrupt Mask Clear Register	W	0x????????	0x10010328	32
PDPAT1	PORT D Pattern 1 Register	R	0xFFFFFFFF	0x10010330	32
PDPAT1S	PORT D Pattern 1 Set Register	W	0x????????	0x10010334	32
PDPAT1C	PORT D Pattern 1 Clear Register	W	0x????????	0x10010338	32
PDPAT0	PORT D Pattern 0 Register	RW	0x00000000	0x10010340	32
PDPAT0S	PORT D Pattern 0 Set Register	W	0x????????	0x10010344	32
PDPAT0C	PORT D Pattern 0 Clear Register	W	0x????????	0x10010348	32
PDFLG	PORT D FLAG Register	R	0x00000000	0x10010350	32
PDFLGC	PORT D FLAG Clear Register	W	0x????????	0x10010358	32
PDPEN	PORT D PULL Disable Register	RW	0x00000000	0x10010370	32
PDPENS	PORT D PULL Disable Set Register	W	0x????????	0x10010374	32
PDPENC	PORT D PULL Disable Clear Register	W	0x????????	0x10010378	32
PDDRVL	-	RW	0x00000000	0x10010380	32
PDDRVLS	-	W	0x????????	0x10010384	32
PDDRVLC	-	W	0x????????	0x10010388	32
PDDIR	-	RW	0x00000000	0x10010390	32
PDDIRS	-	W	0x????????	0x10010394	32
PDDIRC	-	W	0x????????	0x10010398	32
PDDRVH	-	RW	0x00000000	0x100103A0	32
PDDRVHS	-	W	0x????????	0x100103A4	32
PDDRVHC	-	W	0x????????	0x100103A8	32
GPIO PORT E					

PEPIN	PORT E PIN Level Register	R	0x????????	0x10010400	32
PEINT	PORT E Interrupt Register	RW	0x00000000	0x10010410	32
PEINTS	PORT E Interrupt Set Register	W	0x????????	0x10010414	32
PEINTC	PORT E Interrupt Clear Register	W	0x????????	0x10010418	32
PEMSK	PORT E Interrupt Mask Register	R	0xFFFFFFFF	0x10010420	32
PEMSKS	PORT E Interrupt Mask Set Register	W	0x????????	0x10010424	32
PEMSKC	PORT E Interrupt Mask Clear Register	W	0x????????	0x10010428	32
PEPAT1	PORT E Pattern 1 Register	R	0xFFFFFFFF	0x10010430	32
PEPAT1S	PORT E Pattern 1 Set Register	W	0x????????	0x10010434	32
PEPAT1C	PORT E Pattern 1 Clear Register	W	0x????????	0x10010438	32
PEPAT0	PORT E Pattern 0 Register	RW	0x00000000	0x10010440	32
PEPAT0S	PORT E Pattern 0 Set Register	W	0x????????	0x10010444	32
PEPAT0C	PORT E Pattern 0 Clear Register	W	0x????????	0x10010448	32
PEFLG	PORT E FLAG Register	R	0x00000000	0x10010450	32
PEFLGC	PORT E FLAG Clear Register	W	0x????????	0x10010458	32
PEPEN	PORT E PULL Disable Register	RW	0x00000000	0x10010470	32
PEPENS	PORT E PULL Disable Set Register	W	0x????????	0x10010474	32
PEPENC	PORT E PULL Disable Clear Register	W	0x????????	0x10010478	32
PEDRVL	-	RW	0x00000000	0x10010480	32
PEDRVLS	-	W	0x????????	0x10010484	32
PEDRVLC	-	W	0x????????	0x10010488	32
PEDIR	-	RW	0x00000000	0x10010490	32
PEDIRS	-	W	0x????????	0x10010494	32
PEDIRC	-	W	0x????????	0x10010498	32
PEDRVH	-	RW	0x00000000	0x100104A0	32
PEDRVHS	-	W	0x????????	0x100104A4	32
PEDRVHC	-	W	0x????????	0x100104A8	32
GPIO PORT F					
PFPIN	PORT F PIN Level Register	R	0x????????	0x10010500	32
PFINT	PORT F Interrupt Register	RW	0x00000000	0x10010510	32
PFINTS	PORT F Interrupt Set Register	W	0x????????	0x10010514	32
PFINTC	PORT F Interrupt Clear Register	W	0x????????	0x10010518	32
PFMSK	PORT F Interrupt Mask Register	R	0xFFFFFFFF	0x10010520	32
PFMSKS	PORT F Interrupt Mask Set Register	W	0x????????	0x10010524	32
PFMSKC	PORT F Interrupt Mask Clear Register	W	0x????????	0x10010528	32
PFPAT1	PORT F Pattern 1 Register	R	0xFFFFFFFF	0x10010530	32
PFPAT1S	PORT F Pattern 1 Set Register	W	0x????????	0x10010534	32
PFPAT1C	PORT F Pattern 1 Clear Register	W	0x????????	0x10010538	32
PFPAT0	PORT F Pattern 0 Register	RW	0x00000000	0x10010540	32
PFPAT0S	PORT F Pattern 0 Set Register	W	0x????????	0x10010544	32
PFPAT0C	PORT F Pattern 0 Clear Register	W	0x????????	0x10010548	32

PFFLG	PORT F FLAG Register	R	0x00000000	0x10010550	32
PFFLGC	PORT F FLAG Clear Register	W	0x????????	0x10010558	32
PFPEN	PORT F PULL Disable Register	RW	0x00000000	0x10010570	32
PFPENS	PORT F PULL Disable Set Register	W	0x????????	0x10010574	32
PFPENC	PORT F PULL Disable Clear Register	W	0x????????	0x10010578	32
PFDRVL	-	RW	0x00000000	0x10010580	32
PFDRVLS	-	W	0x????????	0x10010584	32
PFDRVLC	-	W	0x????????	0x10010588	32
PFDIR	-	RW	0x00000000	0x10010590	32
PFDIRS	-	W	0x????????	0x10010594	32
PFDIRC	-	W	0x????????	0x10010598	32
PFDRVH	-	RW	0x00000000	0x100105A0	32
PFDRVHS	-	W	0x????????	0x100105A4	32
PFDRVHC	-	W	0x????????	0x100105A8	32

NOTE: PX**** in the description of register as follows means PA****, PB****, PC****, PD****, PE**** and PF****.

1.2.1 PORT PIN Level Registers (PxPIN)

PAPIN, PBPIN, PCPIN, PDPIN, PEPIN and PFPIN are six 32-bit PORT PIN level registers. They are read-only registers.

PAPIN, PBPIN, PCPIN, PDPIN, PEPIN, PFPIN — **0x10010000, 0x10010100, 0x10010200, 0x10010300, 0x10010400, 0x10010500**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PINL31	PINL30	PINL29	PINL28	PINL27	PINL26	PINL25	PINL24	PINL23	PINL22	PINL21	PINL20	PINL19	PINL18	PINL17	PINL16	PINL15	PINL14	PINL13	PINL12	PINL11	PINL10	PINL09	PINL08	PINL07	PINL06	PINL05	PINL04	PINL03	PINL02	PINL01	PINL00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	PINL n	Where n = 0 ~ 31 and PINL n = PINL0 ~ PINL31. The PORT PIN level can be read by reading PINL n bit in register PXPIN.	R

1.2.2 PORT Interrupt Registers (PxINT)

PAINT, PBINT, PCINT, PDINT, PEINT and PFINT are six 32-bit interrupt registers.

PAINT, PBINT, PCINT, PDINT, PEINT, PFINT — **0x10010010, 0x10010110, 0x10010210, 0x10010310, 0x10010410, 0x10010510**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16	INT15	INT14	INT13	INT12	INT11	INT10	INT09	INT08	INT07	INT06	INT05	INT04	INT03	INT02	INT01	INT00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
N	INT n	Where n = 0 ~ 31 and INT n = INT31 ~ INT00. Interrupt enable. 0: Corresponding pin is used as device functions or normal gpio 1: Corresponding pin is used as interrupt	RW

1.2.3 PORT Interrupt Set Registers (PxINTS)

PAINTS, PBINTS, PCINTS, PDINTS, PEINTS and PFINTS are six 32-bit interrupt set registers.

PAINTS, PBINTS, PCINTS, PDINTS, PEINTS, PFINTS — **0x10010014, 0x10010114, 0x10010214, 0x10010314, 0x10010414, 0x10010514**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INTS31	INTS30	INTS29	INTS28	INTS27	INTS26	INTS25	INTS24	INTS23	INTS22	INTS21	INTS20	INTS19	INTS18	INTS17	INTS16	INTS15	INTS14	INTS13	INTS12	INTS11	INTS10	INTS09	INTS08	INTS07	INTS06	INTS05	INTS04	INTS03	INTS02	INTS01	INTS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
N	INTS n	Writing 1 to INTS n will set INT n to 1 in register PXINT. Writing 0 to INTS n will no use.	W

1.2.4 PORT Interrupt Clear Registers (PxINTC)

PAINTC, PBINTC, PCINTC, PDINTC, PEINTC and PFINTC are six 32-bit interrupt clear registers.

PAINTC, PBINTC, PCINTC, PDINTC, PEINTC, PFINTC — **0x10010018, 0x10010118, 0x10010218, 0x10010318, 0x10010418, 0x10010518**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INTC31	INTC30	INTC29	INTC28	INTC27	INTC26	INTC25	INTC24	INTC23	INTC22	INTC21	INTC20	INTC19	INTC18	INTC17	INTC16	INTC15	INTC14	INTC13	INTC12	INTC11	INTC10	INTC09	INTC08	INTC07	INTC06	INTC05	INTC04	INTC03	INTC02	INTC01	INTC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	INTC n	Writing 1 to INTC n will set INT n to 0 in register PXINT. Writing 0 to INTC n will no use.	W

1.2.5 PORT Mask Registers (PxMSK)

PAMSK, PBMSK, PCMSK, PDMSK, PEMSK and PFMSK are six 32-bit PORT MASK registers.

PAMSK, PBMSK, PCMSK, PDMSK. PEMSK, PFMSK — **0x10010020, 0x10010120, 0x10010220, 0x10010320, 0x10010420, 0x10010520**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MSK31	MSK30	MSK29	MSK28	MSK27	MSK26	MSK25	MSK24	MSK23	MSK22	MSK21	MSK20	MSK19	MSK18	MSK17	MSK16	MSK15	MSK14	MSK13	MSK12	MSK11	MSK10	MSK09	MSK08	MSK07	MSK06	MSK05	MSK04	MSK03	MSK02	MSK01	MSK00
RST	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	R/W
n	MSK n	Where n = 0 ~ 31 and MSK n = MSK31 ~ MSK0. When PXINT n = 1: 0: Enable the pin as an interrupt source 1: Disable the pin as an interrupt source When PXINT n = 0: 0: Corresponding pin will be used as device function 1: Corresponding pin will be used as gpio	RW

1.2.6 PORT Mask Set Registers (PxMSKS)

PAMSKS, PBMSKS, PCMSKS, PDMSKS, PEMSKS and PFMSKS are six 32-bit PORT MASK set registers.

PAMSKS, PBMSKS, PCMSKS, PDMSKS, PEMSKS, PFMSKS — **0x10010024, 0x10010124, 0x10010224, 0x10010324, 0x10010424, 0x10010524**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MSKS31	MSKS30	MSKS29	MSKS28	MSKS27	MSKS26	MSKS25	MSKS24	MSKS23	MSKS22	MSKS21	MSKS20	MSKS19	MSKS18	MSKS17	MSKS16	MSKS15	MSKS14	MSKS13	MSKS12	MSKS11	MSKS10	MSKS09	MSKS08	MSKS07	MSKS06	MSKS05	MSKS04	MSKS03	MSKS02	MSKS01	MSKS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	MSKS n	Writing 1 to MSKS n will set MSK n to 1 in register PXMSK. Writing 0 to MSKS n will no use.	W

1.2.7 PORT Mask Clear Registers (PxMSKC)

PAMSKC, PBMSKC, PCMSKC, PDMSKC, PEMSKC and PFMSKC are six 32-bit PORT MASK clear registers.

PAMSKS, PBMSKC, PCMSKC, PDMSKC, PEMSKC, PFMSKC — **0x10010028, 0x10010128, 0x10010228, 0x10010328, 0x10010428, 0x10010528**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MSKC31	MSKC30	MSKC29	MSKC28	MSKC27	MSKC26	MSKC25	MSKC24	MSKC23	MSKC22	MSKC21	MSKC20	MSKC19	MSKC18	MSKC17	MSKC16	MSKC15	MSKC14	MSKC13	MSKC12	MSKC11	MSKC10	MSKC09	MSKC08	MSKC07	MSKC06	MSKC05	MSKC04	MSKC03	MSKC02	MSKC01	MSKC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	MSKC n	Writing 1 to MSKC n will set MSK n to 0 in register PXMSK. Writing 0 to MSKC n will no use.	W

1.2.8 PORT PAT1/Direction Registers (PxPAT1)

PAPAT1, PBPAT1, PCPAT1, PDPAT1, PEPAT1 and PFPAT1 are six 32-bit PORT pattern1/direction registers.

PAPAT1, PBPAT1, PCPAT1, PDPAT1, PEPAT1, PFPAT1 — **0x10010030, 0x10010130, 0x10010230, 0x10010330, 0x10010430, 0x10010530**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PAT131	PAT130	PAT129	PAT128	PAT127	PAT126	PAT125	PAT124	PAT123	PAT122	PAT121	PAT120	PAT119	PAT118	PAT117	PAT116	PAT115	PAT114	PAT113	PAT112	PAT111	PAT110	PAT109	PAT108	PAT107	PAT106	PAT105	PAT104	PAT103	PAT102	PAT101	PAT100
RST	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	R/W
n	PAT1 n	Where n = 0 ~ 31 and PAT1 n = PAT131 ~ PAT10. When PINT n = 1 (Interrupt function): 0: Level trigger interrupt 1: Edge trigger interrupt When PINT n = 0 and PMSK = 0 (Device function): 0: Corresponding pin is used as device 0 or device 1 function 1: Corresponding pin is used as device 2 or device 3 function When PINT n = 0 and PMSK = 0 (GPIO function): 0: Corresponding pin is used as gpio output 1: Corresponding pin is used as gpio input	RW

1.2.9 PORT PAT1/Direction Set Registers (PxPAT1S)

PAPAT1S, PBPAT1S, PCPAT1S, PDPAT1S, PEPAT1S and PFPAT1S are six 32-bit PORT pattern1/direction set registers.

PAPAT1S, PBPAT1S, PCPAT1S, PDPAT1S, PEPAT1S, PFPAT1S — **0x10010034, 0x10010134, 0x10010234, 0x10010334, 0x10010434, 0x10010534**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PAT1S31	PAT1S30	PAT1S29	PAT1S28	PAT1S27	PAT1S26	PAT1S25	PAT1S24	PAT1S23	PAT1S22	PAT1S21	PAT1S20	PAT1S19	PAT1S18	PAT1S17	PAT1S16	PAT1S15	PAT1S14	PAT1S13	PAT1S12	PAT1S11	PAT1S10	PAT1S09	PAT1S08	PAT1S07	PAT1S06	PAT1S05	PAT1S04	PAT1S03	PAT1S02	PAT1S01	PAT1S00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	PAT1S n	Writing 1 to PAT1S n will set PAT1 n to 1 in register PXPAT1. Writing 0 to PAT1S n will no use.	W

1.2.10 PORT PAT1/Direction Clear Registers (PxPAT1C)

PAPAT1C, PBPAT1C, PCPAT1C, PPAT1C, PEPAT1C and PFPAT1C are six 32-bit PORT pattern1 or direction clear registers.

PAPAT1S, PBPAT1C, PCPAT1C, **0x10010038, 0x10010138, 0x10010238,**
PDPAT1C, PEPAT1C, PFPAT1C **0x10010338, 0x10010438, 0x10010538**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PAT1C31	PAT1C30	PAT1C29	PAT1C28	PAT1C27	PAT1C26	PAT1C25	PAT1C24	PAT1C23	PAT1C22	PAT1C21	PAT1C20	PAT1C19	PAT1C18	PAT1C17	PAT1C16	PAT1C15	PAT1C14	PAT1C13	PAT1C12	PAT1C11	PAT1C10	PAT1C09	PAT1C08	PAT1C07	PAT1C06	PAT1C05	PAT1C04	PAT1C03	PAT1C02	PAT1C01	PAT1C00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	PAT1C n	Writing 1 to PAT1C n will set PAT1 n to 0 in register PXPAT1. Writing 0 to PAT1C n will no use.	W

1.2.11 PORT PAT0/Data Registers (PxPAT0)

PAPAT0, PBPAT0, PCPAT0, PDPAT0, PEPAT0 and PFPAT0 are six 32-bit PORT Pattern 0 or DATA registers.

PAPAT0, PBPAT0, PCPAT0, **0x10010040, 0x10010140, 0x10010240,**
PDPAT0, PEPAT0, PFPAT0 **0x10010340, 0x10010440, 0x10010540**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PAT031	PAT030	PAT029	PAT028	PAT027	PAT026	PAT025	PAT024	PAT023	PAT022	PAT021	PAT020	PAT019	PAT018	PAT017	PAT016	PAT015	PAT014	PAT013	PAT012	PAT011	PAT010	PAT009	PAT008	PAT007	PAT006	PAT005	PAT004	PAT003	PAT002	PAT001	PAT000
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
N	PAT0 n	Where n = 0 ~ 31 and PAT0 n = PAT00 ~ PAT031. When PINTn = 1 and PPAT1 = 0: 0: Port is low level triggered interrupt input 1: Port is low high triggered interrupt input When PINTn = 1 and PPAT1 = 1: 0: Port is falling edge triggered interrupt input 1: Port is rising edge triggered interrupt input When PINTn = 0 and PMSK = 0 and PPAT1 = 0: 0: Port is pin of device 0 1: Port is pin of device 1 When PINTn = 0 and PMSK = 0 and PPAT1 = 1: 0: Port is pin of device 2	RW

		1: Port is pin of device 3 When PINTn = 0 and PMSK = 1 and PPAT1 = 0: 0: Port is GPIO output 0 1: Port is GPIO output 1	

1.2.12 PORT PAT0/Data Set Registers (PxPAT0S)

PAPAT0S, PBPAT0S, PCPAT0S, PDPAT0S, PEPAT0S and PFPAT0S are six 32-bit PORT Pattern0 or DATA set registers.

PAPAT0S, PBPAT0S, PCPAT0S, PDPAT0S, PEPAT0S, PFPAT0S — **0x10010044, 0x10010144, 0x10010244, 0x10010344, 0x10010444, 0x10010544**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PAT0S31	PAT0S30	PAT0S29	PAT0S28	PAT0S27	PAT0S26	PAT0S25	PAT0S24	PAT0S23	PAT0S22	PAT0S21	PAT0S20	PAT0S19	PAT0S18	PAT0S17	PAT0S16	PAT0S15	PAT0S14	PAT0S13	PAT0S12	PAT0S11	PAT0S10	PAT0S09	PAT0S08	PAT0S07	PAT0S06	PAT0S05	PAT0S04	PAT0S03	PAT0S02	PAT0S01	PAT0S00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	PAT0S n	Writing 1 to PAT0S n will set PAT0 n to 1 in register PXPAT0. Writing 0 to PAT0S n will no use.	W

1.2.13 PORT PAT0/Data Clear Registers (PxPAT0C)

PAPAT0C, PBPAT0C, PCPAT0C, PDPAT0C, PEPAT0C and PFPAT0C are six 32-bit PORT Pattern 0 or DATA clear registers.

PAPAT0C, PBPAT0C, PCPAT0C, PDPAT0C, PEPAT0C, PFPAT0C — **0x10010048, 0x10010148, 0x10010248, 0x10010348, 0x10010448, 0x10010548**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PAT0C31	PAT0C30	PAT0C29	PAT0C28	PAT0C27	PAT0C26	PAT0C25	PAT0C24	PAT0C23	PAT0C22	PAT0C21	PAT0C20	PAT0C19	PAT0C18	PAT0C17	PAT0C16	PAT0C15	PAT0C14	PAT0C13	PAT0C12	PAT0C11	PAT0C10	PAT0C09	PAT0C08	PAT0C07	PAT0C06	PAT0C05	PAT0C04	PAT0C03	PAT0C02	PAT0C01	PAT0C00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	PAT0C n	Writing 1 to PAT0C n will set PAT0 n to 0 in register PXPAT0. Writing 0 to PAT0C n will no use.	W

1.2.14 PORT FLAG Registers (PxFLG)

PAFLG, PBFLG, PCFLG, PDFLG, PEFLG and PFFLG are six 32-bit GPIO FLAG registers. They are read-only registers.

PAFLG, PBFLG, PCFLG, **0x10010050, 0x10010150, 0x10010250,**
PDFLG, PEFLG, PFFLG **0x10010350, 0x10010450, 0x10010550**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FLAG31	FLAG30	FLAG29	FLAG28	FLAG27	FLAG26	FLAG25	FLAG24	FLAG23	FLAG22	FLAG21	FLAG20	FLAG19	FLAG18	FLAG17	FLAG16	FLAG15	FLAG14	FLAG13	FLAG12	FLAG11	FLAG10	FLAG09	FLAG08	FLAG07	FLAG06	FLAG05	FLAG04	FLAG03	FLAG02	FLAG01	FLAG00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	FLAG n	Where n = 0 ~ 31 and FLAG n = FLAG00 ~ FLAG31. FLAG n is interrupt flag bit for checking the interrupt whether to happen. When GPIO is used as interrupt function and the interrupt happened, the FLAG n in PXFLG will be set to 1.	R

1.2.15 PORT FLAG Clear Registers (PxFLGC)

PAFLGC, PBFLGC, PCFLGC, PDFLGC, PEFLGC and PFFLGC are six 32-bit GPIO FLAG Clear registers. They are read-only registers.

PAFLGC, PBFLGC, PCFLGC, **0x10010058, 0x10010158, 0x10010258,**
PDFLGC, PEFLGC, PFFLGC **0x10010358, 0x10010458,0x10010558**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FLAGC31	FLAGC30	FLAGC29	FLAGC28	FLAGC27	FLAGC26	FLAGC25	FLAGC24	FLAGC23	FLAGC22	FLAGC21	FLAGC20	FLAGC19	FLAGC18	FLAGC17	FLAGC16	FLAGC15	FLAGC14	FLAGC13	FLAGC12	FLAGC11	FLAGC10	FLAGC09	FLAGC08	FLAGC07	FLAGC06	FLAGC05	FLAGC04	FLAGC03	FLAGC02	FLAGC01	FLAGC00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	FLAGC n	When GPIO is used as interrupt function and when write 1 to the bit, the bit FLAG n in PXFLG will be cleared.	R

1.2.16 PORT PULL Disable Registers (PxPE)

PAPE, PBPE, PCPE, PDPE, PEPE and PFPE are six 32-bit PORT PULL disable registers.

PAPE, PBPE, PCPE, PDPE, PEPE, PFPE — **0x10010070, 0x10010170, 0x10010270, 0x10010370, 0x10010470, 0x10010570**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PULL31	PULL30	PULL29	PULL28	PULL27	PULL26	PULL25	PULL24	PULL23	PULL22	PULL21	PULL20	PULL19	PULL18	PULL17	PULL16	PULL15	PULL14	PULL13	PULL12	PULL11	PULL10	PULL09	PULL08	PULL07	PULL06	PULL05	PULL04	PULL03	PULL02	PULL01	PULL00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
N	PULL n	Where n = 0 ~ 31 and PULL n = PULL0 ~ PULL31. PULL n is used for setting the port to be PULL UP or PULL DOWN enable. 0: An internal pull up or pull down resistor connects to the port. Up or down is pin dependence 1: No pull up or pull down resistor connects to the port	RW

1.2.17 PORT PULL Set Registers (PxPES)

PAPES, PBPES, PCPES, PDPES, PEPES and PFPES are six 32-bit PORT PULL set registers. They are write-only registers.

PAPES, PBPES, PCPES, PDPES, PEPES, PFPES — **0x10010074, 0x10010174, 0x10010274, 0x10010374, 0x10010474, 0x10010574**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PULLS31	PULLS30	PULLS29	PULLS28	PULLS27	PULLS26	PULLS25	PULLS24	PULLS23	PULLS22	PULLS21	PULLS20	PULLS19	PULLS18	PULLS17	PULLS16	PULLS15	PULLS14	PULLS13	PULLS12	PULLS11	PULLS10	PULLS09	PULLS08	PULLS07	PULLS06	PULLS05	PULLS04	PULLS03	PULLS02	PULLS01	PULLS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	PULLS n	Writing 1 to PULLS n will set PULL n to 1 in register PXPE. Writing 0 to PULLS n will no use.	W

1.2.18 PORT PULL Clear Registers (PxPEC)

PAPEC, PBPEC, PCPEC, PDPEC, PEPEC and PFPEC are six 32-bit PORT PULL clear registers.

PAPES, PBPEC, PCPEC, PDPEC, PEPEC, PFPEC																																0x10010078, 0x10010178, 0x10010278, 0x10010378, 0x10010478, 0x10010578
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PULLC31	PULLC30	PULLC29	PULLC28	PULLC27	PULLC26	PULLC25	PULLC24	PULLC23	PULLC22	PULLC21	PULLC20	PULLC19	PULLC18	PULLC17	PULLC16	PULLC15	PULLC14	PULLC13	PULLC12	PULLC11	PULLC10	PULLC09	PULLC08	PULLC07	PULLC06	PULLC05	PULLC04	PULLC03	PULLC02	PULLC01	PULLC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	PULLC n	Writing 1 to PULLC n will set PULL n to 0 in register PXPE. Writing 0 to PULLC n will no use.	W

1.3 Program Guide

1.3.1 Port Function Guide

INT	MASK	PAT1	PAT0	Port Description
1	0	0	0	Port is low level triggered interrupt input.
1	0	0	1	Port is high level triggered interrupt input.
1	0	1	0	Port is fall edge triggered interrupt input.
1	0	1	1	Port is rise edge triggered interrupt input.
1	1	0	0	Port is low level triggered interrupt input. Interrupt is masked. Flag is recorded.
1	1	0	1	Port is high level triggered interrupt input. Interrupt is masked. Flag is recorded.
1	1	1	0	Port is fall edge triggered interrupt input. Interrupt is masked. Flag is recorded.
1	1	1	1	Port is rise edge triggered interrupt input. Interrupt is masked. Flag is recorded.
0	0	0	0	Port is pin of device 0.
0	0	0	1	Port is pin of device 1.
0	0	1	0	Port is pin of device 2.
0	0	1	1	Port is pin of device 3.
0	1	0	0	Port is GPIO output 0.
0	1	0	1	Port is GPIO output 1.
0	1	1	?	Port is GPIO input.

2 I2C Controller

2.1 Overview

The I2C bus is a two-wire serial interface, consisting of a serial data line (SDA) and a serial clock (SCL). These wires carry information between the devices connected to the bus. Each device is recognized by a unique address and can operate as either a “transmitter” or “receiver,” depending on the function of the device. Devices can also be considered as masters or slaves when performing data transfers. A master is the device that initializes/terminates a data transfer on the bus and generates clock signals to permit that transfer. During that time, any addressed device is considered as a slave. The I2C controller is software controlled. It behaves as a master or a slave. However, operating as a master and slave simultaneously is not supported.

2.1.1 Features

- Two-wire I2C serial interface – consists of a serial data line (SDA) and a serial clock (SCL)
- Two speeds
  - Standard mode (100 Kb/s)
  - Fast mode (400 Kb/s)
- Device clock is identical with pclk
- Programmable SCL generator
- Master or slave I2C operation
- 7-bit addressing/10-bit addressing
- 16-level transmit and receive FIFOs
- Interrupt operation
- The number of devices that you can connect to the same I2C-bus is limited only by the maximum bus capacitance of 400pF
- APB interface
- 3 independent I2C channels (I2C0, I2C1, I2C2)

2.1.2 Pin Description

Table 2-1 I2C Pin Description

Name	Width	IO	Description
SDA	1-bit	IO	I2C serial data
SCL	1-bit	IO	I2C serial clock

Please note that dedicate pull-up resistors must be introduced on board-level. The low-to-high (rise time) transition is highly dependent on RC time constant of the bus.

Totally speaking, for standard-mode I2C-bus system, the pull up resistor depends on following

parameters:

- Supply voltage
- Bus capacitance
- Number of connected devices

For fast-mode I2C-bus system, switched pull-up circuit may be essential for strict speed and load requirement. (Refer to ' Philips Semiconductor, *THE I^2C-BUS SPECIFICATION*, Version 2.1. Jan, 2000')

2.2 Registers

2.2.1 Registers Memory Map

A read operation to an address location that contains unused bits results in a 0 value being returned on each of the unused bits.

Registers in I2C controller can be accessed by indicating 24-bit Address Base combined with 8-bit Address Offset.

Table 2-2 Registers Memory Map-Address Base

Name	Addr Base	Description
I2C0	0x10050000	Address base of I2C0
I2C1	0x10051000	Address base of I2C1
I2C2	0x10055000	Address base of I2C2

Table 2-3 Registers Memory Map-Address Offset

Name	Addr Offset	Description	Width	RW	Reset
I2CCON	0x00	I2C control	8bits	RW	0x6D
I2CTAR	0x04	I2C target address	13bits	RW	0x0855
I2CSAR	0x08	I2C slave address	10bits	RW	0x055
I2CDC	0x10	I2C data buffer and command	9bits	RW	0x000
I2CSHCNT	0x14	Standard speed I2C SCL high count	16bits	RW	0x0190
I2CSLCNT	0x18	Standard speed I2C SCL low count	16bits	RW	0x01D6
I2CFHCNT	0x1C	Fast speed I2C SCL high count	16bits	RW	0x003C
I2CFLCNT	0x20	Fast speed I2C SCL low count	16bits	RW	0x0082
I2CINTST	0x2C	I2C Interrupt Status	12bits	R	0x000
I2CINTM	0x30	I2C Interrupt Mask	12bits	R/W	12'h8FF
I2CRXTL	0x38	I2C RxFIFO Threshold	4 bits	RW	0xF
I2CTXTL	0x3C	I2C TxFIFO Threshold	4 bits	RW	0xF
I2CCINT	0x40	Clear Interrupts	1 bit	R	0x0
I2CCRXUF	0x44	Clear RXUF Interrupt	1 bit	R	0x0
I2CCRXOF	0x48	Clear RX_OVER Interrupt	1 bit	R	0x0
I2CCTXOF	0x4C	Clear TX_OVER Interrupt	1 bit	R	0x0
I2CCRXREQ	0x50	Clear RDREQ Interrupt	1 bit	R	0x0
I2CCTXABT	0x54	Clear TX_ABRT Interrupt	1 bit	R	0x0
I2CCRXDN	0x58	Clear RX_DONE Interrupt	1 bit	R	0x0
I2CCACT	0x5c	Clear ACTIVITY Interrupt	1 bit	R	0x0
I2CCSTP	0x60	Clear STOP Interrupt	1 bit	R	0x0
I2CCSTT	0x64	Clear START Interrupt	1 bit	R	0x0
I2CCGC	0x68	Clear GEN_CALL Interrupt	1 bit	R	0x0
I2CENB	0x6C	I2C Enable	1 bit	RW	0x0

I2CST	0x70	I2C Status register	7 bits	R	0x06
I2CABTSRC	0x80	I2C Transmit Abort Status Register	16 bits	R	0x0000
I2CDMACR	0x88	DMA Control Register	2 bits	R/W	0x0
I2CDMATDLR	0x8c	DMA Transmit Data Level	4 bits	R/W	0x0
I2CDMARDLR	0x90	DMA Receive Data Level	4 bits	R/W	0x0
I2CSDASU	0x94	I2C SDA Setup Register	8 bits	RW	0x64
I2CACKGC	0x98	I2C ACK General Call Register	1 bit	RW	0x1
I2CENBST	0x9C	I2C Enable Status Register	3 bits	R	0x0
I2CSDAHD	0xD0	I2C SDA HolD time Register	9 bits	RW	0x000

2.2.2 Registers and Fields Description

2.2.2.1 I2CCON

I2C control register. Only STPHLD can be written during I2C disable and enable, others only can be written when I2C is disable.

I2CCON **BASE + 0x00**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								STPHLD	SLVDIS	REST	MATP	SATP	SPD		MD
RST	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	1

Bits	Name	Description	RW
15:8	Reserved	Writing has no effect, read as zero.	R
7	STPHLD	Stop Hold Enable bit. 0: When tx fifo is empty, a STP condition will be generated 1: STP will never be generated until this bit changed to 0 Only this bit in I2CCON can be written during I2C disable and enable status.	RW
6	SLVDIS	This bit controls whether I2C has its slave disabled after reset. 0: slave is enabled 1: slave is disabled	RW
5	REST	Determines whether RESTART conditions may be sent when acting as a master.	RW

		0: disable 1: enable	
4	MATP	This bit controls whether the I2C starts its transfers in 7-bit or 10-bit addressing mode when acting as a master. The function of this bit is handled by bit 12 of I2CTAR register. 0: 7-bit addressing 1: 10-bit addressing	R
3	SATP	When acting as a slave, this bit controls whether the I2C responds to 7-bit or 10-bit addresses. 0: 7-bit addressing 1: 10-bit addressing	RW
2:1	SPD	These bits control at which speed the I2C operates. 1: standard mode (100 kbps) 2: fast mode (400 kbps) **NOTE:** when these two bits are set to 2'b00 or 2'b11, the speed mode will be automatically set to 2'b10 i.e. fast mode.	RW
0	MD	This bit controls whether the I2C master is enabled. 0: master disabled 1: master enabled	RW

2.2.2.2 I2CTAR

I2CTAR BASE + 0x04

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved			MATP	SPECIAL	GC_OR_START	I2CTAR									
RST	0	0	0	0	1	0	0	0	0	1	0	1	0	1	0	1

Bits	Name	Description	R/W
15:13	Reserved	Writing has no effect, read as zero.	R
12	MATP	This bit controls whether the I2C starts its transfers in 7- or 10-bit addressing mode when acting as a master. 0: 7-bit addressing 1: 10-bit addressing **NOTE:** this bit is initially set to 0.	RW
11	SPECIAL	This bit indicates whether software performs a General Call or START BYTE command. 0: ignore the bit of GC_OR_START and use I2CTAR normally 1: perform special I2C command as specified in GC_OR_START bit	RW

		NOTE: this bit is initially set to 1.	
10	GC_OR_START	If bit 11 (SPECIAL) is set to 1, then this bit indicates whether a General Call or START byte command is to be performed by the I2C. 0: General Call Address – after issuing a General Call, only writes may be performed. Attempting to issue a read command results in setting bit 6 (TX_ABRT) of the I2CINTST register. The I2C remains in General Call mode until the SPECIAL bit value (bit 11) is cleared 1: START BYTE	RW
9:0	I2CTAR	This is the target address for any master transaction. When transmitting a General Call, these bits are ignored. To generate a START BYTE, the CPU needs to write only once into these bits. If the I2CTAR and I2CSAR are the same, loopback exists but the FIFOs are shared between master and slave, so full loopback is not feasible. Only one direction loopback mode is supported (simplex), not duplex. A master cannot transmit to itself, but only to a slave.	RW

NOTE: It is not necessary to perform any write to this register if I2C is enabled as an I2C slave only.

2.2.2.3 I2CSAR

I2CSAR BASE + 0x08

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved						I2CSAR									
RST	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1

Bits	Name	Description	R/W
15:10	Reserved	Writing has no effect, read as zero.	R
9:0	I2CSAR	The I2CSAR holds the slave address when the I2C is operating as a slave. For 7-bit addressing, only I2CSAR[6:0] is used. This register can be written only when the I2C interface is disabled. Writes at other times have no effect.	RW

NOTE: It is not necessary to perform any write to this register if I2C is enabled as an I2C master only.

2.2.2.4 I2CDC

I2C Rx/Tx Data Buffer and Command Register, which the CPU writes to when filling the TX FIFO and the CPU reads from when retrieving bytes from RX FIFO.

I2CDC BASE + 0x10

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved							CMD	DAT							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
15:9	Reserved	Writing has no effect, read as zero.	R
8	CMD	This bit controls whether a read or a write is performed. This bit does not control the direction when the I2C acts as a slave. It controls only the direction when it acts as a master. 1: Read 0: Write	RW
7:0	DAT	This register contains the data to be transmitted or received on the I2C bus.	RW

NOTE: this command only transfer 8-bit data combined with 1-bit CMD, extra bits on the bus will be eliminated. i.e. only 8-0 bits on the bus are accepted by I2C controller.

2.2.2.5 I2CSHCNT

I2CSHCNT BASE + 0x14

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	I2CSHCNT															
RST	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0

Bits	Name	Description	R/W
15:0	I2CSHCNT	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. The register sets the SCL clock high-period count for standard speed. This register can be written only when the I2C interface is disabled. Writes at other times have no effect. SCL high time of i2c is (I2CSHCNT + 8) i2c_clk periods.	RW

NOTE: Minimum value allowed for the I2CSHCNT registers is 6. If the set value was less than 6, it will be automatically set to 6.

2.2.2.6 I2CSLCNT

I2CSLCNT BASE + 0x18

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	I2CSLCNT															
RST	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	0

Bits	Name	Description	R/W
15:0	I2CSLCNT	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for standard speed. This register can be written only when the I2C interface is disabled. Writes at other times have no effect. SCL low time of i2c is (I2CSLCNT + 1) i2c_clk periods.	RW

NOTE: Minimum value that can be programmed in the I2CSLCNT registers is 8. If the set value was less than 8, it will be automatically set to 8.

2.2.2.7 I2CFHCNT

I2CFHCNT BASE + 0x1C

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	I2CFHCNT															
RST	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0

Bits	Name	Description	R/W
15:0	I2CFHCNT	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for fast speed. This register can be written only when the I2C interface is disabled. Writes at other times have no effect.	RW

NOTE: Minimum value allowed for the I2CSHCNT registers is 6. If the set value was less than 6, it will be automatically set to 6.

2.2.2.8 I2CFLCNT

I2CFLCNT BASE + 0x20

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	I2CFLCNT															
RST	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0

Bits	Name	Description	R/W
15:0	I2CFLCNT	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for fast speed. This register can be written only when the I2C interface is disabled. Writes at other times have no effect.	RW

NOTE: Minimum value that can be programmed in the I2CSLCNT registers is 8. If the set value was less than 8, it will be automatically set to 8.

2.2.2.9 I2CINTST

Each bit in this register has a corresponding mask bit in the I2CINTM register. These bits are cleared by reading the matching interrupt clear register.

I2CINTST BASE + 0x2C

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				IGC	ISTT	ISTP	IACT	RXDN	TXABT	RDREQ	TXEMP	TXOF	RXFL	RXOF	RXUF
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
15:12	Reserved	Writing has no effect, read as zero.	R
11	IGC	Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling I2C or when the CPU reads bit 0 of the I2CCGC register. I2C stores the received data in the Rx buffer.	R
10	ISTT	Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether I2C is operating in slave or master mode.	R
9	ISTP	Indicates whether a STOP condition has occurred on the I2C interface regardless of whether I2C is operating in slave or master mode.	R

8	IACT	This bit captures I2C activity and stays set until it is cleared. There are four ways to clear it: 1 Disabling the I2C 2 Reading the I2CCACT register 3 Reading the I2CCINT register 4 System reset Once this bit is set, it stays set unless one of the four methods is used to clear it. Even if the I2C module is idle, this bit remains set until cleared, indicating that there was activity on the bus.	R
7	RXDN	When the I2C is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done.	R
6	TXABT	This bit indicates if I2C, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or an I2C slave, and is referred to as a “transmit abort”. When this bit is set to 1, the I2CABTSRC register indicates the reason why the transmit abort takes places. **NOTE:** The I2C flushes/resets/empties the TX FIFO whenever this bit is set. The TX FIFO remains in this flushed state until the register I2CCTXABT is read. Once this read is performed, the TX FIFO is then ready to accept more data bytes from the APB interface.	R
5	RDREQ	This bit is set to 1 when I2C is acting as a slave and another I2C master is attempting to read data from I2C. The I2C holds the I2C bus in a wait state (SCL=0) until this interrupt is serviced, which means that the slave has been addressed by a remote master that is asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the I2CDC register. This bit is set to 0 just after the processor reads the I2CCRREQregister.	R
4	TXEMP	This bit is set to 1 when the transmit buffer is at or below the threshold value set in the I2C_TXTL register. It is automatically cleared by hardware when the buffer level goes above the threshold. When the I2CENB bit 0 is 0, the TX FIFO is flushed and held in reset. There the TX FIFO looks like it has no data within it, so this bit is set to 1, provided there is activity in the master or slave state machines. When there is no longer activity, then with I2CENB = 0, this bit is set to 0.	R
3	TXOF	Set during transmit if the transmit buffer is filled to I2CTX_BUFFER_DEPTH and the processor attempts to issue another I2C command by writing to the I2CDC register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, and when i2c_en goes to 0, this interrupt is cleared.	R
2	RXFL	Set when the receive buffer reaches or goes above the I2C_RXTL threshold in the I2C_RXTL register. It is automatically cleared by	R

		hardware when buffer level goes below the threshold. It is automatically cleared by hardware when buffer level goes below the threshold. If the module is disabled (I2CENB[0]=0), the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full. So this bit is cleared once the I2CENB bit 0 is programmed with a 0, regardless of the activity that continues.	
1	RXOF	Set if the receive buffer is completely filled to 2 and an additional byte is received from an external I2C device. The I2C acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled (I2CENB[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when i2c_en goes to 0, this interrupt is cleared.	R
0	RXUF	Set if the processor attempts to read the receive buffer when it is empty by reading from the I2CDC register. If the module is disabled (I2CENB[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when i2c_en goes to 0, this interrupt is cleared.	R

2.2.2.10 I2CINTM

These bits mask their corresponding interrupt status bits. They are active low; a value of 0 prevents a bit from generating an interrupt.

I2CINTM **BASE + 0x30**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				MIGC	MISTT	MISTP	MIACT	MRXDN	MTXABT	MRDREQ	MTXEMP	MTXOF	MRXFL	MRXOF	MRXUF
RST	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1

Bits	Name	Description	R/W
15:12	Reserved	Writing has no effect, read as zero.	R
11	MIGC	These bits mask their corresponding interrupt status bits in the I2CINTST register.	RW
10	MISTT		RW
9	MISTP		RW
8	MIACT		RW
7	MRXDN		RW
6	MTXABT		RW
5	MRDREQ		RW
4	MTXEMP		RW
3	MTXOF		RW

2	MRXFL		RW
1	MRXOF		RW
0	MRXUF		RW

2.2.2.11 I2CRXTL

I2CRXTL BASE + 0x38

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved												RXTL			
RST	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Bits	Name	Description	R/W
15:5	Reserved	Writing has no effect, read as zero.	R
3:0	RXTL	Receive FIFO Threshold Level. Controls the level of entries that triggers the RxFIFO full interrupt. A value of n sets the threshold for (n+1) entries.	RW

2.2.2.12 I2CTXTL

I2CTXTL BASE + 0x3C

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved												TXTL			
RST	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Bits	Name	Description	R/W
15:5	Reserved	Writing has no effect, read as zero.	R
3:0	TXTL	Transmit FIFO Threshold Level. Controls the level of entries that trigger the TxFIFO empty interrupt. A value of n sets the threshold for n entries.	RW

2.2.2.13 I2CCINT

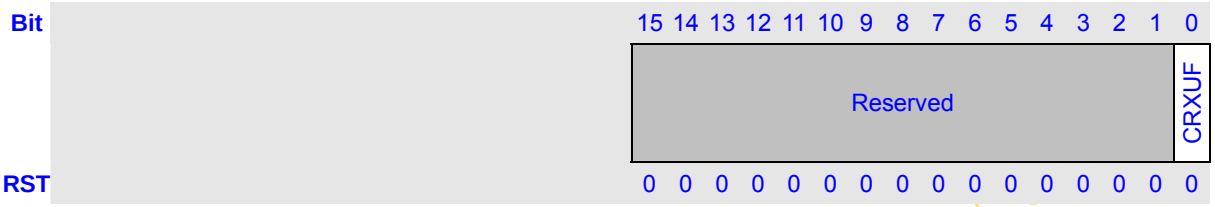
I2CCINT BASE + 0x40

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved															CINT
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
15:1	Reserved	Writing has no effect, read as zero.	R
0	CINT	Read this register to clear the combined interrupt, all individual interrupts, and the I2CABTSRC register. This bit does not clear hardware clearable interrupts but software clearable interrupts. Refer to Bit 9 of the I2CABTSRC register for an exception to clearing I2CABTSRC.	R

2.2.2.14 I2CCRXUF

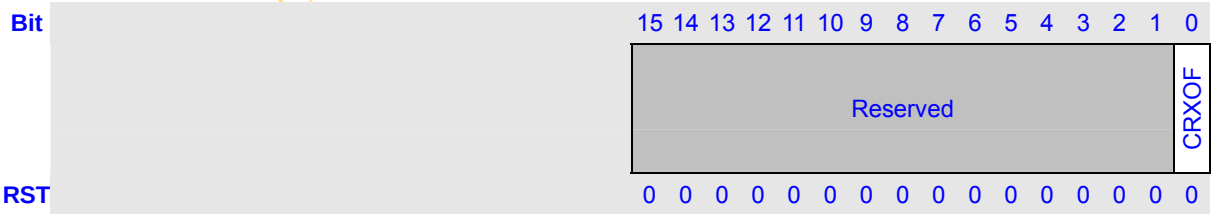
I2CCRXUF BASE + 0x44



Bits	Name	Description	R/W
15:1	Reserved	Writing has no effect, read as zero.	R
0	CRXUF	Read this register to clear the RXUF interrupt (bit 0) of the I2CINTST register.	R

2.2.2.15 I2CCRXOF

I2CCRXOF BASE + 0x48



Bits	Name	Description	R/W
15:1	Reserved	Writing has no effect, read as zero.	R
0	CRXOF	Read this register to clear the RXOF interrupt (bit 1) of the I2CINTST register.	R

2.2.2.16 I2CCTXOF

I2CCTXOF BASE + 0x4C

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved															CTXOF
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
15:1	Reserved	Writing has no effect, read as zero.	R
0	CTXOF	Read this register to clear the TX_OVER interrupt (bit 3) of the I2CINTST register.	R

2.2.2.17 I2CCRXREQ

I2CCRXREQ BASE + 0x50

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved															CLRDREQ
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
15:1	Reserved	Writing has no effect, read as zero.	R
0	CLRDREQ	Read this register to clear the RDREQ interrupt (bit 5) of the I2CINTST register.	R

2.2.2.18 I2CCTXABT

I2CCTXABT BASE + 0x54

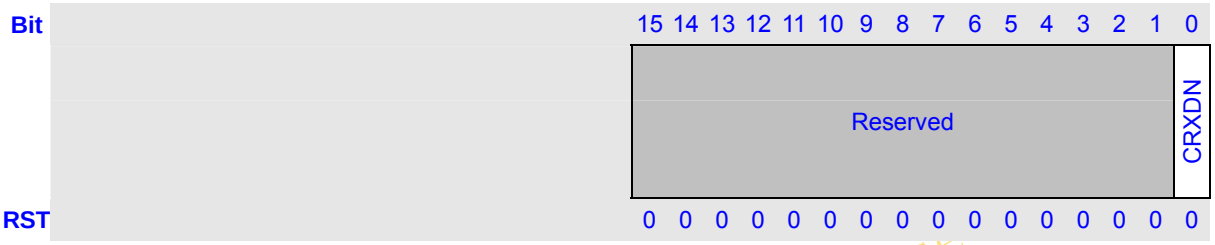
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved															CTXABT
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
15:1	Reserved	Writing has no effect, read as zero.	R

0	CTXABT	Read this register to clear the TX_ABRT interrupt (bit 6) of the I2CINTST register, and the I2CABTSRC register. This also releases the TX FIFO from the flushed/reset state, allowing more writes to the TX FIFO. Refer to Bit 9 of the I2CABTSRC register for an exception to clearing I2CABTSRC.	R

2.2.2.19 I2CCRXDN

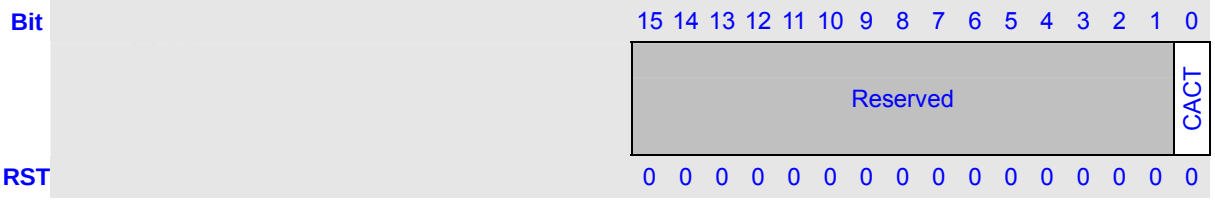
I2CCRXDN BASE + 0x58



Bits	Name	Description	R/W
15:1	Reserved	Writing has no effect, read as zero.	R
0	CRXDN	Read this register to clear the RX_DONE interrupt (bit 7) of the I2CINTST register.	R

2.2.2.20 I2CCACT

I2CCACT BASE + 0x5C



Bits	Name	Description	R/W
15:1	Reserved	Writing has no effect, read as zero.	R
0	CACT	Reading this register clears the ACTIVITY interrupt if the I2C is not active anymore. If the I2C module is still active on the bus, the ACTIVITY interrupt bit continues to be set. It is automatically cleared by hardware if the module is disabled and if there is no further activity on the bus. The value read from this register to get status of the ACTIVITY interrupt (bit 8) of the I2CINTST register.	R

2.2.2.21 I2CCSTP

I2CCSTP BASE + 0x60

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved															CSTP
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
15:1	Reserved	Writing has no effect, read as zero.	R
0	CSTP	Read this register to clear the STOP interrupt (bit 9) of the I2CINTST register.	R

2.2.2.22 I2CCSTT

I2CCSTT BASE + 0x64

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved															CSTT
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
15:1	Reserved	Writing has no effect, read as zero.	R
0	CSTT	Read this register to clear the START interrupt (bit 10) of the I2CINTST register.	R

2.2.2.23 I2CCGC

I2CCGC BASE + 0x68

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved															CGC
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
15:1	Reserved	Writing has no effect, read as zero.	R
0	CGC	Read this register to clear the GEN_CALL interrupt (bit 11) of I2CINTST register.	R

2.2.2.24 I2CENB

I2CENB BASE + 0x6C

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved															I2CENB
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
15:1	Reserved	Writing has no effect, read as zero.	R
0	I2CENB	Controls whether the I2C is enabled. 0: Disables I2C (TX and RX FIFOs are held in an erased state) 1: Enables I2C Software can disable I2C while it is active. However, it is important that care be taken to ensure that I2C is disabled properly. When I2C is disabled, the following occurs: – The TX FIFO and RX FIFO get flushed. – Status bits in the I2CINTST register are still active until I2C goes into IDLE state. If the module is transmitting, it stops as well as deletes the contents of the transmit buffer after the current transfer is complete. If the module is receiving, the I2C stops the current transfer at the end of the current byte and does not acknowledge the transfer.	R

2.2.2.25 I2CST

This is a read-only register used to indicate the current transfer status and FIFO status. The status register may be read at any time. None of the bits in this register request an interrupt.

When the I2C is disabled by writing 0 in bit 0 of the I2CENB register:

– Bits 1 and 2 are set to 1

– Bits 3 and 4 are set to 0

When the master or slave state machine goes to idle and ic_en=0:

– Bits 5 and 6 are set to 0

I2CST BASE + 0x70

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved									SLVACT	MSTACT	RFF	RFNE	TFE	TFNF	ACT
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Bits	Name	Description	R/W
15:7	Reserved	Writing has no effect, read as zero.	R
6	SLVACT	Slave FSM Activity Status. 0: Slave FSM is in IDLE state 1: Slave FSM is not in IDLE state	R
5	MSTACT	Master FSM Activity Status. 0: Master FSM is in IDLE state 1: Master FSM is not in IDLE state	R
4	RFF	Receive FIFO Completely Full. When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared. 0: Receive FIFO is not full 1: Receive FIFO is full	R
3	RFNE	Receive FIFO Not Empty. This bit is set when the receive FIFO contains one or more entries; it is cleared when the receive FIFO is empty. 0: Receive FIFO is empty 1: Receive FIFO is not empty	R
2	TFE	Transmit FIFO Completely Empty. When the transmit FIFO is completely empty, this bit is set. When it contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt. 0: Transmit FIFO is not empty 1: Transmit FIFO is empty	R
1	TFNF	Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full. 0: Transmit FIFO is full 1: Transmit FIFO is not full	R
0	ACT	I2C Activity Status. The OR of SLVACT and MSTACT bits.	R

2.2.2.26 I2CABTSRC

This register has 16 bits that indicate the source of the TX_ABRT bit. Except for Bit 9, this register is cleared whenever the I2CCTXABT register or the I2CCINT register is read. To clear Bit 9, the source of the SBYTE_NORSTRT must be fixed first; RESTART must be enabled (I2CCON[5]=1), the SPECIAL bit must be cleared (I2CTAR[11]), or the GC_OR_START bit must be cleared (I2CTAR[10]). Once the source of the SBYTE_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the SBYTE_NORSTRT is not fixed before attempting to clear this bit, Bit 9 clears for one cycle and is then re-asserted.

I2CABTSRC **BASE + 0x80**

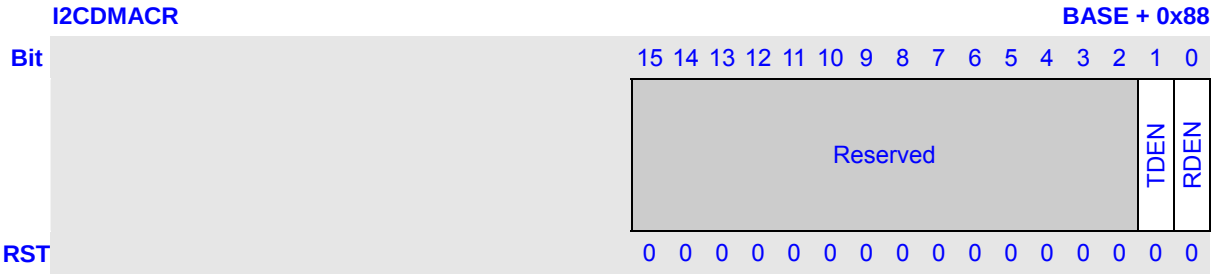
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SLVRD_INTX	SLV_ARBLOST	SLVFLUSH_TXFIFO	ARB_LOST	ABRT_MASTER_DIS	ABRT_10B_RD_NORSTRT	SBYTE_NORSTRT	ABRT_HS_NORSTRT	SBYTE_ACKDET	ABRT_HS_ACKDET	ABRT_GCALL_READ	ABRT_GCALL_N	ABRT_TXDATA_N OACK	ABRT_10ADDR2_NOACK	ABRT_10ADDR1_NOACK	ABRT_7B_ADDR_NOACK
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
15	SLVRD_INTX	1: When the processor side responds to a slave mode request for data to be transmitted to a remote master and user writes a 1 in CMD (bit 8) of I2CDC register.	R
14	SLV_ARBLOST	1: Slave lost the bus while transmitting data to a remote master. I2CABTSRC[12] is set at the same time. **NOTE:** Even though the slave never “owns” the bus, something could go wrong on the bus. This is a fail safe check. For instance, during a data transmission at the low-to-high transition of SCL, if what is on the data bus is not what is supposed to be transmitted, then I2C no longer own the bus. Reset value: 0x0.	R
13	SLVFLUSH_TXFIFO	1: Slave has received a read command and some data exists in the TX FIFO so the slave issues a TX_ABRT interrupt to flush old data in TX FIFO. Reset value: 0x0.	R
12	ARB_LOST	1: Master has lost arbitration, or if I2CABTSRC[14] is also set, then the slave transmitter has lost arbitration. **NOTE:** I2C can be both master and slave at the same time. Reset value: 0x0.	R
11	ABRT_MASTER_DIS	1: User tries to initiate a Master operation with the Master mode disabled. Reset value: 0x0.	R
10	ABRT_10B_RD_NORSTRT	1: The restart is disabled (I2CRESTART_EN bit (I2CCON[5]) = 0) and the master sends a read command in 10-bit addressing mode. Reset value: 0x0.	R
9	SBYTE_NORSTRT	To clear Bit 9, the source of the SBYTE_NORSTRT must be fixed first; restart must be enabled (I2CCON[5]=1), the SPECIAL bit must be cleared (I2CTAR[11]), or the GC_OR_START bit must be	R

		cleared (I2CTAR[10]). Once the source of the SBYTE_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the SBYTE_NORSTRT is not fixed before attempting to clear this bit, bit 9 clears for one cycle and then gets re-asserted. 1: The restart is disabled (I2CRESTART_EN bit (I2CCON[5]) = 0) and the user is trying to send a START Byte. Reset value: 0x0.	
8	ABRT_HS_NORSTRT	1: The restart is disabled (I2CRESTART_EN bit (I2CCON[5]) = 0) and the user is trying to use the master to transfer data in High Speed mode. Reset value: 0x0.	R
7	SBYTE_ACKDET	1: Master has sent a START Byte and the START Byte was acknowledged (wrong behavior). Reset value: 0x0.	R
6	ABRT_HS_ACKDET	1: Master is in High Speed mode and the High Speed Master code was acknowledged (wrong behavior). Reset value: 0x0.	R
5	ABRT_GCALL_READ	1: I2C in master mode sent a General Call but the user programmed the byte following the General Call to be a read from the bus (I2CDC[9] is set to 1). Reset value: 0x0.	R
4	ABRT_GCALL_NOACK	1: I2C in master mode sent a General Call and no slave on the bus acknowledged the General Call. Reset value: 0x0.	R
3	ABRT_TXDATA_NOACK	1: This is a master-mode only bit. Master has received an acknowledgement for the address, but when it sent data byte(s) following the address, it did not receive an acknowledgement from the remote slave(s). Reset value: 0x0.	R
2	ABRT_10ADDR2_NOACK	1: Master is in 10-bit address mode and the second address byte of the 10-bit address was not acknowledged by any slave. Reset value: 0x0.	R
1	ABRT_10ADDR1_NOACK	1: Master is in 10-bit address mode and the first 10-bit address byte was not acknowledged by any slave. Reset value: 0x0.	R
0	ABRT_7B_ADDR_NOACK	1: Master is in 7-bit addressing mode and the address sent was not acknowledged by any slave. Reset value: 0x0.	R

2.2.2.27 I2CDMACR

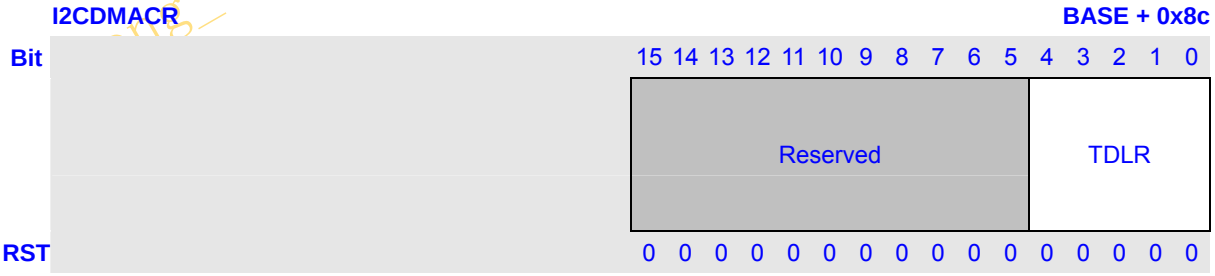
The register is used to enable the DMA Controller interface operation. There is a separate bit for transmit and receive. This can be programmed regardless of the state of i2c enable.



Bits	Name	Description	R/W
15:2	Reserved	Writing has no effect, read as zero.	R
1	TDEN	**Transmit DMA Enable**. This bit enables/disables the transmit DMA channel. 0: Transmit DMA disabled 1: Transmit DMA enabled	R/W
0	RDEN	**Receive DMA Enable**. This bit enables/disables the receive DMA channel. 0: Receive DMA disabled 1: Receive DMA enabled	R/W

2.2.2.28 I2CDMATDLR

The register is used to config dma transmit data level.



Bits	Name	Description	R/W
15:5	Reserved	Writing has no effect, read as zero.	R
4:0	TDLR	**DMA Transmit Data Level**. This bit field controls the level at which a DMA request is made by the transmit logic. The dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value.	R/W

2.2.2.29 I2CDMARDLR

The register is used to config dma receive data level.

I2CDMACR BASE + 0x90

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved											RDLR				
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
15:5	Reserved	Writing has no effect, read as zero.	R
4:0	RDLR	**DMA Receive Data Level**. This bit field controls the level at which a DMA request is made by the receive logic. The dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or more than this field value + 1.	R/W

2.2.2.30 I2CSDASU

This register controls the amount of time delay (in terms of number of i2c_clk clock periods) introduced in the rising edge of SCL, relative to SDA changing, when I2C services a read request in a slave-transmitter operation. The relevant I2C requirement is tSU:DAT (NOTE 2) as detailed in the I2C Bus Specification.

NOTE: The length of setup time is calculated using [(I2CSDASU - 1) * (ic_clk_period)], so if the user requires 10 ic_clk periods of setup time, they should program a value of 11.

A Fast-mode I2C-bus device can be used in a Standard-mode I2C-bus system, but the requirement $t_{SU;DAT}$ >= 250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line $t_{r\ max} + t_{SU;DAT} = 1000 + 250 = 1250$ ns (according to the Standard-mode I2C-bus specification) before the SCL line is released. (Refer to ' Philips Semiconductor, *THE I²C-BUS SPECIFICATION*, Version 2.1. Jan, 2000')

I2CSDASU BASE + 0x94

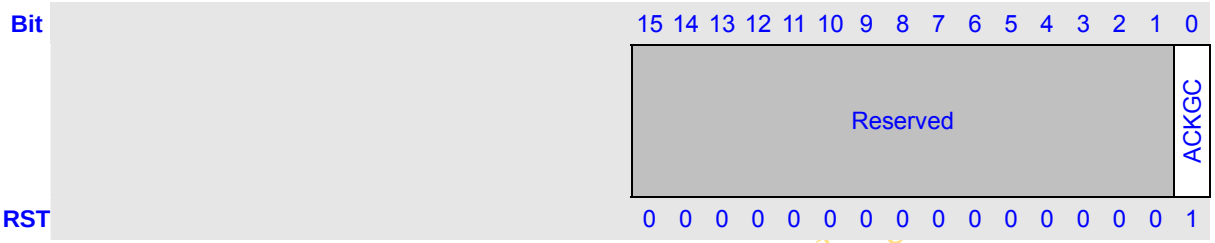
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								SDASU							
RST	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

Bits	Name	Description	R/W
15:8	Reserved	Writing has no effect, read as zero.	R
7:0	SDASU	SDA Setup. It is recommended that if the required delay is 1000ns, then for an i2c_clk frequency of 10 MHz, I2CSDASU should be programmed to a value of 11.	R/W

2.2.2.31 I2CACKGC

The register controls whether I2C responds with an ACK or NACK when it receives an I2C General Call address.

I2CACKGC BASE + 0x98



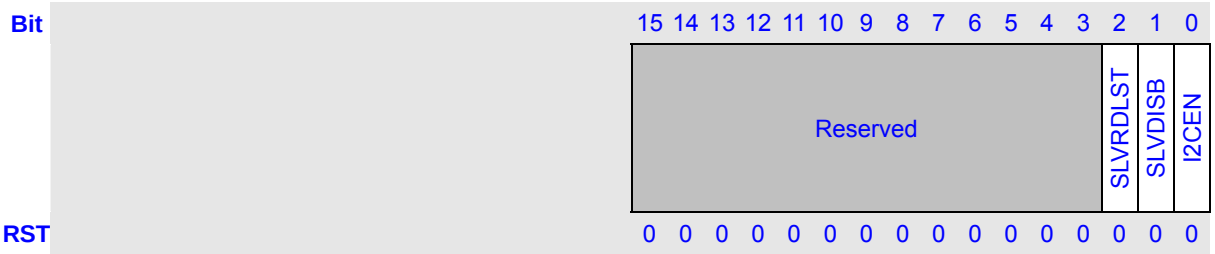
Bits	Name	Description	R/W
15:1	Reserved	Writing has no effect, read as zero.	R
0	ACKGC	ACK General Call. When set to 1, I2C responds with an ACK (by asserting ic_data_oe) when it receives a General Call. When set to 0, the I2C does not generate General Call interrupts.	R/W

2.2.2.32 I2CENBST

The register is used to report the I2C hardware status when the I2CENB register is set from 1 to 0; that is, when I2C is disabled.

If I2CENB has been set to 1, bits 2:1 are forced to 0, and bit 0 is forced to 1.
If I2CENB has been set to 0, bits 2:1 is only be valid as soon as bit 0 is read as '0'.

I2CENBST BASE + 0x9C



Bits	Name	Description	R/W
15:3	Reserved	Writing has no effect, read as zero.	R

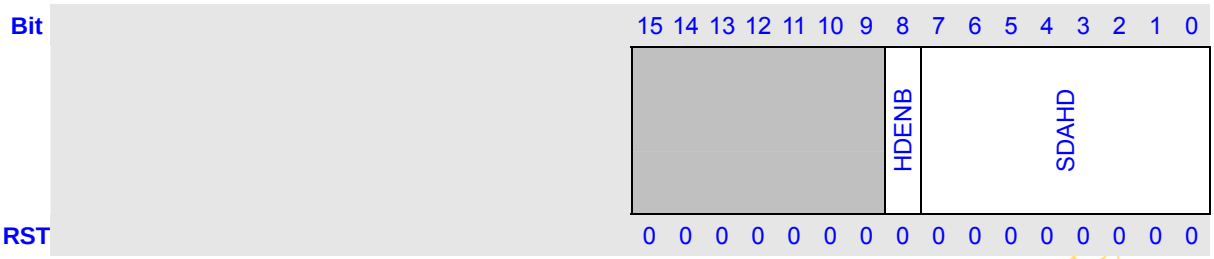
2	SLVRDLST	Slave Received Data Lost. This bit indicates if a Slave-Receiver operation has been aborted with at least one data byte received from an I2C transfer due to the setting of I2CENB from 1 to 0. When read as 1, I2C is deemed to have been actively engaged in an aborted I2C transfer (with matching address) and the data phase of the I2C transfer has been entered, even though a data byte has been responded with a NACK. **NOTE:** If the remote I2C master terminates the transfer with a STOP condition before the I2C has a chance to NACK a transfer, and I2CENB has been set to 0, then this bit is also set to 1. When read as 0, I2C is deemed to have been disabled without being actively involved in the data phase of a Slave-Receiver transfer. **NOTE:** The CPU can safely read this bit when IC_EN (bit 0) is read as 0. Reset value: 0x0.	R
1	SLVDISB	Slave Disabled While Busy (Transmit, Receive). This bit indicates if a potential or active Slave operation has been aborted due to the setting of the I2CENB register from 1 to 0. This bit is set when the CPU writes a 0 to the I2CENB register while: (a) I2C is receiving the address byte of the Slave-Transmitter operation from a remote master; OR, (b) address and data bytes of the Slave-Receiver operation from a remote master. When read as 1, I2C is deemed to have forced a NACK during any part of an I2C transfer, irrespective of whether the I2C address matches the slave address set in I2C (I2CSAR register) OR if the transfer is completed before I2CENB is set to 0 but has not taken effect. **NOTES:** 1 If the remote I2C master terminates the transfer with a STOP condition before the I2C has a chance to NACK a transfer, and I2CENB has been set to 0, then this bit will also be set to 1. When read as 0, I2C is deemed to have been disabled when there is master activity, or when the I2C bus is idle. 2 The CPU can safely read this bit when IC_EN (bit 0) is read as 0. Reset value: 0x0.	R
0	I2CEN	ic_en Status. This bit always reflects the value driven on the output port ic_en. When read as 1, I2C is deemed to be in an enabled state. When read as 0, I2C is deemed completely inactive. **NOTE:** The CPU can safely read this bit anytime. When this bit is read as 0, the CPU can safely read SLVRDLST (bit 2) and SLVDISB (bit 1). Reset value: 0x0.	R

2.2.2.33 I2CSDAHD

This register controls the amount of delay (in terms of i2c_clk cycles) that introduced in the falling edge of SCL, relative to SDA changing. It works when I2C services a read request in master-transmitter operation.

The length of hold time is calculated using: T_{delay} = (I2CSDAHD + 1) * (i2c_clk perild).

I2CSDAHD **BASE + 0xD0**



Bits	Name	Description	R/W
15:9	Reserved	Reserved.	N/A
8	HDENB	I2C Hold Enable. 0: The actual hold time is 1 * (i2c_clk period). 1: The actual hold time is (SDAHD + 1) * (i2c_clk period).	RW
7:0	SDAHD	I2C Hold Time. In terms of number of i2c_clk clock periods	RW

2.3 Operating Flow

This section provides information on the following topics:

- "Slave Mode Operation"
- "Master Mode Operation"
- "Disabling I2C"

NOTE: It is important to note that the I2C should only be set to operate as an I2C Master, or I2C Slave, but not both simultaneously. This is achieved by ensuring that bit 6 (I2CSLAVE_DISABLE) and 0 (I2CMASTER_MODE) of the I2CCON register are never set to 0 and 1, respectively.

2.3.1 I2C Behavior

The I2C can be controlled via software to be either:

- An I2C master only, communicating with other I2C slaves.

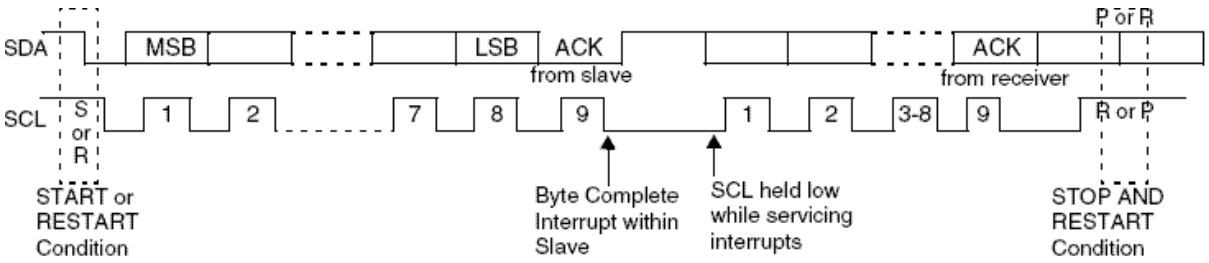
OR

- An I2C slave only, communicating with one more I2C masters.

The master is responsible for generating the clock and controlling the transfer of data. The slave is responsible for either transmitting or receiving data to/from the master. The device that is receiving data, which can be either a master or a slave, sends the acknowledgement of data. As mentioned previously, the I2C protocol also allows multiple masters to reside on the I2C bus and uses an arbitration procedure to determine bus ownership.

Each slave has a unique address that is determined by the system designer. When a master wants to communicate with a slave, the master transmits a START/RESTART condition that is then followed by the slave's address and a control bit (R/W) to determine if the master wants to transmit data or receive data from the slave. The slave then sends an acknowledgement (ACK) pulse after the address.

If the master (master-transmitter) is writing to the slave (slave-receiver), the receiver gets one byte of data. This transaction continues until the master terminates the transmission with a STOP condition. If the master is reading from a slave (master-receiver), the slave transmits (slave-transmitter) a byte of data to the master, and the master then acknowledges the transaction with the ACK pulse. This transaction continues until the master terminates the transmission by not acknowledging (NACK) the transaction after the last byte is received, and then the master issues a STOP condition or addresses another slave after issuing a RESTART condition.



2.3.2 Master Mode Operation

This section includes the following topics:

- "Initial Configuration"
- "Dynamic I2CTAR or I2C10BITADDR_MASTER Update"
- "Master Transmit and Master Receive"

2.3.2.1 Configuration

To use the I2C as a master perform the following steps:

1 Disable the I2C by writing 0 to the I2CENB register. And wait for the I2CENBST.I2CEN = 0.

2 Write to the I2CCON register to set the speed mode supported (bits 2:1). Please note that the MATP (bit4) is NOT writable. The addressing mode is controlled by Register I2CTAR.

3 Set the expected SCL frequency. I2CCON.SPD = 2'b01, only I2CSHCNT and I2CSLCNT are needed to be configured; I2CCON.SPD = 2'b10, only I2CFHCNT and I2CFLCNT are needed to be configured.

Supposed:

T_{scl} : I2C SCL period

T_{i2c_clk} : I2C device clock period

$T_{min\_scl\_l}$: Protocol minimum SCL low time

$T_{min\_scl\_h}$: Protocol minimum SCL high time

Then can get the equation:

$T_{scl} = T_{i2c\_clk} * ((I2C*HCNT + 8) + (I2C*LCNT + 1))$

And the following conditions should be met:

$(I2C*HCNT + 8) * T_{i2c\_clk} >= T_{min\_scl\_h}$

$(I2C*LCNT + 1) * T_{i2c\_clk} >= T_{min\_scl\_l}$

I2C*HCNT >= 6

I2C*LCNT >= 8

4 Write to the I2CTAR register the address of the I2C device to be addressed. It also indicates whether a General Call or a START BYTE command is going to be performed by I2C. The addressing mode that master starts, either 7-bit or 10-bit addressing, is controlled by the I2C10BITADDR_MASTER bit field (bit 12).

5 Enable the I2C by writing a 1 into I2CENB register. And wait for the I2CENBST.I2CEN = 1.

6 Now write the transfer direction and data to be sent to the I2CDC register (This can be done by DMA). If the I2CDC register is written before the I2C is enabled, the data and commands are lost as the buffers are kept cleared when I2C is not enabled.

NOTE: For multiple I2C transfers, perform additional writes to the TX FIFO such that the TX FIFO does not become empty during the I2C transaction. If the TX FIFO is completely emptied at any stage and I2CCON.STPHLD is 0, then further writes to the TX FIFO results in an independent I2C transaction.

2.3.2.2 Dynamic I2CTAR or I2C10BITADDR_MASTER Update

The I2C supports dynamic updating of the I2CTAR (bits 9:0) and I2C10BITADDR_MASTER (bit 12) bit fields of the I2CTAR register. You can dynamically write to the I2CTAR register provided the following conditions are met:

1 I2C is not enabled (I2CENB=0);

OR

2 I2C is enabled (I2CENB=1);
AND
I2C is NOT engaged in any Master (tx, rx) operation (I2CST[5]=0);
AND
I2C is enabled to operate in Master mode (I2CCON[0]=1);
AND
there are NO entries in the TX FIFO (I2CST[2]=0).

2.3.2.3 Master Transmit and Master Receive

The I2C supports switching back and forth between reading and writing dynamically. To transmit data, write the data to be transferred to lower byte of the I2C Rx/Tx Data Buffer and Command Register (I2CDC). The *CMD* bit (I2CDC[8]) should be written to 0 for I2C write operations.

Subsequently, a read command may be issued by writing "don't cares" to the lower byte of the I2CDC register, and a 1 should be written to the *CMD* bit.

2.3.3 Slave Mode Operation

This section includes the following procedures:

- "Initial Configuration"
- "Slave-Transmitter Operation for a Single Byte"
- "Slave-Receiver Operation for a Single Byte"
- "Slave-Transfer Operation For Bulk Transfers"

2.3.3.1 Initial Configuration

To use the I2C as a slave, perform the following steps:

1 Disable the I2C by writing a '0' to bit 0 of the I2CENB register.
2 Write to the I2CSAR register (bits 9:0) to set the slave address. This is the address to which the I2C responds.
3 Write to the I2CCON register to specify which type of addressing is supported (7- or 10-bit by setting bit 3). Enable the I2C in slave-only mode by writing a '0' into bit 6 (I2CSLAVE_DISABLE) and a '0' to bit 0 (MASTER_MODE).

NOTE: Slaves and masters do not have to be programmed with the same type of addressing 7- or 10-bit address. For instance, a slave can be programmed with 7-bit addressing

and a master with 10-bit addressing, and vice versa.

4 Enable the I2C by writing a '1' in bit 0 of the I2CENB register.

NOTE: Depending on the reset values chosen, steps 2 and 3 may not be necessary because the reset values can be configured. For instance, if the device is only going to be a master, there would be no need to set the slave address because you can configure I2C to have the slave disabled after reset and to enable the master after reset. The values stored are static and do not need to be reprogrammed if the I2C is disabled.

2.3.3.2 Slave-Transmitter Operation for a Single Byte

When another I2C master device on the bus addresses the I2C and requests data, the I2C acts as a slave-transmitter and the following steps occur:

1 The other I2C master device initiates an I2C transfer with an address that matches the slave address in the I2CSAR register of the I2C.
2 The I2C acknowledges the sent address and recognizes the direction of the transfer to indicate that it is acting as a slave-transmitter.
3 The I2C asserts the RDREQ interrupt (bit 5 of the I2CINTST register) and holds the SCL line low. It is in a wait state until software responds.
If the RDREQ interrupt has been masked, due to I2CINTM[5] register (MRDREQ bit field) being set to 0, then it is recommended that a hardware and/or software timing routine be used to instruct the CPU to perform periodic reads of the I2CINTST register.
   a Reads that indicate I2CINTST[5] (RDREQ bit field) being set to 1 must be treated as the equivalent of the RDREQ interrupt being asserted.
   b Software must then act to satisfy the I2C transfer.
   c The timing interval used should be in the order of 10 times the fastest SCL clock period the I2C can handle. For example, for 400 kb/s, the timing interval is 25us.

NOTE: The value of 10 is recommended here because this is approximately the amount of time required for a single byte of data transferred on the I2C bus.

4 If there is any data remaining in the TX FIFO before receiving the read request, then the I2C asserts a TX_ABRT interrupt (bit 6 of the I2CINTST register) to flush the old data from the TX FIFO.

NOTE: Because the I2C's TX FIFO is forced into a flushed/reset state whenever a TX_ABRT event occurs, it is necessary for software to release the I2C from this state by reading the I2CCTXABT register before attempting to write into the TX FIFO. See register I2CINTST for more details. If the TX_ABRT interrupt has been masked, due to of I2CINTM[6] register (MTX_ABRT bit field) being set to 0, then it is recommended that re-using the timing routine (described in the previous step), or a similar one, be used to

read the I2CINTST register.

a Reads that indicate bit 6 (TXABT) being set to 1 must be treated as the equivalent of the TX_ABRT interrupt being asserted.
b There is no further action required from software.
c The timing interval used should be similar to that described in the previous step for the I2CINTST[5] register.

5 Software writes to the I2CDC register with the data to be written (by writing a '0' in bit 8).
6 Software must clear the RDREQ and TX_ABRT interrupts (bits 5 and 6, respectively) of the I2CINTST register before proceeding.
If the RDREQ and/or TX_ABRT interrupts have been masked, then clearing of the I2CINTST register will have already been performed when either the RDREQ or TXABT bit has been read as 1.
7 The I2C releases the SCL and transmits the byte.
8 The master may hold the I2C bus by issuing a RESTART condition or release the bus by issuing a STOP condition.

2.3.3.3 Slave-Receiver Operation for a Single Byte

When another I2C master device on the bus addresses the I2C and is sending data, the I2C acts as a slave-receiver and the following steps occur:

1 The other I2C master device initiates an I2C transfer with an address that matches the I2C's slave address in the I2CSAR register.
2 The I2C acknowledges the sent address and recognizes the direction of the transfer to indicate that the I2C is acting as a slave-receiver.
3 I2C receives the transmitted byte and places it in the receive buffer.

NOTE: If the RX FIFO is completely filled with data when a byte is pushed, then an overflow occurs and the I2C continues with subsequent I2C transfers. Because a NACK is not generated, software must recognize the overflow when indicated by the I2C (by the RXOF bit in the I2CINTST register) and take appropriate actions to recover from lost data. Hence, there is a real time constraint on software to service the RX FIFO before the latter overflow as there is no way to re-apply pressure to the remote transmitting master. You must select a deep enough RX FIFO depth to satisfy the interrupt service interval of their system.

4 I2C asserts the RX_FULL interrupt (I2CINTST[2] register).
If the RX_FULL interrupt has been masked, due to setting I2CINTM[2] register to 0 or setting I2CTX_TL to a value larger than 0, then it is recommended that a timing routine (described in "Slave-Transmitter Operation for a Single Byte" on page 57) be implemented for periodic reads of the "I2CST" on page 136 register. Reads of the I2CST register, with bit 3 (RFNE) set at 1, must then be treated by software as the equivalent of the RX_FULL interrupt being asserted.
5 Software may read the byte from the I2CDC register (bits 7:0).

6 The other master device may hold the I2C bus by issuing a RESTART condition or release the bus by issuing a STOP condition.

2.3.3.4 Slave-Transfer Operation For Bulk Transfers

In the standard I2C protocol, all transactions are single byte transactions and the programmer responds to a remote master read request by writing one byte into the slave's TX FIFO.

When a slave (slave-transmitter) is issued with a read request (RDREQ) from the remote master (master-receiver), at a minimum there should be at least one entry placed into the slave-transmitter's TX FIFO.

I2C is designed to handle more data in the TX FIFO so that subsequent read requests can take that data without raising an interrupt to get more data. Ultimately, this eliminates the possibility of significant latencies being incurred between raising the interrupt for data each time had there been a restriction of having only one entry placed in the TX FIFO.

This mode only occurs when I2C is acting as a slave-transmitter. If the remote master acknowledges the data sent by the slave-transmitter and there is no data in the slave's TX FIFO, the I2C holds the I2C SCL line low while it raises the read request interrupt (RDREQ) and waits for data to be written into the TX FIFO before it can be sent to the remote master.

If the RDREQ interrupt is masked, due to bit 5 (MRDREQ) of the I2CINTST register being set to 0, then it is recommended that a timing routine be used to activate periodic reads of the I2CINTST register. Reads of I2CINTST that return bit 5 (RDREQ) set to 1 must be treated as the equivalent of the RDREQ interrupt referred to in this section.

The RDREQ interrupt is raised upon a read request, and like interrupts, must be cleared when exiting the interrupt service handling routine (ISR). The ISR allows you to either write 1 byte or more than 1 byte into the TX FIFO. During the transmission of these bytes to the master, if the master acknowledges the last byte. Then the slave must raise the RDREQ again because the master is requesting for more data.

2.3.4 Disabling I2C

The register I2CENBST is added to allow software to unambiguously determine when the hardware has completely shutdown in response to the I2CENB register being set from 1 to 0.

2.3.4.1 Procedure

1 Define a timer interval (ti2c_poll) equal to the 10 times the signaling period for the highest I2C transfer speed used in the system and supported by I2C. For example, if the highest I2C transfer mode is 400 kb/s, then this ti2c_poll is 25us.

2 Define a maximum time-out parameter, MAX_T_POLL_COUNT, such that if any repeated polling operation exceeds this maximum value, an error is reported.

3 Execute a blocking thread/process/function that prevents any further I2C master transactions to be started by software, but allows any pending transfers to be completed.
NOTE: This step can be ignored if I2C is programmed to operate as an I2C slave only.

4 The variable POLL_COUNT is initialized to zero.

5 Set I2CENB to 0.

6 Read the I2CENBST register and test the I2C_EN bit (bit 0). Increment POLL_COUNT by one. If POLL_COUNT >= MAX_T_POLL_COUNT, exit with the relevant error code.

7 If I2CENBST[0] is 1, then sleep for ti2c_poll and proceed to the previous step.
Otherwise, exit with a relevant success code.

3 Synchronous Serial Interface

3.1 Overview

The SSI is a full-duplex synchronous serial interface and can connect to a variety of external analog-to-digital (A/D) converters, audio and telecom codecs, and other devices that use serial protocols for transferring data. The SSI supports National's Microwire, Texas Instruments Synchronous Serial Protocol (SSP), and Motorola's Serial Peripheral Interface (SPI) protocol.

The SSI operates in master mode (the attached peripheral functions as a slave) and supports serial bit rates from 7.2 KHz to 54 MHz. Serial data formats may range from 2 to 17 bits in length. The SSI provides 128 entries deep x 17 bits wide transmit and receive data FIFOs.

The FIFOs may be loaded or emptied by the Central Processor Unit (CPU) using programmed I/O, or DMA transfers while receiving or transmitting.

Features:

- 3 protocols support: National's Microwire, TI's SSP, and Motorola's SPI
- Full-duplex or transmit-only or receive-only operation
- Programmable transfer order: MSB first or LSB first
- 128 entries deep x 17 bits wide transmit and receive data FIFOs
- Configurable normal transfer mode or Interval transfer mode
- Programmable clock phase and polarity for Motorola's SSI format
- Two slave select signal (SSI_CE_ / SSI_CE2_) supporting up to 2 slave devices
- Back-to-back character transmission/reception mode
- Loop back mode for testing

3.2 Pin Description

Table 3-1 Micro Printer Controller Pins Description

Name	I/O	Description
SSI_CLK	Output	Serial bit-rate clock
SSI_CE_	Output	First slave select enable
SSI_CE2_	Output	Second slave select enable
SSI_GPC	Output	General purpose control signal to external chip
SSI_DT	Output	Transmit data (serial data out)
SSI_DR	Input	Receive data (serial data in)

SSI_CLK is the bit-rate clock driven from the SSI to the peripheral. SSI_CLK is toggled only when data is actively being transmitted and received.

SSI_CE_ or SSI_CE2_ are the framing signal, indicating the beginning and the end of a serialized data word.

SSI_DT and SSI_DR are the Transmit and Receive serial data lines.

SSI_GPC is general-purpose control signal, synchronized with SSI_CLK, can be used for LCD control.

3.3 Register Description

The SSI has seven registers: one data, two control, one status, one bit-rate control, and two interval control registers. The table lists these registers.

Table 3-2 SSI Serial Port Registers

Name	RW	Reset Value	Address Offset	Access Size
SSIDR0	RW	0x??	0x10043000	32
SSICR00	RW	0x0000	0x10043004	16
SSICR10	RW	0x00087860	0x10043008	32
SSISR0	RW	0x00000098	0x1004300C	32
SSIITR0	RW	0x0000	0x10043010	16
SSIICR0	RW	0x00	0x10043014	8
SSIGR0	RW	0x0000	0x10043018	16
SSIDR1	RW	0x??	0x10044000	32
SSICR01	RW	0x0000	0x10044004	16
SSICR11	RW	0x00087860	0x10044008	32
SSISR1	RW	0x00000098	0x1004400C	32
SSIITR1	RW	0x0000	0x10044010	16
SSIICR1	RW	0x00	0x10044014	8
SSIGR1	RW	0x0000	0x10044018	16

NOTE: There two SSI controller. SSI0 whose base address is 0x100430xx and SSI1 whose base address is 0x100440xx.

3.3.1 SSI Data Register (SSIDR)

SSIDR **0x10043000, 0x10044000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved															GPC/D16	Transmit/Receive Data															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:17	Reserved	Writing has no effect, read as zero.	R
16	GPC/D16	This bit can be used as normal data bus bit 16 or GPC bit alternatively. When it is used as normal data bus bit, it's readable / writable; when SSI_GPC is used, it is GPC bit for SSI_GPC pin output and it's write-only.	RW
15:0	Transmit/	Data word to be written to/read from Transmit/Receive FIFO.	RW

	Receive Data	When the transfer frame length is less than 17-bit, received data is automatically right justified in the receive-FIFO and the upper unused bits are filled with '0'. For transmission, the upper unused bits of the data written into SSIDR is ignored by the transmit logic. (**NOTE:** "upper unused bits" does not include the SSIDR.GPC bit. National microwire format includes format 1 and format2, when national microwire format 2 is selected, Bit 16 of SSIDR is defined as read/write operation judge bit, if it is 0, bit 15~0 represent one read command; if it is 1, bit 15~0 represent one write command and following is the written data. So the maximum length of one command (is defined in MCOM) is 16, the maximum length of one written or read data (is defined in FLEN) can be 17. Transmit-FIFO only contain one read operation command once, or one write operation command and its data once, after transmit-FIFO is empty, next command can be filled in transmit-FIFO.	

3.3.2 SSI Control Register0 (SSICR0)

SSICR0 **0x10043004, 0x10044004**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SSIE	TIE	RIE	TEIE	REIE	LOOP	RFINE	RFINC	EACLRUN	FSEL	Reserved		TFMODE	TFLUSH	RFLUSH	DISREV
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15	SSIE	This bit is used to enable/disable SSI module. 0: disable; 1: enable. Clearing SSIE will not reset SSI FIFO, SSICR0, SSICR1, SSIGR, SSIITR and SSIICR automatically. Software should ensure the FIFOs/registers are properly configured and be flush/reset manually when necessary before enabling SSI.	RW
14	TIE	This bit enables/disables the transmit-FIFO half-empty interrupt TXI. 0: disable; 1: enable.	RW
13	RIE	This bit enables/disables the receive-FIFO half-full interrupt RXI. 0: disable; 1: enable.	RW
12	TEIE	This bit enables/disables the transmit-error interrupt TEI. 0: disable; 1: enable.	RW
11	REIE	This bit enables/disables the receive-error interrupt REI. 0: disable; 1: enable.	RW
10	LOOP	Used for test purpose. In loop mode, the output of SSI transmit shift register is connected to input of SSI receive shift register internally. The data	RW

<table>
<tr><td></td><td></td><td colspan="2">received should be the same as the data transmitted. And do not output any valid signals on the pins.
0: normal SSI mode; 1: LOOP mode.</td><td></td></tr>
<tr><td>9</td><td>RFINE</td><td>This bit enables/disables receive finish control function.
0: disable; 1: enable.
For SSICR1.FMAT = B’10 (National Microwire format 1 is selected), SSICR0.RFINE must be 0.</td><td>The receive finish condition list below:
<table><tr><th>RFINE</th><th>RFINC</th><th>Receive Finish Condition</th></tr><tr><td>0</td><td>x</td><td>Same as transmit completion condition (transmit-fifo is empty and SSICR1.UNFIN = 0)</td></tr><tr><td>1</td><td>0</td><td>Receive continue</td></tr><tr><td>1</td><td>1</td><td>Receive finish</td></tr></table></td><td>RW</td></tr>
<tr><td>8</td><td>RFINC*</td><td>Receive finish control bit.
0: receive continue
1: receive finished</td><td></td><td>RW</td></tr>
<tr><td>7</td><td>EACLRUN</td><td colspan="2">0: don’t auto clear under flag, software clear under
1: software auto clear under flag when tfifo don’t empty</td><td>RW</td></tr>
<tr><td>6</td><td>FSEL</td><td colspan="2">This bit sets the frame signal to be used for slave select. The unselected frame signal always output invalid level.
0: SSI_CE_ is selected
1: SSI_CE2_ is selected</td><td>RW</td></tr>
<tr><td>5:4</td><td>Reserved</td><td colspan="2">Writing has no effect, read as zero.</td><td>R</td></tr>
<tr><td>3</td><td>TFMODE</td><td colspan="2">0: new fifo empty mode
1: old fifo empty mode</td><td>RW</td></tr>
<tr><td>2</td><td>TFLUSH</td><td colspan="2">Flush the transmit FIFO when set to 1. Always return 0 when read.</td><td>RW</td></tr>
<tr><td>1</td><td>RFLUSH</td><td colspan="2">Flush the receive FIFO when set to 1. Always return 0 when read.</td><td>RW</td></tr>
<tr><td>0</td><td>DISREV</td><td colspan="2">This bit enables/disables receive function. 0: enable; 1: disable.</td><td>RW</td></tr>
</table>

NOTE:

***:** When transmitting finished or for receive-only operation, transmit function can be disabled and this bit is used to control receiving completion, and the SSI will consume less power.

When the finish condition is set, the receiving will complete after present character is completely shifted in, then the SSI will stop the SSI_CLK and negate the SSI_CE_ / SSI_CE2_ if necessary. To make sure present transfer is completed, user must read and get SSISR.END = 1 (or SSISR.BUSY = 0).

3.3.3 SSI Control Register1 (SSICR1)

SSICR1 **0x10043008, 0x10044008**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FRMHL		TFVCK		TCKFI		LFST	ITFRM	UNFIN	Reserved	FMAT		TTRG				MCOM				RTRG				FLEN				Reserved		PHA	POL
RST	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	0	0	0	0	0

Bits	Name	Description	RW
31:30	FRMHL	Frame valid level select, FRMHL [1: 0] correspond to SSI_CE2_ and SSI_CE_ respectively. **FRMHL[1:0]** / **Description** / 00 / SSI_CE_ is low level valid and SSI_CE2_ is low level valid / Initial value 01 / SSI_CE_ is high level valid and SSI_CE2_ is low level valid / 10 / SSI_CE_ is low level valid and SSI_CE2_ is high level valid / 11 / SSI_CE_ is high level valid and SSI_CE2_ is high level valid /	RW
29:28	TFVCK	Time from frame valid to clock start, that provide programmable time delay from frame (SSI_CE_ /SSI_CE2_) assert edge to SSI_CLK leading edge. When TFVCK = B'00, the time is fixed half SSI_CLK or one SSI_CLK cycle according to SSICR1.POL and SSICR1.PHA configuration. For SSICR1.FMAT = B'01, SSICR1.TFVCK is ignored. **TFVCK[1:0]** / **Description** / 00 / Ignore (default half or one SSI_CLK cycle delay time) / Initial value 01 / 1 more SSI_CLK cycle delay time is added / 10 / 2 more SSI_CLK cycle delay time is added / 11 / 3 more SSI_CLK cycle delay time is added /	RW
27:26	TCKFI	Time from clock stop to frame invalid, provide programmable time delay from SSI_CLK last edge to frame (SSI_CE_ /SSI_CE2_) negate edge. When TCKFI = B'00, the time is fixed one SSI_CLK or half SSI_CLK cycle according to SSICR1.POL and SSICR1.PHA configuration. For SSICR1.FMAT = B'01, SSICR1.TFVCK is ignored. **TCKFI[1:0]** / **Description** /	RW

00	Ignore (default half or one SSI_CLK cycle delay time)	Initial value	
01	1 more SSI_CLK cycle delay time is added		
10	2 more SSI_CLK cycle delay time is added		
11	3 more SSI_CLK cycle delay time is added		

25	LFST	Set to LSB first or MSB first when transfer. 0: MSB first; 1: LSB first.	RW
24	ITFRM	Frame during interval, selects if the Frame (SSI_CE_ /SSI_CE2_) signal is negated or not during interval time at Interval Mode (SSICR1.FMAT = B'00 and SSIITR.IVLTM ≠ H'0000). It's ignored at Normal Mode. 0: SSI_CE_ /SSI_CE2_ de-asserts during interval time at Interval Mode 1: SSI_CE_ /SSI_CE2_ keeps asserted during interval time at Interval Mode	RW
23	UNFIN	This bit controls whether the SSI finishes transmission or wait for data filling (underrun happen) after all data in transmit-FIFO are sent out during transfer. This bit must be cleared to 0 when SSICR1.FMAT = B'01. (TI's SSP format) 0: Transmit-FIFO empty means end of transmission 1: Transmission didn't finish when transmit-FIFO is empty, SSI underrun error would occur and SSI waits for data filling; SSI_CLK and SSI_CE_ /SSI_CE2_ keeps asserted, SSI_CLK stop at the current level **NOTE:** For transmit-FIFO empty before any transfer after SSI enabled, if SSICR1.UNFIN = 1 or SSICR0.RFINE = 0, SSI will wait till transmit-FIFO isn't empty then start to transfer and no underrun error will occur; if SSICR1.UNFIN = 0 and SSICR0.RFINE = 1, after transmit-FIFO become empty, SSI will start a receive-only transfer.	RW
22	Reserved	Writing has no effect, read as zero.	R
21:20	FMAT	These bits set the operating transfer format.	RW

FMAT[1:0]	Description	
00	Motorola's SPI format	Initial value
01	TI's SSP format	
10	National Microwire 1 format	
11	National Micowire 2 format	

19:16	TTRG	These bits define the transmit-FIFO half-empty threshold value, which when equal or less characters left in transmit-FIFO, the SSISR.TFHE will be set to '1'. 0000: less than or equal to 1 n: less than or equal to nx8	RW
15:12	MCOM	When SSICR1.FMAT = B'10 or B'11 (National Microwire format 1 or 2 is	RW

<table>
<tr><td></td><td></td><td>selected), this bit decides the length of command from 1-bit to 16-bit. The length of written or read data is defined in FLEN. For SSICR1.FMAT ≠ B’10 or B’11, this bit is ignored.
<table>
<tr><th>MCOM[1:0]</th><th>Description</th><th></th></tr>
<tr><td>0000</td><td>1-bit command selected</td><td></td></tr>
<tr><td>0001</td><td>2-bit command selected</td><td></td></tr>
<tr><td>0010</td><td>3-bit command selected</td><td></td></tr>
<tr><td>0011</td><td>4-bit command selected</td><td></td></tr>
<tr><td>0100</td><td>5-bit command selected</td><td></td></tr>
<tr><td>0101</td><td>6-bit command selected</td><td></td></tr>
<tr><td>0110</td><td>7-bit command selected</td><td></td></tr>
<tr><td>0111</td><td>8-bit command selected</td><td>Initial value</td></tr>
<tr><td>1000</td><td>9-bit command selected</td><td></td></tr>
<tr><td>1001</td><td>10-bit command selected</td><td></td></tr>
<tr><td>1010</td><td>11-bit command selected</td><td></td></tr>
<tr><td>1011</td><td>12-bit command selected</td><td></td></tr>
<tr><td>1100</td><td>13-bit command selected</td><td></td></tr>
<tr><td>1101</td><td>14-bit command selected</td><td></td></tr>
<tr><td>1110</td><td>15-bit command selected</td><td></td></tr>
<tr><td>1111</td><td>16-bit command selected</td><td></td></tr>
</table></td><td></td></tr>
<tr><td>11:8</td><td>RTRG
(SSI1)</td><td>These bits define the receive-FIFO half-full threshold value, which when equal or more characters received in receive-FIFO, the SSISR.RFHF will be set to ‘1’.
0000: more than or equal to 1
n: more than or equal to nx8</td><td>RW</td></tr>
<tr><td>9:8</td><td>RTRG
(SSI0)</td><td>These bits define the receive-FIFO half-full threshold value, which when equal or more characters received in receive-FIFO, the SSISR.RFHF will be set to “1”.
<table>
<tr><th>RTRG[1:0]</th><th>Description</th><th></th></tr>
<tr><td>00</td><td>more than or equal to 1</td><td>Initial value</td></tr>
<tr><td>01</td><td>more than or equal to 4</td><td></td></tr>
<tr><td>10</td><td>more than or equal to 8</td><td></td></tr>
<tr><td>11</td><td>more than or equal to 14</td><td></td></tr>
</table></td><td>RW</td></tr>
<tr><td>7:4</td><td>FLEN</td><td>These bits set the bit length of every character to be transmitted/received. The maximum data length can be configured is 17 bits. For data length longer than 17 bits (multiples of the SSICR1.FLEN configured length), the software should ensure properly processing. When SSI_GPC pin is used, the FLEN shouldn’t be configured as B’1111 (17-bit data). When TI SSP mode is selected (FMAT = 2’b01), 2-bit data length (FLEN = 4’b0000) isn’t supported.
<table>
<tr><th>MCOM[1:0]</th><th>Description</th><th></th></tr>
<tr><td>0000</td><td>2-bit data</td><td></td></tr>
</table></td><td>RW</td></tr>
</table>

		0001	3-bit data		
		0010	4-bit data		
		0011	5-bit data		
		0100	6-bit data		
		0101	7-bit data		
		0110	8-bit data	Initial value	
		0111	9-bit data		
		1000	10-bit data		
		1001	11-bit data		
		1010	12-bit data		
		1011	13-bit data		
		1100	14-bit data		
		1101	15-bit data		
		1110	16-bit data		
		1111	17-bit data		
3:2	Reserved	Writing has no effect, read as zero.			R
1	PHA	This bit sets the phase of the SSI_CLK from the beginning of a data frame for Motorola's SPI format (SSICR1.FMAT = B'00). 0: The leading edge of SSI_CLK is used to sample data from SSI_DR after the SSI_CE_ /SSI_CE2_ goes valid, it is initial value 1: The leading edge of SSI_CLK is used to drive data onto SSI_DT after the SSI_CE_ /SSI_CE2_ goes valid			RW
0	POL	This bit sets SSI_CLK's idle state polarity for Motorola's SPI format. (SSICR1.FMAT = B'00). 0: SSI_CLK keeps low level when idle, when SSI_CE_ /SSI_CE2_ goes valid the leading clock edge is a rising edge, it is initial value 1: SSI_CLK keeps high level when idle, when SSI_CE_ /SSI_CE2_ goes valid the leading clock edge is a falling edge			RW

3.3.4 SSI Status Register1 (SSISR)

SSISR **0x1004300C, 0x1004400C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								TFIFO-NUM								RFIFO-NUM								END	BUSY	TFF	RFE	TFHE	RFHF	UNDR	OVER
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0

Bits	Name	Description	RW
31:24	Reserved	Writing has no effect, read as zero.	R

23:16	TFIFO-NUM	These bits indicate the Characters Number in Transmit-FIFO.	R
15:8	RFIFO-NUM	These bits indicate the Characters Number in Transmit-FIFO.	R
7	END	This bit indicates transfer end status. It is the inverse of SSISR.BUSY when transfer is in process, but it'll keep cleared at interval time before transfer is completed. It'll be set when transfer finished.	R
6	BUSY	This bit indicates SSI's working status. 0: SSI is idle or at interval time 1: Transmission and/or reception is in process	R
5	TFF	This bit denotes transmit-FIFO is full or not. 0: Transmit-FIFO is not full 1: Transmit-FIFO is full	R
4	RFE	This bit denotes receive-FIFO is empty or not. 0: Receive-FIFO is not empty 1: Receive-FIFO is empty	R
3	TFHE	This bit denotes whether the characters number in transmit-FIFO being less or equal to SSICR1.TTRG. 0: The data in transmit-FIFO is more than the condition set by SSICR1.TTRG 1: The data in transmit-FIFO meets the condition set by SSICR1.TTRG, If SSICR0.TIE = 1, it will generate SSI TXI interrupt	R
2	RFHF	This bit denotes whether the characters number in receive-FIFO being more or equal to the number set by SSICR1.RTRG. 0: The data in receive-FIFO is less than the condition set by SSICR1.RTRG 1: The data in receive-FIFO meets the condition set by SSICR1.RTRG, If SSICR0.RIE = 1, it will generate SSI RXI interrupt	R
1	UNDR	Transmit-FIFO underrun status. When underrun happens, SSI set this bit and keeps the current status of SSI_CLK and SSI_CE_/SSI_CE2_, waiting for transmit-FIFO filling. 0: Underrun has not occurred 1: Underrun has occurred, when SSICR0.TEIE is set, it will generate SSI TEI interrupt. Write '0' to clear this bit, writing '1' has no effect	RW
0	OVER	Receive-FIFO overrun status, new received data will lose. 0: Overrun has not occurred 1: Overrun has occurred; When SSICR0.REIE is set, it will generate SSI REI interrupt. Write '0' to clear this bit, writing '1' has no effect	RW

3.3.5 SSI Interval Time Control Register (SSIITR)

SSIITR 0x10043010, 0x10044010

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CNTCLK	IVLTM														
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15	CNTCLK	Counting clock source select. 0: Use SSI bit clock (SSI_CLK) as the interval counter clock source 1: Use 32K clock as the interval counter clock source	RW
14:0	IVLTM	Interval time set, set the cycle number of counting clock source for desired interval time. When SSIITR.IVLTM = 0x0000, normal mode is selected, and SSIITR.CNTCLK and SSIICR are ignored. When SSIITR.IVLTM ≠ 0x0000, interval mode is selected. The interval time is calculated as follows: *Interval time ≈ [CNTCLK clock period] * [Value of IVLTM]* The actual interval time is as follow: When SSIITR.CNTCLK = 0: *Interval time = [CNTCLK clock period] * [Value of IVLTM] + 3 * device_clock period* When SSIITR.CNTCLK = 1: *Interval time ≥ [CNTCLK clock period] * [Value of IVLTM + 1] + 1 * device_clock period;* *Interval time ≤ [CNTCLK clock period] * [Value of IVLTM + 2] + 2 * device_clock period*	RW

3.3.6 SSI Interval Character-per-frame Control Register (SSIICR)

SSIICR 0x10043014, 0x10044014

Bit	7	6	5	4	3	2	1	0
	Reserved					ICC		
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:3	Reserved	Writing has no effect, read as zero.	R
2:0	ICC	Sets the fixed number of characters to be transmitted / received each time during SSI_CLK changing (and SSI_CE_ / SSI_CE2_ asserting) in interval mode for SSICR1.FMAT = B'00 (Motorola's SPI format is selected). SSIICR is ignored for SSICR1.FMAT ≠ B'00. The desired transfer number of characters-per-frame is (SSIICR set value + 1).	RW

3.3.7 SSI Clock Generator Register (SSIGR)

SSIGR 0x10043018, 0x10044018

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								CGV							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:8	Reserved	Writing has no effect, read as zero.	R
7:0	CGV	Sets the frequency of serial bit clock (SSI_CLK). The serial bit clock (SSI_CLK) is generated by dividing device-clock as follows: F_{SSI_CLK} = [Frequency of device clock] / (2 * (CGV + 1)) Device clock is generated in CPM module. The value in SSIGR can be set from 0 to 255, and initialized to 0x0000 on power-on reset.	RW

3.4 Functional Description

Serial data is transferred between the processor and external peripheral through FIFO buffers in the SSI. Data transfers to system memory are handled by either the CPU (using programmed I/O) or by DMA. Operation is full duplex - separate buffers and serial data paths permit simultaneous transfers to and from the external peripheral.

Programmed I/O transmits and receives data directly between the CPU and the transmit/receive FIFO's. The DMA controller transfers data during transmit and receive operations between memory and the FIFO's.

Transmit data is written by the CPU or DMA to the SSI's transmit FIFO. The SSI then takes the data from the FIFO, serializes it, and transmits it via the SSI_DT signal to the peripheral. Data from the peripheral is received via the SSI_DR signal, converted to parallel words and is stored in the Receive FIFO. Read operations automatically target the receive FIFO, while write operations write data to the transmit FIFO. Both the transmit and receive FIFO buffers are 128 entries deep by 17 bits wide. As the received data fills the receive FIFO, a programmable threshold triggers an interrupt to the Interrupt Controller. If enabled, an interrupt service routine responds by identifying the source of the interrupt and then performs one or several read operations from the inbound (receive) FIFO buffer.

3.5 Data Formats

Four signals are used to transfer data between the processor and external peripheral. The SSI supports three formats: Motorola SPI, Texas Instruments SSP, and National Microwire. Although they have the same basic structure the three formats have significant differences, as described below:

1 SSI_CE_/SSI_CE2_ varies for each protocol as follows:
- For SPI and Microwire formats, SSI_CE_/SSI_CE2_ functions as a chip select to enable the external device (target of the transfer), and is held active-low during the data transfer.
- For SSP format, this signal is pulsed high for one serial bit-clock period at the start of each frame.

2 SSI_CLK varies for each protocol as follows:
- For Microwire, both transmit and receive data sources switch data on the falling edge of SSI_CLK, and sample incoming data on the rising edge.
- For SSP, transmit and receive data sources switch data on the rising edge of SSI_CLK, and sample incoming data on the falling edge.
- For SPI, the user has the choice of which edge of SSI_CLK to use for switching outgoing data, and for sampling incoming data. In addition, the user can move the phase of SSI_CLK, shifting its active state one-half period earlier or later at the start and end of a frame.

While SSP and SPI are full-duplex protocols, Microwire uses a half-duplex master-slave messaging protocol. At the start of a frame, a 1 or 2-byte control message is transmitted from the controller to the peripheral. The peripheral does not send any data. The peripheral interprets the message and, if it is a READ request, responds with requested data, one clock after the last bit of the requesting message.

The serial clock (SSI_CLK) only toggles during an active frame. At other times it is held in an inactive or idle state, as defined by its specified protocol.

3.5.1 Motorola's SPI Format Details

3.5.1.1 General Single Transfer Formats

The figures below show the timing of general single transfer format.

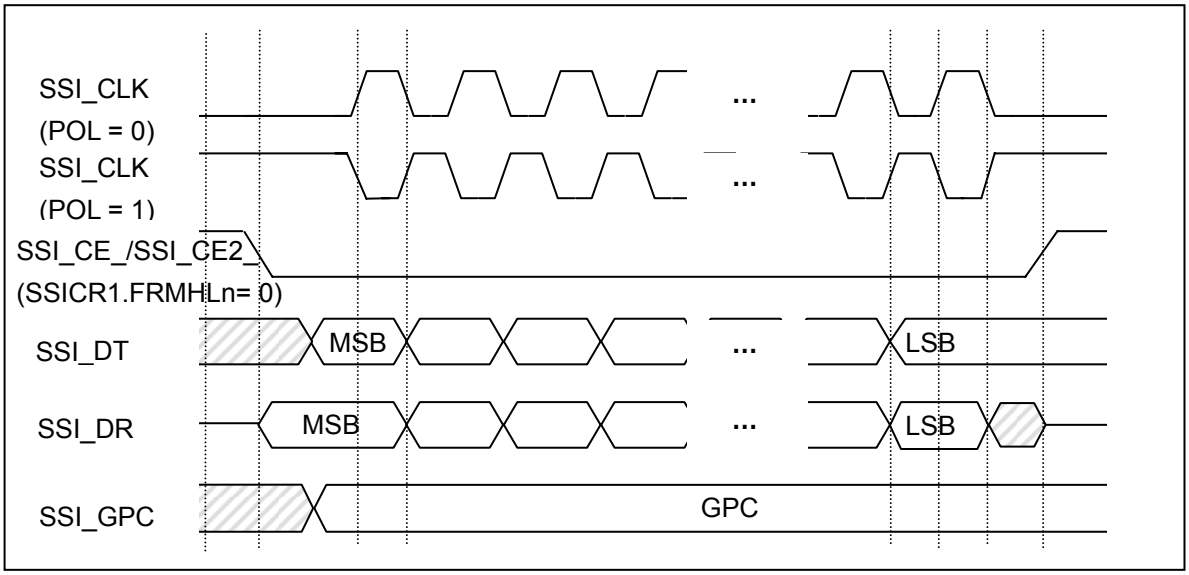


Figure 3-1 SPI Single Character Transfer Format (PHA = 0)

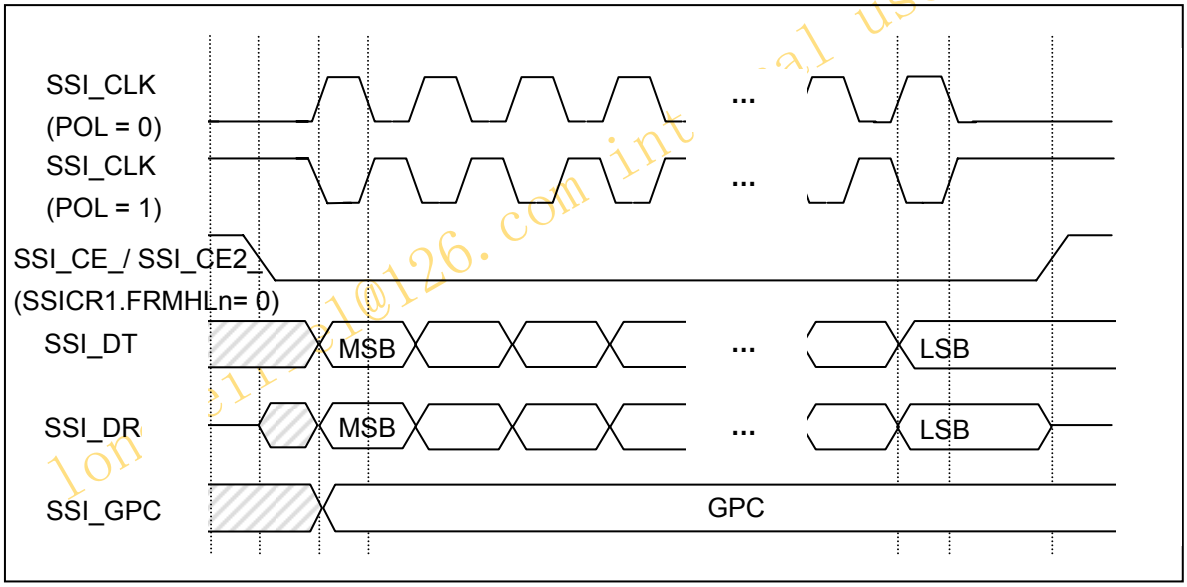


Figure 3-2 SPI Single Character Transfer Format (PHA = 1)

For SSICR1.PHA = 0, when SSICR1.TFVCK = B'00, hardware ensures the first clock edge appears one SSI_CLK period after SSI_CE_ / SSI_CE2_ goes valid; when SSICR1.TCKFI = B'00, hardware ensures the SSI_CE_ / SSI_CE2_ negated half SSI_CLK period after last clock change edge; when SSICR1.TFVCK ≠ B'00 or SSICR1.TCKFI ≠ B'00, 1/2/3 more clock cycles are inserted.

For SSICR1.PHA = 1, when SSICR1.TFVCK = B'00, hardware ensures the first clock edge appears half SSI_CLK period after SSI_CE_ / SSI_CE2_ goes valid; when SSICR1.TCKFI = B'00, hardware ensures the SSI_CE_ / SSI_CE2_ negated one SSI_CLK period after last clock change edge; when SSICR1.TFVCK ≠ B'00 or SSICR1.TCKFI ≠ B'00, 1/2/3 more clock cycles are inserted.

Data is sampled from SSI_DR at every rising edge (when PHA = 0, POL = 0 or PHA = 1, POL = 1) or at every falling edge (when PHA = 0, POL = 1 or PHA = 1, POL = 0). According to SPI protocol, input data on SSI_DR should be stable at every sample clock edge.

Drive data onto SSI_DT at every rising edge (when PHA = 0, POL = 1 or PHA = 1, POL = 0) or at every falling edge (when PHA = 0, POL = 0 or PHA = 1, POL = 1).

3.5.1.2 Back-to-Back Transfer Formats

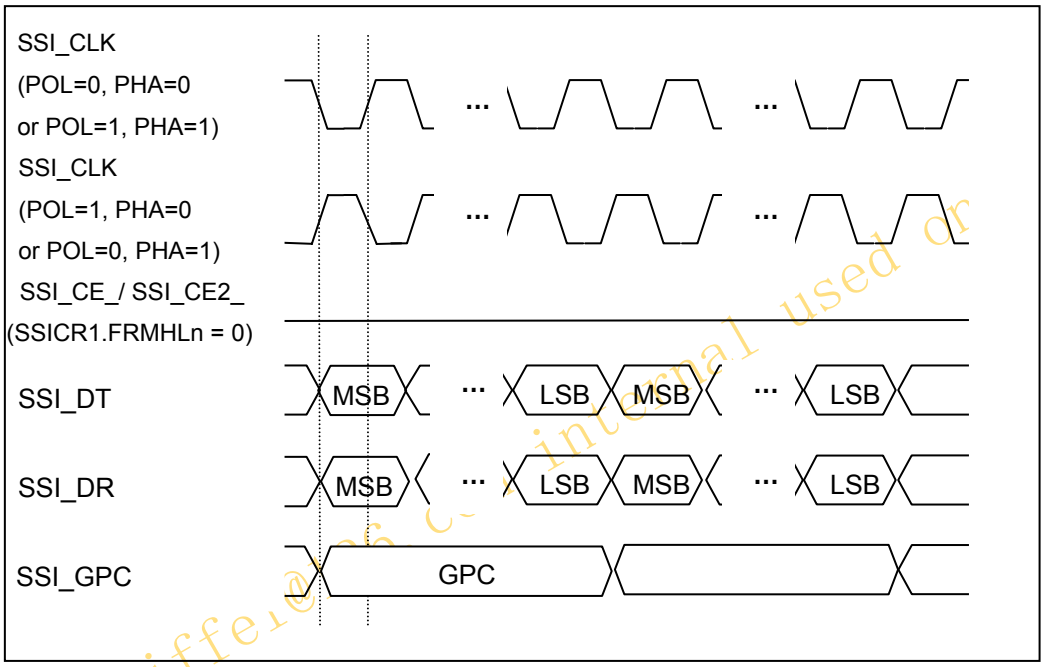


Figure 3-3 SPI Back-to-Back Transfer Format

For Motorola's SPI format transfers those continuous characters are exchanged during SSI_CE_ / SSI_CE2_ being valid, the timing is illustrated in the figure (SSICR1.LFST = 0).

Back-to-back transfer is performed as transmit-only/full-duplex operation when transmit-FIFO is not empty before the completion of the last character's transfer or performed as receive-only operation.

3.5.1.3 Frame Interval Mode Transfer Format

When in interval mode (SSIITR. IVLTM ≠ '0'), SSI always wait for an interval time (SSIITR.IVLTM), transfer fixed number of characters (SSIICR), then repeats the operation.

When SSICR0.RFINE = 1, if transmit-FIFO is still empty after the interval time, receive-only transfer will occur.

During interval-wait time, SSI stops SSI_CLK, and when SSICR1.ITFRM = 0 it negates the SSI_CE_ / SSI_CE2_, when SSICR1.ITFRM = 1 it keeps asserting the SSI_CE_ / SSI_CE2_.

For transfers finished with transmit-FIFO empty, if the SSI transmit-FIFO is empty before fixed number of characters being loaded to transfer (SSICR1.UNFIN must be 1), then the SSI will set SSISR.UNDR = 1; if enabled, it'll send out a SSI underrun interrupt. At the same time, SSI will hold the SSI_CE_ / SSI_CE2_ and SSI_CLK signals at current status and wait for the transmit-FIFO filling. The SSI will continue transfer after transmit-FIFO being filled. The SSI always stops after completion of fixed number of characters' transfer (SSICR1.UNFIN must be 0) with transmit-FIFO empty.

For transfers finished by SSICR0.RFINC being valid set, the SSI will stop after finished current character transfer and needn't wait for a whole completion of fixed number of characters' transfer.

Two Interval transfer mode are illustrated in the following figures. In these timing diagram, SSICR1.PHA = 0, SSICR1.POL = 0 and SSIICR = 0.

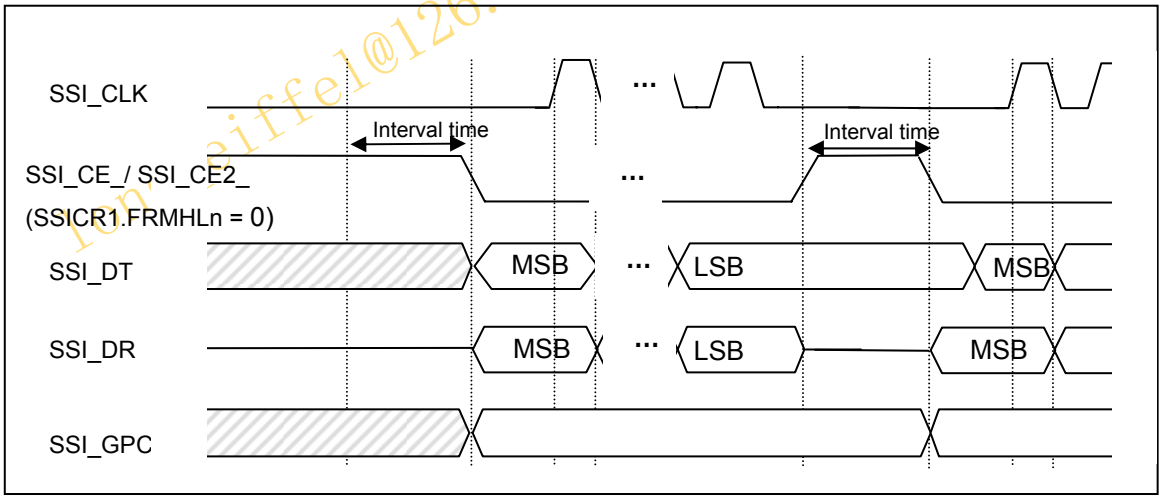


Figure 3-4 SPI Frame Interval Mode Transfer Format (ITFRM = 0, LFST = 0)

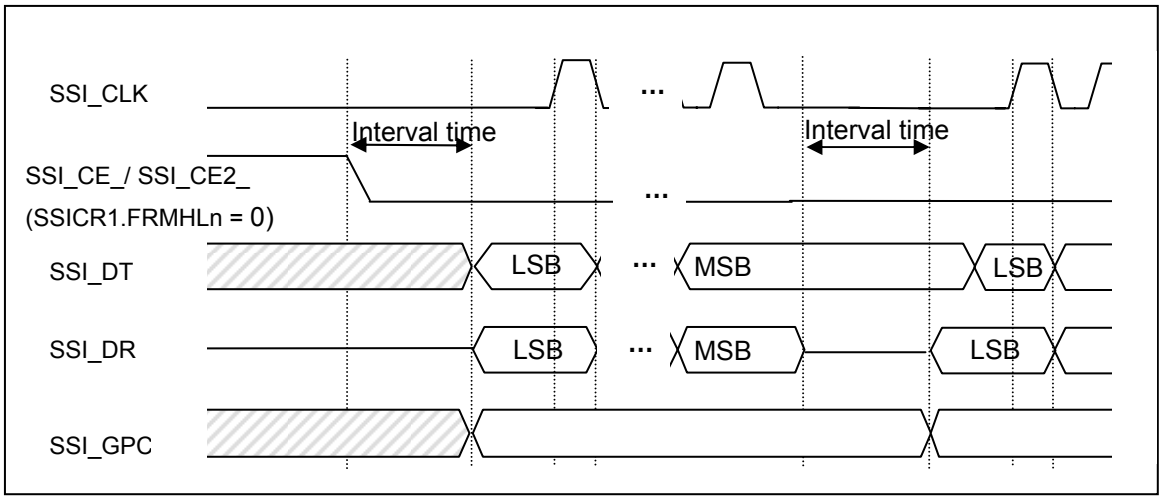


Figure 3-5 SPI Frame Interval Mode Transfer Format (ITFRM = 1, LFST = 1)

3.5.2 TI's SSP Format Details

In this format, each transfer begins with SSI_CE_ pulsed high for one SSI_CLK period. Then both master and slave drive data at SSI_CLK's rising edge and sample data at the falling edge. Data are transferred with MSB first or LSB first. At the end of the transfer, SSI_DT retains the value of the last bit sent through the next idle period.

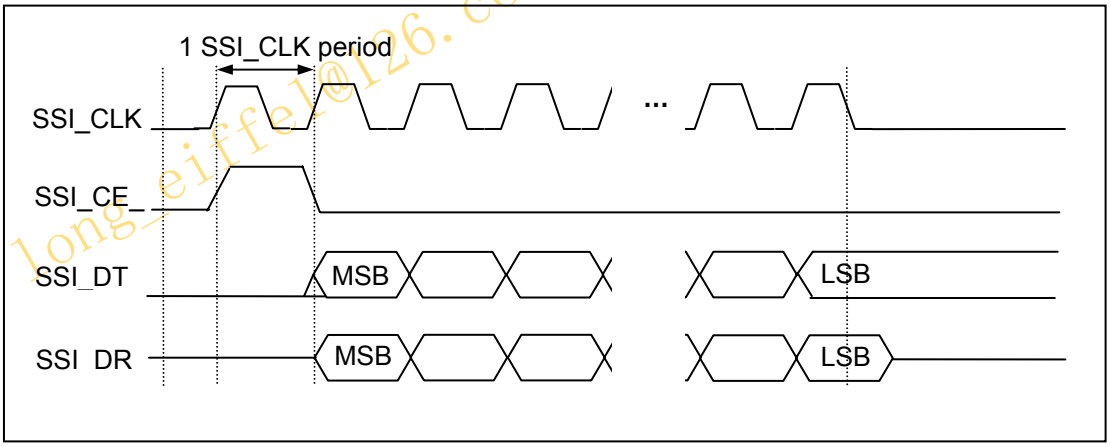


Figure 3-6 TI's SSP Single Transfer Format

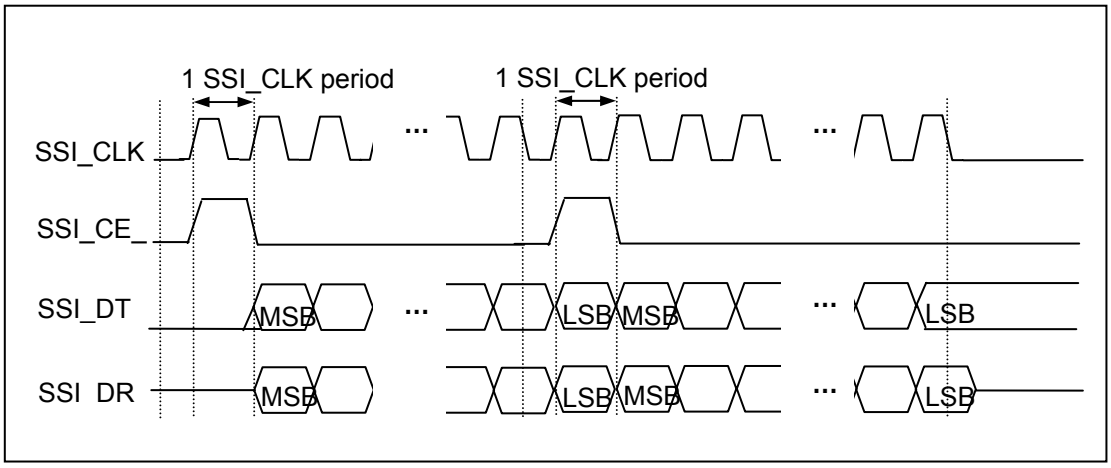


Figure 3-7 TI's SSP Back-to-back Transfer Format

3.5.3 National Microwire Format Details

It supports format 1 and format 2. If format 1 is selected, both master and slave drive data at SSI_CLK falling edge and sample data at the rising edge. If format 2 is selected, master drive and sample data at SSI_CLK falling edge, slave drive and sample data at SSI_CLK rising edge. SSI_CLK goes high midway through the command's most significant bit (or LSB) and continues to toggle at the bit rate. One bit clock (format 1) or half one bit clock (format 2) period after the last command bit, the external slave must return the serial data requested, with most significant bit first (or LSB first) on SSI_DR. SSI_CE_ / SSI_CE2 de-asserts high half clock (SSI_CLK) period (and 1/2/3 additional clock periods) later. Format 1 support back-to-back transfer, the start and end of back-to-back transfers are similar to those of a single transfer. However, SSI_CE_ / SSI_CE2 remains asserted throughout the transfer. The end of a character data on SSI_DR is immediately followed by the start of the next command byte on SSI_DT.

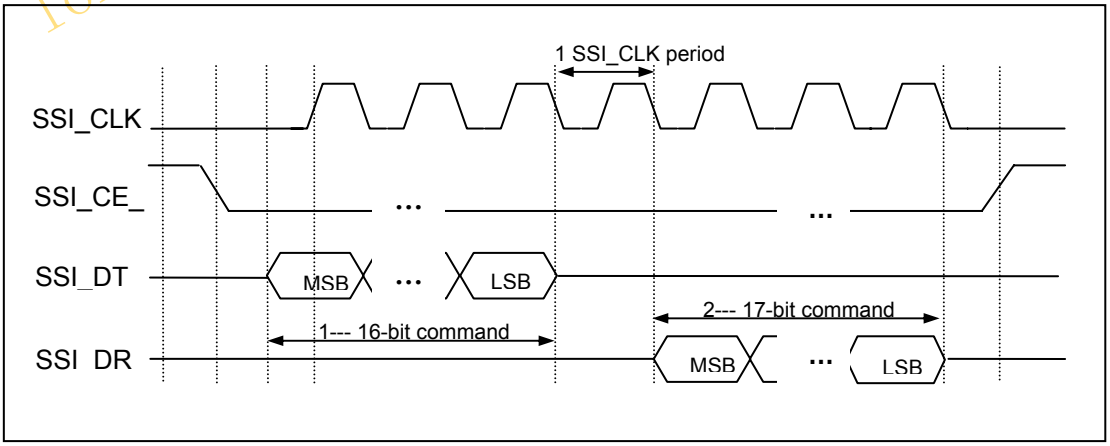


Figure 3-8 National Microwire Format 1 Single Transfer

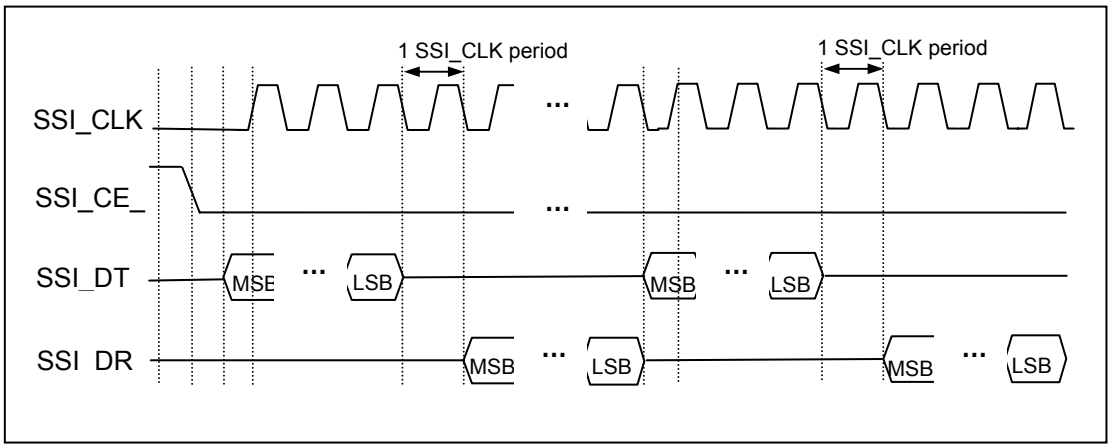


Figure 3-9 National Microwire Format 1 Back-to-back Transfer

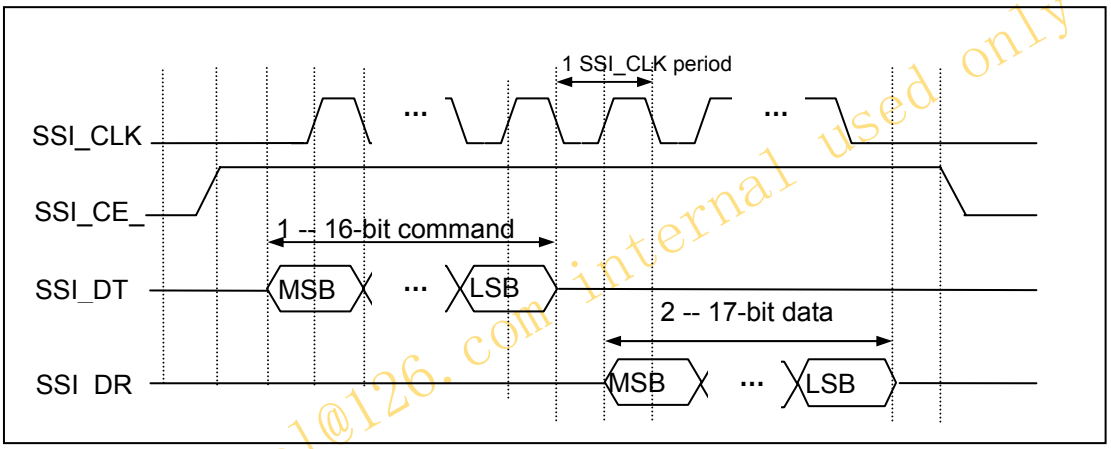


Figure 3-10 National Microwire Format 2 Read Timing

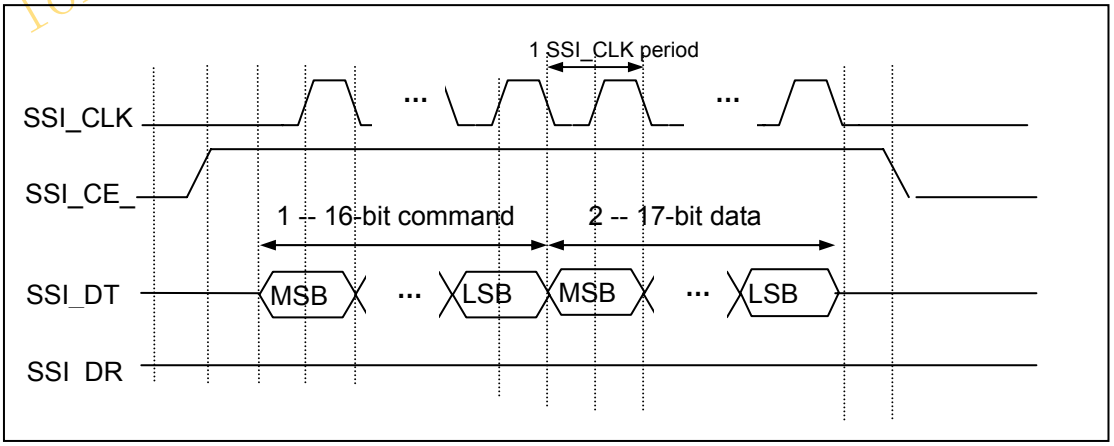


Figure 3-11 National Microwire Format 2 Write Timing

3.6 Interrupt Operation

In SSI, there are TXI, RXI, TEI and REI total 4 interrupts, all these interrupts are combined together to make one SSI interrupt, which can be masked by writing '1' into corresponding mask bit in INTC interrupt mask register (IMR).

Table 3-3 SSI Interrupts

Operation	Condition	Flag Bit	Mask Bit	Interrupt	DMAC Activation
Transmit	T-FIFO is half-empty or less	SSISR.TFHE	SSICR0.TIE	**TXI**	Possible
	Transmit underrun error	SSISR.UNDR	SSICR0.TEIE	**TEI**	Impossible
Receive	R-FIFO is half-full or more	SSISR.RFHF	SSICR0.RIE	**RXI**	Possible
	Receive overrun error	SSISR.OVER	SSICR0.REIE	**REI**	Impossible

Either SSISR.TFHE or SSISR.RFHF can activate DMA transferring when corresponding individual interrupt mask bit in SSICR0 is cleared (masked) and DMA is enabled and configured.

4 One-Wire Bus Interface

4.1 Overview

The OWI has the following features:

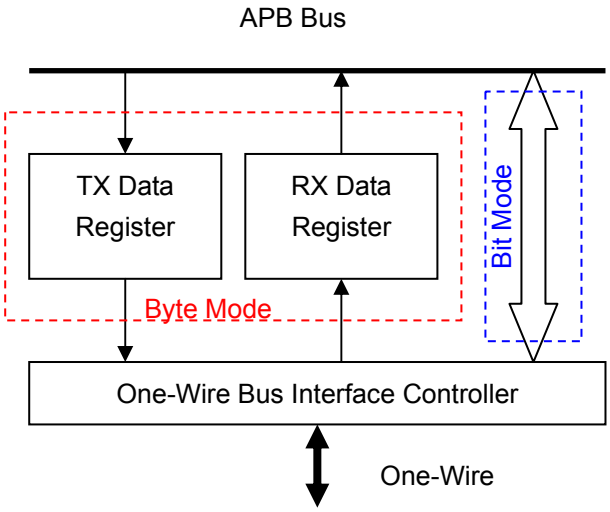
- Support 1-wire bus protocol
- Support Overdrive speed mode and Regular speed mode
- Data is transferred with the LSB first
- Support bit operate mode and byte operate mode
- OWI is the only master on the bus

4.2 Pin Description

Table 4-1 One-Wire Controller Pins Description

Name	I/O	Description
OWDAT	Input/Output	One-Wire Data signal

4.3 Structure



4.4 Register Description

Table 4-2 OWI Registers Description

Name	Description	RW	Reset Value	Address	Access Size
OWCFG	Configure Register	RW	0x00	0x10072000	8
OWCTL	Control Register	RW	0x00	0x10072004	8
OWSTS	Status Register	RW	0x00	0x10072008	8
OWDAT	Data Register	RW	0x00	0x1007200C	8
OWDIV	Clock Divide Register	RW	0x00	0x10072010	8

4.4.1 One-Wire Configure Register (OWCFG)

OWCFG 0x10072000

Bit	7	6	5	4	3	2	1	0
	MODE	RDDATA	WRDATA	RDST	WR1/RD	WR0	RST	ENA
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7	MODE	OWI mode select. 0: Regular speed mode 1: Overdrive speed mode	RW
6	RDDATA	Receive a byte from one-wire bus. This bit is cleared when receive data is complete. The value of received data is stored in OWDAT, and is valid after RDDATA is self-cleared. 0:Do nothing/Receive data is completed 1:Receive data from one-wire bus and stored in OWDAT	RW
5	WRDATA	Transmit the data in OWDAT. This bit is cleared when the transmission of data is complete. 0: Do nothing/Transmission of data completed 1: Transmit the data in OWDAT	RW
4	RDST	Read status. This bit is valid after the WR1/RD bit is self-cleared. 0: 0 was sampled during a read 1: 1 was sampled during a read	R
3	WR1/RD	Write 1/ Read. This bit is cleared when the write of the bit is complete. The valve of one wire can be read, since the Write 1 and Read timing are identical. The value of the read bit is stored in RDST, and is valid after WR1/RD is self-cleared. 0: Do nothing/Write 1 sequence completed	RW

		1: Generate Write 1 sequence on line	
2	WR0	Write 0 on line. This bit is cleared after the presence is determined. 0: Do nothing/Write 0 sequence completed 1: Generate Write 0 sequence on line	RW
1	RST	Reset presence pulse. This bit is cleared after the presence is determined. 0: Do nothing. Reset pulse completed 1: Generate reset pulse and sample slaves presence pulse	RW
0	ENA	Enable of OWI operation. 1: Write 1 to this bit to enable the OWI operation 0: Write 0 to this bit to disable the OWI operation	RW

NOTE: To make the OWI operate normally, only one of the RST, RDDATA, WRDATA, WR1/RD and WR0 is equal to 1.

4.4.2 One-Wire Control Register (OWCTL)

OWCTL **0x10072004**

Bit	7	6	5	4	3	2	1	0
	Reserved					EBYTE	EBIT	ERST
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:3	Reserved	Writing has no effect, read as zero.	R
2	EBYTE	Enable byte write / read interrupt.	RW
1	EBIT	Enable bit write / read interrupt.	RW
0	ERST	Enable reset sequence finished interrupt.	RW

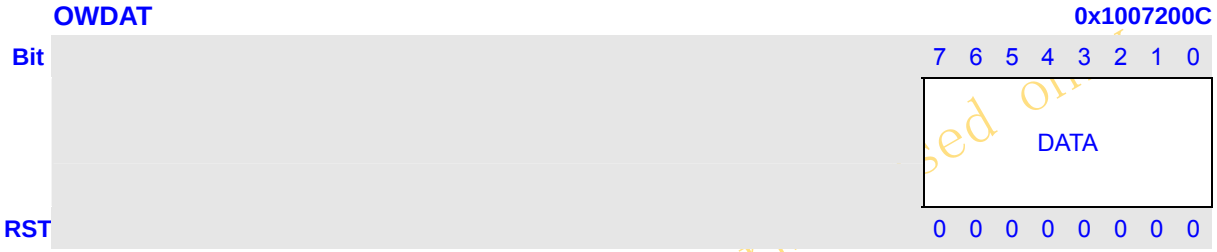
4.4.3 One-Wire Status Register (OWSTS)

OWSTS **0x10072008**

Bit	7	6	5	4	3	2	1	0
	PST	Reserved				BYTE_RDY	BIT_RDY	PST_RDY
RST	0	0	0	0	0	0	0	0

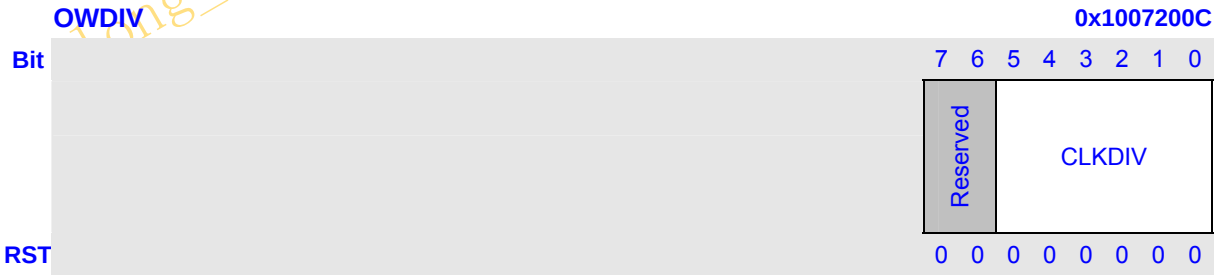
Bits	Name	Description	RW
7	PST	Whether the 1-wire bus has device or not. This bit is valid after the RST bit is self-cleared. 1:The 1-wire bus has device on it 0:The 1-wire bus has no device on it	RW
6:3	Reserved	Writing has no effect, read as zero.	R
2	BYTE_RDY	Have received or transmitted a data.	RW
1	BIT_RDY	Have received or transmitted a bit.	RW
0	PST_RDY	Have finished a reset pulse.	RW

4.4.4 One-Wire Data Register (OWDAT)



Bits	Name	Description	RW
7:0	DATA	Store the received data and is valid after RDDATA is self-cleared. Prepare the transmission data for transmitting.	RW

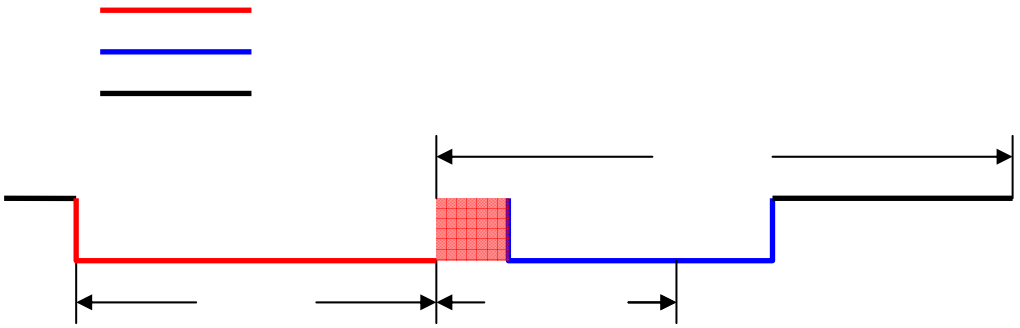
4.4.5 One-Wire Clock Divide Register (OWDIV)



Bits	Name	Description	RW
7:6	Reserved	Writing has no effect, read as zero.	R
5:0	CLKDIV	Controls the divider used to create the DEV_CLK based upon the CPM_OWI_SYSCLK. When the OWI work in the regular speed mode: 1 MHz = DEV_CLK = CPM_OWI_SYSCLK / (CLKDIV + 1). When the OWI work in the overdrive speed mode: 4 MHz = DEV_CLK = CPM_OWI_SYSCLK / (CLKDIV + 1).	RW

4.5 One-Wire Bus Protocol

4.5.1 Reset Timing and ACK Timing

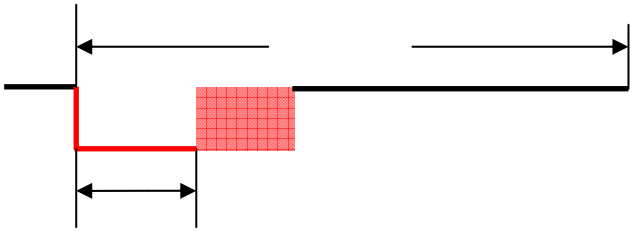


	RSTH	RSTL	RSTS
Regular Speed mode	512us	512us	68us
Overdrive Speed mode	64us	64us	8us

4.5.2 Write 0 Timing

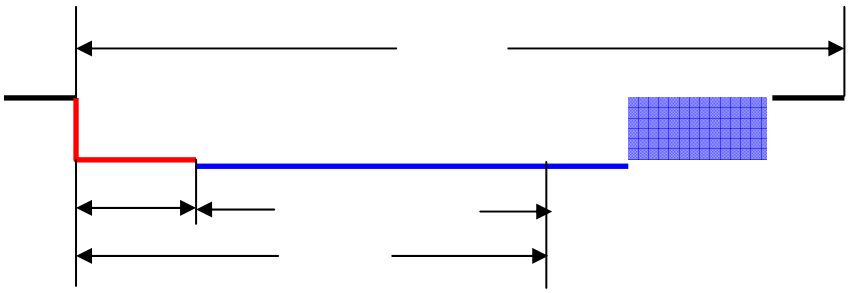
	SLOT	LOW0
Regular Speed mode	128us	100us
Overdrive Speed mode	16us	12us

4.5.3 Write 1 Timing



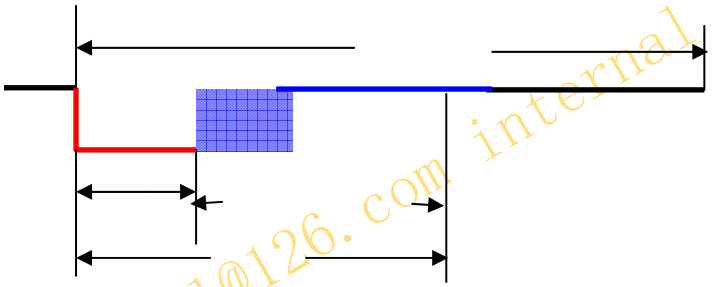
	SLOT	LOW1
Regular Speed mode	128us	5us
Overdrive Speed mode	16us	1.25us

4.5.4 Read0 Timing



	SLOT	LOWR	RDS
Regular Speed mode	128us	5us	13us
Overdrive Speed mode	16us	1.25us	1.75us

4.5.5 Read1 Timing



	SLOT	LOWR	RDS
Regular Speed mode	128us	5us	13us
Overdrive Speed mode	16us	1.25us	1.75us

4.6 One-Wire Operation Guide

- **Interrupt operation guide:**
  1 Write the frequency divider to I2CGR.
  2 Set OWSTS to clear the flag.
  3 Set OWCTL to enable the operation interrupt.
  4 Select OWI mode (Regular speed mode or Overdrive speed mode).
  5 Set RST, RDDATA, WRDATA, WR1/RD or WR0 to choose one kind of operation.
  6 Set ENA to enable OWI.
  7 Wait till the interrupt happened, and the operation is finished.
  8 If you want to disable OWI when OWI is working, you should set ENA to 0 and till ENA is really set to 0, then you can do next operation.

- **CPU operation guide:**
  1 Write the frequency divider to I2CGR.
  2 Select OWI mode (Regular speed mode or Overdrive speed mode).
  3 Set RST, RDDATA, WRDATA, WR1/RD or WR0 to choose one kind of operation.
  4 Set ENA to enable OWI.
  5 Wait the ENA is cleared, and the operation is finished.
  6 If you want to disable OWI when OWI is working, you should set ENA to 0 and till ENA is really set to 0, then you can do next operation.

5 USB Host Controller

5.1 Overview

This chapter describes the Universal Serial Bus host controller (UHC) implemented in the XBurst processor.

The Universal Serial Bus (USB) supports serial data exchanges between a host computer and a variety of simultaneously accessible peripherals. The attached peripherals share USB bandwidth through a host-scheduled, token-based protocol. Peripherals can be attached, configured, used, and detached, while the host and other peripherals continue operation.

Familiarity with the *Universal Serial Bus Specification*, Revision 1.1 and the OHCI specification are necessary to fully understand the material contained in this section.

Features:

- USB Rev. 1.1 compatible
- Supports both low-speed and full-speed USB devices
- Open Host Controller Interface (OHCI) Rev 1.0 compatible
- Root hub supports two data ports

5.2 Pin Description

Table 5-1 UHC Pins Description

Name	Type	Description
DPLS0	Inout	Data Positive to Port 0
DPLS1	Inout	Data Positive to Port 1
DMNS0	Inout	Data Minus to Port 0
DMNS1	Inout	Data Minus to Port 1

5.3 Register Description

The Host Controller (HC) contains a set of on-chip operational registers which are mapped into a noncacheable portion of the system addressable space. These registers are used by the Host Controller Driver (HCD). According to the function of these registers, they are divided into four partitions, specifically for Control and Status, Memory Pointer, Frame Counter and Root Hub. All of the registers should be read and written as words.

Register Name	Description	RW	Reset Value	Address	Access Size
HcRevision	Control and Status group	R	0x00000010	0x13430000	32
HcControl		RW	0x00000000	0x13430004	32
HcCommandStatus		RW	0x00000000	0x13430008	32
HcInterruptStatus		RW	0x00000000	0x1343000C	32
HcInterruptEnable		RW	0x00000000	0x13430010	32
HcInterruptDisable		RW	0x00000000	0x13430014	32
HcHCCA	Memory pointer group	RW	0x00000000	0x13430018	32
HcPeriodCurrentED		R	0x00000000	0x1343001C	32
HcControlHeadED		RW	0x00000000	0x13430020	32
HcControlCurrentED		RW	0x00000000	0x13430024	32
HcBulkHeadED		RW	0x00000000	0x13430028	32
HcBulkCurrentED		RW	0x00000000	0x1343002C	32
HcDoneHead		R	0x00000000	0x13430030	32
HcFmInterval	Frame counter group	RW	0x00002EDF	0x13430034	32
HcFmRemaining		R	0x00000000	0x13430038	32
HcFmNumber		R	0x00000000	0x1343003C	32
HcPeriodicStart		RW	0x00000000	0x13430040	32
HcLSThreshold		RW	0x00000628	0x13430044	32
HcRhDescriptorA	Root hub group	R/W	0x02000902	0x13430048	32
HcRhDescriptorB		RW	0x00060000	0x1343004C	32
HcRhStatus		RW	0x00000000	0x13430050	32
HcRhPortStatus 1		RW	0x00000100	0x13430054	32
HcRhPortStatus 2		RW	0x00000100	0x13430058	32

NOTE: *Open HCI – Open Host Controller Specification for USB* for details of the each register.

5.4 Introduction

The Host Controller is the device which is located between the USB bus and the Host Controller Driver in the OpenHCI architecture. The Host Controller is charged with processing all of the Data Type lists built by the Host Controller Driver. Additionally, the USB Root Hub is attached to the Host Controller.

The main functions as following:

- **USB States**: the Host Controller Operation with respect to the possible USB Bus states.
- **Frame Management**: all aspects of managing the 1-ms USB Frame.
- **List Processing**: the main function of the Host Controller. the detailed processing of the HCD-built Data Type lists.
- **Interrupt Processing**: the interrupt events tracked by the Host Controller and how the Host Controller provides interrupts for those events.
- **Root Hub**: the Root Hub support.

6 OTG Controller

6.1 Overview

This chapter describes the USB On-The-Go (OTG) implemented in the processor.

The Universal Serial Bus (USB) supports serial data exchanges between a host computer and a variety of simultaneously accessible portable peripherals. Many of these portable devices would benefit a lot from being able to communicate to each other over the USB interface. And OTG make this possible. An OTG device can plays the role of both host and device.

Familiarity with the *Universal Serial Bus Specification*, Revision 1.1 and OTG supplement are necessary to fully understand the material contained in this section.

Features:

- Complies with the USB 2.0 standard for high-speed (480 Mbps) functions and with the *On-The-Go* supplement to the USB 2.0 specification
- Operates either as the function controller of a high- /full-speed USB peripheral or as the host/peripheral in point-to-point or multi-point communications with other USB functions
- Supports Session Request Protocol (SRP) and Host Negotiation Protocol (HNP)
- UTMI+ Level 3 Transceiver Interface
- Soft connect/disconnect
- 5 DMA channels
- Supports control, interrupt, ISO and bulk transfer

6.2 Pin Description

Table 6-1 OTG Pins Description

Name	Type	Description
DP	Inout	Data Positive
DM	Inout	Data Minus
ID	Inout	Identification
drvvbus	Out	Charge pump enable

6.3 Register Description

The OTG Controller (OTGC) contains a set of on-chip operational registers which are mapped into a noncacheable portion of the system addressable space. These registers are used by the OTG Controller Driver.

Table 6-2 OTG Registers Description

Name	Description	RW		Reset Value	Address	Access Size
		CPU	USB			
FAddr	Function address register	RW	R	8'h00	0x13440000	8
Power	Power management register	RW	RW	8'h20	0x13440001	8
IntrTx	Interrupt register for Endpoint 0 plus TX Endpoints 1 to 15	R	S	16'h0000	0x13440002	16
IntrRx	Interrupt register for Rx Endpoints 1 to 15	R	S	16'h0000	0x13440004	16
IntrTxE	Interrupt enable register for IntrTx	RW	R	16'hffff	0x13440006	16
IntrRxE	Interrupt enable register for IntrRx	RW	R	16'hfffe	0x13440008	16
IntrUSB	Interrupt register for common USB interrupts	R	S	8'h00	0x1344000a	8
IntrUSBE	Interrupt enable register for IntrUSB	RW	R	8'h06	0x1344000b	8
Frame	Frame Number	R	W	16'h0000	0x1344000c	16
Index	Index register for selecting the endpoint status and control registers	RW	R	4'h0	0x1344000e	4
TestMode	Enables the USB 2.0 test modes	RW	R	8'h00	0x1344000f	8
TxMaxP	Maximum packet size for peripheral TX endpoint (Index register set to select Endpoints 1 – 15 only)	RW	R	16'h0000	0x13440010	16
CSR0L/H	Control Status register for Endpoint 0 (Index register set to select Endpoint 0)	RW	RW	16'h0000	0x13440012	16
TxCSRL/H	Control Status register for peripheral TX endpoint (Index register set to select Endpoints 1 – 15)					
RxMaxP	Maximum packet size for	RW	R	16'h00	0x13440014	16

	peripheral Rx endpoint (Index register set to select Endpoints 1 – 15 only)			00		
RxCSRL/H	Control Status register for peripheral Rx endpoint (Index register set to select Endpoints 1 – 15 only)	RW	RW	16'h0000	0x13440016	16
Count0	Number of received bytes in Endpoint 0 FIFO (Index register set to select Endpoint 0)	R	W	7'h00	0x13440018	7
RxCount	Number of bytes to be read from peripheral Rx endpoint FIFO (Index register set to select Endpoints 1 – 15)					
Type0	Defines the speed of Endpoint 0 (Index register set to select Endpoint 0)	RW	R	8'h0	0x1344001a	8
TxType	Sets the transaction protocol, speed and peripheral endpoint number for the host TX endpoint (Index register set to select Endpoints 1 – 15)					
NakLimit0	Sets the NAK response timeout on Endpoint 0 (Index register set to select Endpoint 0)	RW	R	8'h00	0x1344001b	8
TxInterval	Sets the polling interval for Interrupt/ISOC transactions or the NAK response timeout on Bulk transactions for host TX endpoint (Index register set to select Endpoints 1 – 15 only)					
RxType	Sets the transaction protocol, speed and peripheral endpoint number for the host Rx endpoint (Index register set to select Endpoints 1 – 15 only)	RW	R	8'h00	0x1344001c	8

RxInterval	Sets the polling interval for Interrupt/ISOC transactions or the NAK response timeout on Bulk transactions for host Rx endpoint (Index register set to select Endpoints 1 – 15 only)	RW	R	8'h00	0x1344001d	8
ConfigData	Returns details of core configuration (Index register set to select Endpoint 0)	R	R	8'h??	0x1344001f	8
FifoSize	Returns the configured size of the selected Rx FIFO and TX FIFOs (Endpoints 1 – 15 only)					
FIFOx	FIFOs for Endpoints 0 – 15	RW	-	32'h??	0x13440020	32
DevCtl	OTG device control register	RW	RW	8'h80	0x13440060	8
MISC	Miscellaneous Register	RW	RW	8'h00	0x13440061	8
VControl	UTMI+ PHY Vendor registers	R	R	4'h?	0x13440068	4
Vstatus	UTMI+ PHY Vendor registers	R	R	8'h??	0x1344006a	8
HWVers	Hardware Version Number Register	R	R	16'h?? ??	0x1344006c	16
EPInfo	Information about numbers of TX and Rx endpoints	R	R	8'h??	0x13440078	8
RAMInfo	Information about the width of the RAM and the number of DMA channels	R	R	8'h??	0x13440079	8
LinkInfo	Information about delays to be applied	RW	R	8'h5c	0x1344007a	8
VPLen	Duration of the VBus pulsing charge	RW	R	8'h3c	0x1344007b	8
HS_EOF1	Time buffer available on High-Speed transactions	RW	R	8'h80	0x1344007c	8
FS_EOF1	Time buffer available on Full-Speed transactions	RW	R	8'h77	0x1344007d	8
LS_EOF1	Time buffer available on Low-Speed transactions	RW	R	8'h72	0x1344007e	8
SoftRst	Soft reset	RW	R	8'h00	0x1344007f	8
TxFuncAddr	Transmit Endpoint *n* Function Address (Host Mode only)	RW	R	7'h00	0x13440080 +8*n	7
TxHubAddr	Transmit Endpoint *n* Hub Address (Host Mode only)	RW	R	8'h00	0x13440082 +8*n	8
TxHubPort	Transmit Endpoint *n* Hub Port (Host Mode only)	RW	R	7'h00	0x13440083 +8*n	7
RxFuncAddr	Receive Endpoint *n* Function	RW	R	7'h00	0x13440084	7

	Address (Host Mode only)				+8*n	
RxHubAddr	Receive Endpoint *n* Hub Address (Host Mode only)	RW	R	8'h00	0x13440086 +8*n	8
RxHubPort	Receive Endpoint *n* Hub Port (Host Mode only)	RW	R	7'h00	0x13440087 +8*n	7
DMA_INTR	DMA Interupt register	RW	S	8'h00	0x13440200	8
DMA_CNTL	DMA Control Register for DMA channel n (channel 1 thru 8)	RW	R	11'h00	0x13440204 +(n-1)*0x10	11
DMA_ADDR	DMA Address Register for DMA channel n (channel 1 thru 8)	RW	RW	32'h00 00000 0	0x13440208 +(n-1)*0x10	32
DMA_COUNT	DMA Count Register for DMA channel n (channel 1 thru 8)	RW	RW	32'h00 00000 0	0x1344020c +(n-1)*0x10	32
RqPktCount	Number of requested packets for Receive Endpoint *n* (Endpoints 1 – 15 only)	RW	RW	16'h00 00	0x13440300 +4*n	16
RmtWkIntr	Usb remote wake up interrupt register	R	S	1'b0	0x13440398	1
RmtWkIntrE	Usb remote wake up interrupt register enable	RW	R	1'b0	0x1344039c	1
RxDPktBufDis	Double Packet Buffer Disable register for Rx Endpoints 1 to 15	RW	R	16'h00 00	0x13440340	16
TxDPktBufDis	Double Packet Buffer Disable register for TX Endpoints 1 to 15	RW	R	16'h00 00	0x13440342	16
C_T_UCH	This register sets the Chirp Timeout Timer	RW	-	16'h?	0x13440344	16
C_T_HSRTN	This register sets the delay from the end of High Speed resume signaling to enable UTM normal operating mode	RW	-	16'h?	0x13440346	16
C_T_HSBT	HS Timeout Adder	RW	R	4'h0	0x13440348	4

NOTE:

In the following bit descriptions:

'r' means that the bit is read only 'rw' means that the bit can be both read and written.

'set' means that the bit can only be written to set it 'r/set' means that the bit can be read or set but it can't be cleared.

'clear' means that the bit can only be written to clear it 'r/clear' means that the bit can be read or cleared but it can't be set.

'self-clearing' means the bit will be cleared automatically when the associated action has been executed.

6.4 Common registers

6.4.1 FAddr

FAddr is an 8-bit register that should be written with the 7-bit address of the peripheral part of the transaction.

When the MUSBHDRC is being used in Peripheral mode (DevCtl.D2=0), this register should be written with the address received through a SET_ADDRESS command, which will then be used for decoding the function address in subsequent token packets.

NOTE: Peripheral Mode Only!!

FAddr **0x13440000**

Bit	7	6	5	4	3	2	1	0
	Unused	FAddr						
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
7	-	Unused. Always 0.	R	-
6:0	FAddr	The function address.	RW	R

6.4.2 Power

Power is an 8-bit register that is used for controlling Suspend and Resume signaling, and some basic operational aspects of the MUSBHDRC.

Power **0x13440001**

Bit	7	6	5	4	3	2	1	0
	ISO Update	Soft Conn	HS Enab	HS Mode	Reset	Resume	Suspend Mode	Enable SuspendM
RST	0	0	1	0	0	0	0	0

Bits	Name	Description	CPU	USB
7	ISO Update	When set by the CPU, the MUSBHDRC will wait for an SOF token from the time TxPktRdy is set before sending the packet. If an IN token is received before an SOF token, then a zero length data packet will be sent. **NOTE:** *Only valid in Peripheral Mode. Also, this bit only affects endpoints performing Isochronous transfers.*	P: RW H: -	P: R H: -
6	Soft Conn	If Soft Connect/Disconnect feature is enabled, then the USB D+/D- lines are enabled when this bit is set by the CPU and tri-stated when this bit is cleared by the CPU. **NOTE:** *Only valid in Peripheral Mode.*	P: RW H: -	P: R H: -
5	HS Enab	When set by the CPU, the MUSBHDRC will negotiate for High-speed mode when the device is reset by the hub. If not set, the device will only operate in Full-speed mode.	P: RW H: RW	P: R H: R
4	HS Mode	When set, this read-only bit indicates High-speed mode successfully negotiated during USB reset. In Peripheral Mode, becomes valid when USB reset completes (as indicated by USB reset interrupt). In Host Mode, becomes valid when Reset bit is cleared. Remains valid for the duration of the session. **NOTE:** Allowance is made for Tiny-J signaling in determining the transfer speed to select.	P: R H: R	P: RW H: RW
3	Reset	This bit is set when Reset signaling is present on the bus. **NOTE:** This bit is Read/Write from the CPU inHost Mode but Read-Only in Peripheral Mode.	P: R H: RW	P: RW H: RW
2	Resume	Set by the CPU to generate Resume signaling when the function is in Suspend mode. The CPU should clear this bit after 10 ms (a maximum of 15 ms) to end Resume signaling. In Host mode, this bit is also automatically set when Resume signaling from the target is detected while the MUSBHDRC is suspended.	P: RW H: RW	P: R H: R
1	Suspend Mode	In Host mode, this bit is set by the CPU to enter Suspend mode. In Peripheral mode, this bit is set on entry into Suspend mode. It is cleared when the CPU reads the interrupt register, or sets the Resume bit above.	P: R H: set	P: RW H: clr
0	Enable SuspendM	Set by the CPU to enable the SUSPENDM output.	P: RW H: RW	P: R H: R

6.4.3 IntrTx

IntrTx is a 16-bit read-only register that indicates which interrupts are currently active for Endpoint 0 and the Tx Endpoints 1–15.

NOTE: Bits relating to endpoints that have not been configured will always return 0. Note also that all active interrupts are cleared when this register is read.

IntrTx **0x13440002**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EP15Tx	EP14Tx	EP13Tx	EP12Tx	EP11Tx	EP10Tx	EP9Tx	EP8Tx	EP7Tx	EP6Tx	EP5Tx	EP4Tx	EP3Tx	EP2Tx	EP1Tx	EP0
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
15	EP15Tx	Tx Endpoint 15 interrupt.	R	Set
14	EP14Tx	Tx Endpoint 14 interrupt.	R	Set
13	EP13Tx	Tx Endpoint 13 interrupt.	R	Set
12	EP12Tx	Tx Endpoint 12 interrupt.	R	Set
11	EP11Tx	Tx Endpoint 11 interrupt.	R	Set
10	EP10Tx	Tx Endpoint 10 interrupt.	R	Set
9	EP9Tx	Tx Endpoint 9 interrupt.	R	Set
8	EP8Tx	Tx Endpoint 8 interrupt.	R	Set
7	EP7Tx	Tx Endpoint 7 interrupt.	R	Set
6	EP6Tx	Tx Endpoint 6 interrupt.	R	Set
5	EP5Tx	Tx Endpoint 5 interrupt.	R	Set
4	EP4Tx	Tx Endpoint 4 interrupt.	R	Set
3	EP3Tx	Tx Endpoint 3 interrupt.	R	Set
2	EP2Tx	Tx Endpoint 2 interrupt.	R	Set
1	EP1Tx	Tx Endpoint 1 interrupt.	R	Set
0	EP0	Tx Endpoint 0 interrupt.	R	Set

6.4.4 IntrRx

IntrRx is a 16-bit read-only register that indicates which of the interrupts for Rx Endpoints 1 – 15 are currently active.

NOTE: Bits relating to endpoints that have not been configured will always return 0. Note also that all active interrupts are cleared when this register is read.

INTRRX 0x13040004

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EP15Rx	EP14Rx	EP13Rx	EP12Rx	EP11Rx	EP10Rx	EP9Rx	EP8Rx	EP7Rx	EP6Rx	EP5Rx	EP4Rx	EP3Rx	EP2Rx	EP1Rx	-
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
15	EP15Rx	Rx Endpoint 15 interrupt.	R	Set
14	EP14Rx	Rx Endpoint 14 interrupt.	R	Set
13	EP13Rx	Rx Endpoint 13 interrupt.	R	Set
12	EP12Rx	Rx Endpoint 12 interrupt.	R	Set
11	EP11Rx	Rx Endpoint 11 interrupt.	R	Set
10	EP10Rx	Rx Endpoint 10 interrupt.	R	Set
9	EP9Rx	Rx Endpoint 9 interrupt.	R	Set
8	EP8Rx	Rx Endpoint 8 interrupt.	R	Set
7	EP7Rx	Rx Endpoint 7 interrupt.	R	Set
6	EP6Rx	Rx Endpoint 6 interrupt.	R	Set
5	EP5Rx	Rx Endpoint 5 interrupt.	R	Set
4	EP4Rx	Rx Endpoint 4 interrupt.	R	Set
3	EP3Rx	Rx Endpoint 3 interrupt.	R	Set
2	EP2Rx	Rx Endpoint 2 interrupt.	R	Set
1	EP1Rx	Rx Endpoint 1 interrupt.	R	Set
0	-	Unused, always returns 0.	R	R

6.4.5 IntrTxE

IntrTxE is a 16-bit register that provides interrupt enable bits for the interrupts in IntrTx. On reset, the bits corresponding to Endpoint 0 and the Tx endpoints included in the design are set to 1, while the remaining bits are set to 0.

NOTE: Bits relating to endpoints that have not been configured will always return 0.

IntrTxE 0x13440006

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EP15Tx	EP14Tx	EP13Tx	EP12Tx	EP11Tx	EP10Tx	EP9Tx	EP8Tx	EP7Tx	EP6Tx	EP5Tx	EP4Tx	EP3Tx	EP2Tx	EP1Tx	EP0
RST	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	CPU	USB
15	EP15Tx	Tx Endpoint 15 interrupt enable.	RW	R
14	EP14Tx	Tx Endpoint 14 interrupt enable.	RW	R
13	EP13Tx	Tx Endpoint 13 interrupt enable.	RW	R
12	EP12Tx	Tx Endpoint 12 interrupt enable.	RW	R
11	EP11Tx	Tx Endpoint 11 interrupt enable.	RW	R
10	EP10Tx	Tx Endpoint 10 interrupt enable.	RW	R
9	EP9Tx	Tx Endpoint 9 interrupt enable.	RW	R
8	EP8Tx	Tx Endpoint 8 interrupt enable.	RW	R
7	EP7Tx	Tx Endpoint 7 interrupt enable.	RW	R
6	EP6Tx	Tx Endpoint 6 interrupt enable.	RW	R
5	EP5Tx	Tx Endpoint 5 interrupt enable.	RW	R
4	EP4Tx	Tx Endpoint 4 interrupt enable.	RW	R
3	EP3Tx	Tx Endpoint 3 interrupt enable.	RW	R
2	EP2Tx	Tx Endpoint 2 interrupt enable.	RW	R
1	EP1Tx	Tx Endpoint 1 interrupt enable.	RW	R
0	EP0	Tx Endpoint 0 interrupt enable.	RW	R

6.4.6 IntrRxE

IntrRxE is a 16-bit register that provides interrupt enable bits for the interrupts in IntrRx. On reset, the bits corresponding to the Rx endpoints included in the design are set to 1, while the remaining bits are set to 0.

NOTE: Bits relating to endpoints that have not been configured will always return 0.

IntrRxE **0x13440008**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EP15Rx	EP14Rx	EP13Rx	EP12Rx	EP11Rx	EP10Rx	EP9Rx	EP8Rx	EP7Rx	EP6Rx	EP5Rx	EP4Rx	EP3Rx	EP2Rx	EP1Rx	-
RST	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Bits	Name	Description	CPU	USB
15	EP15Rx	Rx Endpoint 15 interrupt enable.	RW	R
14	EP14Rx	Rx Endpoint 14 interrupt enable.	RW	R
13	EP13Rx	Rx Endpoint 13 interrupt enable.	RW	R
12	EP12Rx	Rx Endpoint 12 interrupt enable.	RW	R
11	EP11Rx	Rx Endpoint 11 interrupt enable.	RW	R
10	EP10Rx	Rx Endpoint 10 interrupt enable.	RW	R

9	EP9Rx	Rx Endpoint 9 interrupt enable.	RW	R
8	EP8Rx	Rx Endpoint 8 interrupt enable.	RW	R
7	EP7Rx	Rx Endpoint 7 interrupt enable.	RW	R
6	EP6Rx	Rx Endpoint 6 interrupt enable.	RW	R
5	EP5Rx	Rx Endpoint 5 interrupt enable.	RW	R
4	EP4Rx	Rx Endpoint 4 interrupt enable.	RW	R
3	EP3Rx	Rx Endpoint 3 interrupt enable.	RW	R
2	EP2Rx	Rx Endpoint 2 interrupt enable.	RW	R
1	EP1Rx	Rx Endpoint 1 interrupt enable.	RW	R
0	-	Unused, always returns 0.	R	R

6.4.7 IntrUSB

IntrUSB is an 8-bit read-only register that indicates which USB interrupts are currently active. All active interrupts will be cleared when this register is read.

IntrUSB **0x1344000a**

Bit	7	6	5	4	3	2	1	0
	VBus Error	Sess Req	Discon	Conn	SOF	Reset/Babble	Resume	Suspend
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
7	Vbus Error	Set when VBus drops below the VBus Valid threshold during a session. *Only valid when MUSBHDRC is 'A' device.*	R	Set
6	Sess Req	Set when Session Request signaling has been detected. *Only valid when MUSBHDRC is 'A' device.*	R	Set
5	Discon	Set in Host mode when a device disconnect is detected. Set in Peripheral mode when a session ends. *Valid at all transaction speeds.*	R	Set
4	Conn	Set when a device connection is detected. *Only valid in Host mode. Valid at all transaction speeds.*	R	Set
3	SOF	Set when a new frame starts.	R	Set
2	Reset	Set in Peripheral mode when Reset signaling is detected on the D2 e bus.	R	Set
	Babble	Set in Host mode when babble is detected.		

1	Resume	Set when Resume signaling is detected on the bus while the MUSBHDRC is in Suspend mode.	R	Set
0	Suspend	Set when Suspend signaling is detected on the bus. *Only valid in Peripheral mode.*	R	Set

6.4.8 IntrUSBE

IntrUSBE is an 8-bit register that provides interrupt enable bits for each of the interrupts in IntrUSB.

IntrUSBE **0x1344000b**

Bit	7	6	5	4	3	2	1	0
	VBus Error	Sess Req	Discon	Conn	SOF	Reset/Babble	Resume	Suspend
RST	0	0	0	0	0	1	1	0

Bits	Name	Description	CPU	USB
7	Vbus Error	Interrupt enable.	RW	R
6	Sess Req	Interrupt enable.	RW	R
5	Discon	Interrupt enable.	RW	R
4	Conn	Interrupt enable.	RW	R
3	SOF	Interrupt enable.	RW	R
2	Reset Babble	Interrupt enable.	RW	R
1	Resume	Interrupt enable.	RW	R
0	Suspend	Interrupt enable.	RW	R

6.4.9 Frame

Frame is a 16-bit read-only register that holds the last received frame number.

Frame **0x1344000C**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0					Frame Number										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
15:11	-	Always 0.	R	W
10:0	Frame Number	Frame number.	R	W

6.4.10 Index

Each Tx endpoint and each Rx endpoint have their own set of control/status registers located between 100h – 1FFh. In addition one set of Tx control/status and one set of Rx control/status registers appear at 10h – 19h. Index is a 4-bit register that determines which endpoint control/status registers are accessed.

Before accessing an endpoint's control/status registers at 10h – 19h, the endpoint number should be written to the Index register to ensure that the correct control/status registers appear in the memory map.

Index **0x1344000e**

Bit	3	2	1	0
	Selected Endpoint			
RST	0	0	0	0

Bits	Name	Description	CPU	USB
3:0	Selected Endpoint	Selected endpoint.	RW	R

6.4.11 TestMode

Testmode is an 8-bit register that is primarily used to put the MUSBHDRC into one of the four test modes for High-speed operation described in the USB 2.0 specification – in response to a SET FEATURE: TESTMODE command. It is not used in normal operation.

NOTE: Only one of Bits D0 – D6 should be set at any time.

TestMode **0x1344000f**

Bit	7	6	5	4	3	2	1	0
	Force_Host	FIFO_Access(self-clr)	Force_FS	Force_HS	Test_Packet	Test_K	Test_J	Test_SE0_NAK
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
7	Force_Host	The CPU sets this bit to instruct the core to enter Host mode when the Session bit is set, regardless of whether it is connected to any peripheral. The state of the CID input, HostDisconnect and LineState signals are ignored. The core will then remain in Host mode until the Session bit is cleared, even if a device is disconnected, and if the Force_Host bit remains set, will re-enter Host mode the next time the Session bit is set. While in this mode, the status of the HOSTDISCON signal from the PHY may be read from bit 7 of the DevCtl register. The operating speed is determined from the Force_HS and Force_FS bits as follows: Force_HS / Force_FS / Operating Speed 0 / 0 / Low Speed 0 / 1 / Full Speed 1 / 0 / High Speed 1 / 1 / *Undefined*	RW	R
6	FIFO_Access	The CPU sets this bit to transfer the packet in the Endpoint 0 Tx FIFO to the Endpoint 0 Rx FIFO. It is cleared automatically.	Set	R
5	Force_FS	The CPU sets this bit either in conjunction with bit 7 above or to force the MUSBHDRC into Full-speed mode when it receives a USB reset.	RW	R
4	Force_HS	The CPU sets this bit either in conjunction with bit 7 above or to force the MUSBHDRC into High-speed mode when it receives a USB reset.	RW	R
3	Test_Packet	*(High-speed mode)* The CPU sets this bit to enter the Test_Packet test mode. In this mode, the MUSBHDRC repetitively transmits on the bus a 53-byte test packet, the form of which is defined in the *Universal Serial Bus Specification* Revision 2.0, Section 7.1.20 (and in the MUSBHDRC Programmer's Guide). **NOTE:** The test packet has a fixed format and must be loaded into the Endpoint 0 FIFO before the test mode is entered.	RW	R

2	Test_K	*(High-speed mode)* The CPU sets this bit to enter the Test_K test mode. In this mode, the MUSBHDRC transmits a continuous K on the bus.	RW	R
1	Test_J	*(High-speed mode)* The CPU sets this bit to enter the Test_J test mode. In this mode, the MUSBHDRC transmits a continuous J on the bus.	RW	R
0	Test_SE0_NAK	*(High-speed mode)* The CPU sets this bit to enter the Test_SE0_NAK test mode. In this mode, the MUSBHDRC remains in High-speed mode but responds to any valid IN token with a NAK.	RW	R

6.4.12 DevCtl

DevCtl is an 8-bit register that is used to select whether the MUSBHDRC is operating in Peripheral mode or in Host mode, and for controlling and monitoring the USB VBus line.

DevCtl **0x13440060**

Bit	7	6	5	4	3	2	1	0
	B-Device	FSDev	LSDev	VBus[1]	VBus[0]	Host Mode	Host Req	Session
RST	1	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
7	B-Device	This Read-only bit indicates whether the MUSBHDRC is operating as the ‘A’ device or the ‘B’ device. 0 => ‘A’ device; 1 => ‘B’ device. *Only valid while a session is in progress.* **NOTE:** If the core is in Force_Host mode (i.e. a session has been started with Testmode.D7 = 1), this bit will indicate the state of the HOSTDISCON input signal from the PHY.	R	RW

<table>
<tr><td>6</td><td>FSDev</td><td>This Read-only bit is set when a full-speed or high-speed device has been detected being connected to the port. (High-speed devices are distinguished from full-speed by checking for high-speed chirps when the device is reset.) Only valid in Host mode.</td><td>R</td><td>RW</td></tr>
<tr><td>5</td><td>LSDev</td><td>This Read-only bit is set when a low-speed device has been detected being connected to the port. Only valid in Host mode.</td><td>R</td><td>RW</td></tr>
<tr><td>4:3</td><td>VBus</td><td>These Read-only bits encode the current VBus level as follows:
<table><tr><th>D4</th><th>D3</th><th>Meaning</th></tr><tr><td>0</td><td>0</td><td>Below SessionEnd</td></tr><tr><td>0</td><td>1</td><td>Above SessionEnd, below AValid</td></tr><tr><td>1</td><td>0</td><td>Above AValid, below VBusValid</td></tr><tr><td>1</td><td>1</td><td>Above VBusValid</td></tr></table></td><td>R</td><td>RW</td></tr>
<tr><td>2</td><td>Host Mode</td><td>This Read-only bit is set when the MUSBHDRC is acting as a Host.</td><td>R</td><td>RW</td></tr>
<tr><td>1</td><td>Host Req</td><td>When set, the MUSBHDRC will initiate the Host Negotiation when Suspend mode is entered. It is cleared when Host Negotiation is completed. See Section 15. (‘B’ device only)</td><td>RW</td><td>R/clr</td></tr>
<tr><td>0</td><td>Session</td><td>When operating as an ‘A’ device, this bit is set or cleared by the CPU to start or end a session. When operating as a ‘B’ device, this bit is set/cleared by the MUSBHDRC when a session starts/ends. It is also set by the CPU to initiate the Session Request Protocol. When the MUSBHDRC is in Suspend mode, the bit may be cleared by the CPU to perform a software disconnect.</td><td>RW</td><td>RW</td></tr>
</table>

6.5 Indexed Register

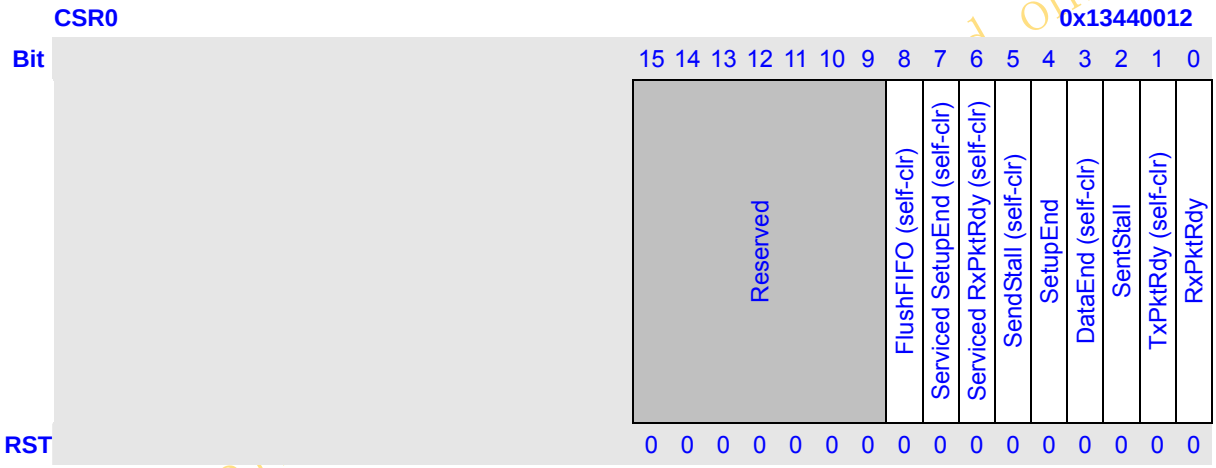
NOTE: The action of the following registers when the selected endpoint has not been configured is undefined.

6.5.1 CSR0

CSR0 is a 16-bit register that provides control and status bits for Endpoint 0.

NOTE: The interpretation of the register depends on whether the MUSBHDRC is acting as a peripheral or as a host. Users should also be aware that the value returned when the register is read reflects the status attained e.g. as a result of writing to the register. (with the Index register set to 0)

Peripheral Mode:



Bits	Name	Description	CPU	USB
15:9	Reserved	*Unused. Return 0 when read.*	R	R
8	FlushFIFO	The CPU writes a 1 to this bit to flush the next packet to be transmitted/read from the Endpoint 0 FIFO. The FIFO pointer is reset and the TxPktRdy/RxPktRdy bit (below) is cleared. **NOTE:** FlushFIFO has no effect unless TxPktRdy/RxPktRdy is set.	Set	R
7	Serviced SetupEnd	The CPU writes a 1 to this bit to clear the SetupEnd bit. It is cleared automatically.	Set	R
6	Serviced RxPKtRdy	The CPU writes a 1 to this bit to clear the RxPktRdy bit. It is cleared automatically.	Set	R

5	SendStall	The CPU writes a 1 to this bit to terminate the current transaction. The STALL handshake will be transmitted and then this bit will be cleared automatically.	Set	R
4	SetupEnd	This bit will be set when a control transaction ends before the DataEnd bit has been set. An interrupt will be generated and the FIFO flushed at this time. The bit is cleared by the CPU writing a 1 to the ServicedSetupEnd bit.	R	Set
3	DataEnd	The CPU sets this bit: 1 When setting TxPktRdy for the last data packet. 2 When clearing RxPktRdy after unloading the last data packet. 3 When setting TxPktRdy for a zero length data packet. It is cleared automatically.	Set	R
2	SentStall	This bit is set when a STALL handshake is transmitted. The CPU should clear this bit.	R/clr	Set
1	TxPKtRdy	The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is also generated at this point (if enabled).	R/set	R
0	RxPKtRdy	This bit is set when a data packet has been received. An interrupt is generated when this bit is set. The CPU clears this bit by setting the ServicedRxPktRdy bit.	R	Set

Host Mode:

CSR0 **0x13440012**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved							FlushFIFO (self-clr)	NAK Timeout	Status Pkt	ReqPkt	Error	SetupPkt	RxStall	TxPktRdy	RxPktRdy
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
15:9	Reserved	*Unused. Return 0 when read.*	R	R
8	FlushFIFO	The CPU writes a 1 to this bit to flush the next packet to be transmitted/read from the Endpoint 0 FIFO. The FIFO pointer is reset and the TxPktRdy/RxPktRdy bit (below) is cleared. **NOTE:** FlushFIFO has no effect unless TxPktRdy/RxPktRdy is set.	Set	R
7	NAK Timeout	This bit will be set when Endpoint 0 is halted following the receipt of NAK responses for longer than the time set by the NAKLimit0 register. The CPU should clear this bit to allow the endpoint to continue.	R/clr	Set
6	StatusPkt	The CPU sets this bit at the same time as the TxPktRdy or ReqPkt bit is set, to perform a status stage transaction. Setting this bit ensures that the data toggle is set to 1 so that a DATA1 packet is used for the Status Stage transaction.	RW	R
5	ReqPkt	The CPU sets this bit to request an IN transaction. It is cleared when RxPktRdy is set.	RW	RW
4	Error	This bit will be set when three attempts have been made to perform a transaction with no response from the peripheral. The CPU should clear this bit. An interrupt is generated when this bit is set.	R	Set
3	SetupPkt	The CPU sets this bit, at the same time as the TxPktRdy bit is set, to send a SETUP token instead of an OUT token for the transaction.	RW	RW
2	RxStall	This bit is set when a STALL handshake is received. The CPU should clear this bit.	R/clr	Set
1	TxPktRdy	The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when the data packet has been transmitted. An interrupt is generated (if enabled) when the bit is cleared.	R/set	Clr

0	RxPktRdy	This bit is set when a data packet has been received. An interrupt is generated (if enabled) when this bit is set. The CPU should clear this bit when the packet has been read from the FIFO.	R/clr	RW

6.5.2 Count0

Count0 is a 7-bit read-only register that indicates the number of received data bytes in the Endpoint 0 FIFO. The value returned changes as the contents of the FIFO change and is only valid while RxPktRdy (CSR0.D0) is set.

Count0 **0x13440018**

Bit	6	5	4	3	2	1	0
	Endpoint 0 Rx Count						
RST	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
6:0	Endpoint 0 Rx Count	Number of received data bytes in the endpoint 0 FIFO.	R	W

6.5.3 ConfigData

ConfigData is an 8-bit Read-Only register that returns information about the selected core configuration.

ConfigData **0x1344001f**

Bit	7	6	5	4	3	2	1	0
	MPRxE	MPTxE	BigEndian	HBRxE	HBTxE	DynFIFOSizing	SoftConE	UTMI Data Width
RST	?	?	?	?	?	?	?	?

Bits	Name	Description	CPU	USB
7	MPRxE	When set to '1', automatic amalgamation of bulk packets is selected.	R	R
6	MPTxE	When set to '1', automatic splitting of bulk packets is selected.	R	R

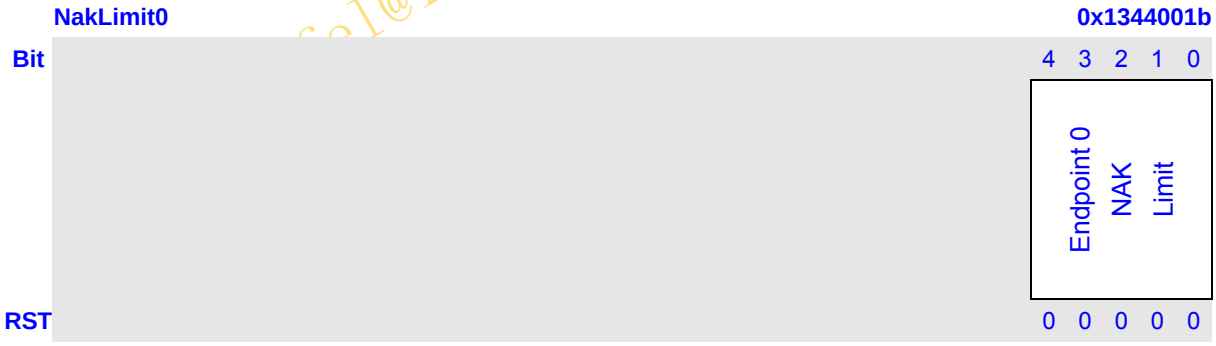
5	BigEndian	When set to '1' indicates Big Endian ordering is selected.	R	R
4	HBRxE	When set to '1' indicates High-bandwidth Rx ISO Endpoint Support selected.	R	R
3	HBTxE	When set to '1' indicates High-bandwidth Tx ISO Endpoint Support selected.	R	R
2	DynFIFOSizing	When set to '1' indicates Dynamic FIFO Sizing option selected.	R	R
1	SoftConE	When set to '1' indicates Soft Connect/Disconnect option selected.	R	R
0	UTMI Data Width	Indicates selected UTMI+ data width. 0 => 8 bits; 1 => 16 bits.	R	R

6.5.4 NakLimit0 (Host Mode Only)

NAKLimit0 is a 5-bit register that sets the number of frames/microframes (High-Speed transfers) after which Endpoint 0 should timeout on receiving a stream of NAK responses. (Equivalent settings for other endpoints can be made through their TxInterval and RxInterval registers.

The number of frames/microframes selected is $2^{(m-1)}$ (where *m* is the value set in the register, valid values 2 – 16). If the host receives NAK responses from the target for more frames than the number represented by the Limit set in this register, the endpoint will be halted.

NOTE: A value of 0 or 1 disables the NAK timeout function.



Bits	Name	Description	CPU	USB
4:0	Endpoint 0 NAK Limit	Endpoint 0 NAK limit.	RW	R

6.5.5 TxMaxP

The TxMaxP register defines the maximum amount of data that can be transferred through the selected Tx endpoint in a single operation. There is a TxMaxP register for each Tx endpoint (except Endpoint 0).

Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can

be up to 1024 bytes but is subject to the constraints placed by the USB Specification on packet sizes for Bulk, Interrupt and Isochronous transfers in Full speed and High-speed operations.

Where the option of High-bandwidth Isochronous/Interrupt endpoints or of packet splitting on Bulk endpoints has been taken when the core is configured, the register includes either 2 or 5 further bits that define a multiplier *m* which is equal to one more than the value recorded.

In the case of Bulk endpoints with the packet splitting option enabled, the multiplier *m* can be up to 32 and defines the maximum number of 'USB' packets (i.e. packets for transmission over the USB) of the specified payload into which a single data packet placed in the FIFO should be split, prior to transfer. (If the packet splitting option is not enabled, D15–D13 is not implemented and D12–D11 (if included) is ignored.) **NOTE:** The data packet is required to be an exact multiple of the payload specified by bits 10:0, which is itself required to be either 8, 16, 32, 64 or (in the case of High Speed transfers) 512 bytes.

For Isochronous/Interrupt endpoints operating in High-Speed mode and with the High-bandwidth option enabled, *m* may only be either 2 or 3 (corresponding to bit 11 set or bit 12 set, respectively) and it specifies the maximum number of such transactions that can take place in a single microframe. If either bit 11 or bit 12 is non-zero, the MUSBHDRC will automatically split any data packet written to the FIFO into up to 2 or 3 'USB' packets, each containing the specified payload (or less). The maximum payload for each transaction is 1024 bytes, so this allows up to 3072 bytes to be transmitted in each microframe. (For Isochronous/Interrupt transfers in Full-speed mode or if High-bandwidth is not enabled, bits 11 and 12 are ignored.)

The value written to bits 10:0 (multiplied by *m* in the case of high-bandwidth Isochronous/Interrupt transfers) must match the value given in the *wMaxPacketSize* field of the Standard Endpoint Descriptor for the associated endpoint (see *USB Specification* Revision 2.0, Chapter 9). A mismatch could cause unexpected results. The total amount of data represented by the value written to this register (specified payload × *m*) must not exceed the FIFO size for the Tx endpoint, and should not exceed half the FIFO size if double-buffering is required.

If this register is changed after packets have been sent from the endpoint, the Tx endpoint FIFO should be completely flushed (using the FlushFIFO bit in TxCSR) after writing the new value to this register.

TxMaxP **0x13440010**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	m-1					Maximum Payload/transaction										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
15/12:11	m-1	Multiplier.	RW	R
10:0	Maximum Payload/transaction	Maximum payload transmitted in a single transaction.	RW	R

6.5.6 TxCSR

TxCSR is a 16-bit register that provides control and status bits for transfers through the currently-selected Tx endpoint. There is a TxCSR register for each configured Tx endpoint (not including Endpoint 0).

NOTE: The interpretation of the register depends on whether the MUSBHDRC is acting as a peripheral or as a host. Users should also be aware that the value returned when the register is read reflects the status attained e.g. as a result of writing to the register.

Peripheral Mode

TxCSR **0x13440012**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	AutoSet	ISO	Mode	DMAReqEnab	FrcDataTog	DMAReqMode	Reserved		IncompTx	ClrDataTog	SentStall	SendStall	FlushFIFO(sefl-clr)	UnderRun	FIFO NotEmpty	TxPktRdy
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
15	AutoSet	If the CPU sets this bit, TxPktRdy will be automatically set when data of the maximum packet size (value in TxMaxP) is loaded into the Tx FIFO. If a packet of less than the maximum packet size is loaded, then TxPktRdy will have to be set manually. **NOTE:** *Should not be set for high-bandwidth Isochronous endpoints.*	RW	R
14	ISO	The CPU sets this bit to enable the Tx endpoint for Isochronous transfers, and clears it to enable the Tx endpoint for Bulk or Interrupt transfers. **NOTE:** *This bit only has any effect in Peripheral mode. In Host mode,* it always returns zero.	RW	R

13	Mode	The CPU sets this bit to enable the endpoint direction as Tx, and clears the bit to enable it as Rx. **NOTE:** This bit *only* has any effect where the same endpoint FIFO is used for both Tx and Rx transactions.	RW	R
12	DMAReqEnab	The CPU sets this bit to enable the DMA request for the Tx endpoint.	RW	R
11	FrcDataTog	The CPU sets this bit to force the endpoint data toggle to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This can be used by Interrupt Tx endpoints that are used to communicate rate feedback for Isochronous endpoints.	RW	R
10	DMAReqMode	The CPU sets this bit to select DMA. Request Mode 1 and clears it to select DMA. Request Mode 0. **NOTE:** This bit must not be cleared either before or in the same cycle as the above DMAReqEnab bit is cleared.	RW	R
9:8	Reserved	Unused, always return 0.	R	R
7	InCompTx	When the endpoint is being used for high-bandwidth Isochronous/Interrupt transfers, this bit is set to indicate where a large packet has been split into 2 or 3 packets for transmission but insufficient IN tokens have been received to send all the parts. **NOTE:** *In anything other than a high-bandwidth transfer, this bit will always return 0.*	R/clr	Set
6	ClrDataTog	The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.	Set	R/clr
5	SentStall	This bit is set when a STALL handshake is transmitted. The FIFO is flushed and the TxPktRdy bit is cleared (see below). The CPU should clear this bit.	R/clr	Set

4	SendStall	The CPU writes a 1 to this bit to issue a STALL handshake to an IN token. The CPU clears this bit to terminate the stall condition. **NOTE:** *This bit has no effect where the endpoint is being used for Isochronous transfers.*	RW	R
3	FlushFIFO	The CPU writes a 1 to this bit to flush the latest packet from the endpoint Tx FIFO. The FIFO pointer is reset, the TxPktRdy bit (below) is cleared and an interrupt is generated. May be set simultaneously with TxPktRdy to abort the packet that is currently being loaded into the FIFO. **NOTE:** FlushFIFO has no effect unless TxPktRdy is set. Also note that, if the FIFO is double-buffered, FlushFIFO may need to be set twice to completely clear the FIFO.	Set	R
2	UnderRun	The USB sets this bit if an IN token is received when the TxPktRdy bit not set. The CPU should clear this bit.	R/clr	Set
1	FIFONotEmpty	The USB sets this bit when there is at least 1 packet in the Tx FIFO.	R/clr	Set
0	TxPktRdy	The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is also generated at this point (if enabled). TxPktRdy is also automatically cleared (but no interrupt is generated) prior to loading a second packet into a double-buffered FIFO.	R/set	Clr

Host Mode:

TxCSR **0x13440012**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	AutoSet	Reserved	Mode	DMAReqEnab	FrcDataTog	DMAReqMode	Reserved		NAK Timeout	ClrDataTog	RxStall	Reserved	FlushFIFO (self-lcr)	Error	FIFONotEmpty	TxPktRdy
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
15	AutoSet	If the CPU sets this bit, TxPktRdy will be automatically set when a packet of the maximum packet size (TxMaxP) is loaded into the Tx FIFO. If a packet of less than the maximum packet size is loaded, then TxPktRdy will have to be set manually. **NOTE:** *Should not be set for high-bandwidth Isochronous endpoints.*	RW	R
14	Reserved	Unused, always returns zero.	RW	R
13	Mode	The CPU sets this bit to enable the endpoint direction as Tx, and clears it to enable the endpoint direction as Rx. **NOTE:** *This bit only has any effect where the same endpoint FIFO is used for both Tx and Rx transactions.*	RW	R
12	DMAReqEnab	The CPU sets this bit to enable the DMA request for the Tx endpoint.	RW	R
11	FrcDataTog	The CPU sets this bit to force the endpoint data toggle to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This can be used by Interrupt Tx endpoints that are used to communicate rate feedback for Isochronous endpoints.	RW	R
10	DMAReqMode	The CPU sets this bit to select DMA Request Mode 1 and clears it to select DMA Request Mode 0. **NOTE:** *This bit must not be cleared either before or in the same cycle as the above DMAReqEnab bit is cleared*.	RW	R
9:8	Reserved	Unused, always returns 0.	R	R
7	NAKTimeout	This bit will be set when the Tx endpoint is halted following the receipt of NAK responses for longer than the time set as the NAK Limit by the TxInterval register. The CPU should clear this bit to allow the endpoint to continue. **NOTE:** *Valid only for Bulk endpoints.*	R/clr	Set
6	ClrDataTog	The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.	Set	R/clr

5	RxStall	This bit is set when a STALL handshake is received. When this bit is set, any DMA request that is in progress is stopped, the FIFO is completely flushed and the TxPktRdy bit is cleared (see below). The CPU should clear this bit.	R/clr	Set
4	Reserved	Unused, always returns 0.	R	R
3	FlushFIFO	The CPU writes a 1 to this bit to flush the latest packet from the endpoint Tx FIFO. The FIFO pointer is reset, the TxPktRdy bit (below) is cleared and an interrupt is generated. May be set simultaneously with TxPktRdy to abort the packet that is currently being loaded into the FIFO. **NOTE:** FlushFIFO has no effect unless TxPktRdy is set. Also note that, if the FIFO is double-buffered, FlushFIFO may need to be set twice to completely clear the FIFO.	Set	R
2	Error	The USB sets this bit when 3 attempts have been made to send a packet and no handshake packet has been received. When the bit is set, an interrupt is generated, TxPktRdy is cleared and the FIFO is completely flushed. The CPU should clear this bit. *Valid only when the endpoint is operating in Bulk or Interrupt mode.*	R/clr	RW
1	FIFONotEmpty	The USB sets this bit when there is at least 1 packet in the Tx FIFO.	R/clr	Set
0	TxPktRdy	The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is also generated at this point (if enabled). TxPktRdy is also automatically cleared (but no interrupt is generated) prior to loading a second packet into a double-buffered FIFO.	R/set	Clr

6.5.7 RxMaxP

The RxMaxP register defines the maximum amount of data that can be transferred through the selected Rx endpoint in a single operation. There is a RxMaxP register for each Rx endpoint (except Endpoint 0).

Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the USB Specification on packet sizes for Bulk, Interrupt and Isochronous transfers in Full speed and High-speed operations.

Where the option of High-bandwidth Isochronous/Interrupt endpoints or of combining Bulk packets has been taken when the core is configured, the register includes either 2 or 5 further bits that define a multiplier *m* which is equal to one more than the value recorded.

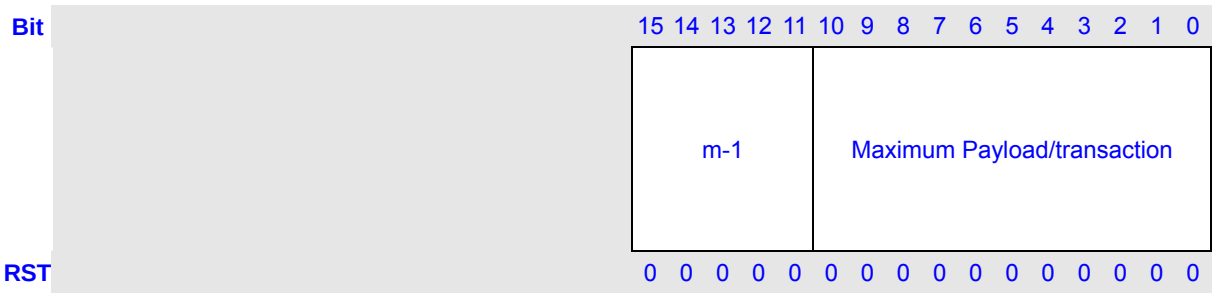
For Bulk endpoints with the packet combining option enabled, the multiplier *m* can be up to 32 and defines the number of USB packets of the specified payload which are to be combined into a single data packet within the FIFO. (If the packet splitting option is not enabled, D15–D13 is not implemented and D12–D11 (if included) is ignored.)

For Isochronous/Interrupt endpoints operating in High-Speed mode and with the High-bandwidth option enabled, *m* may only be either 2 or 3 (corresponding to bit 11 set or bit 12 set, respectively) and it specifies the maximum number of such transactions that can take place in a single microframe. If either bit 11 or bit 12 is non-zero, the MUSBHDRC will automatically combine the separate USB packets received in any microframe into a single packet within the Rx FIFO. The maximum payload for each transaction is 1024 bytes, so this allows up to 3072 bytes to be received in each microframe. (For Isochronous/Interrupt transfers in Full-speed mode or if High-bandwidth is not enabled, bits 11 and 12 are ignored.)

The value written to bits 10:0 (multiplied by *m* in the case of high-bandwidth Isochronous/Interrupt transfers) must match the value given in the *wMaxPacketSize* field of the Standard Endpoint Descriptor for the associated endpoint (see *USB Specification* Revision 2.0, Chapter 9). A mismatch could cause unexpected results.

The total amount of data represented by the value written to this register (specified payload × *m*) must not exceed the FIFO size for the OUT endpoint, and should not exceed half the FIFO size if double-buffering is required.

RxMaxP **0x13440014**



Bits	Name	Description	CPU	USB
15/12:11	m-1	Multiplier.	RW	R

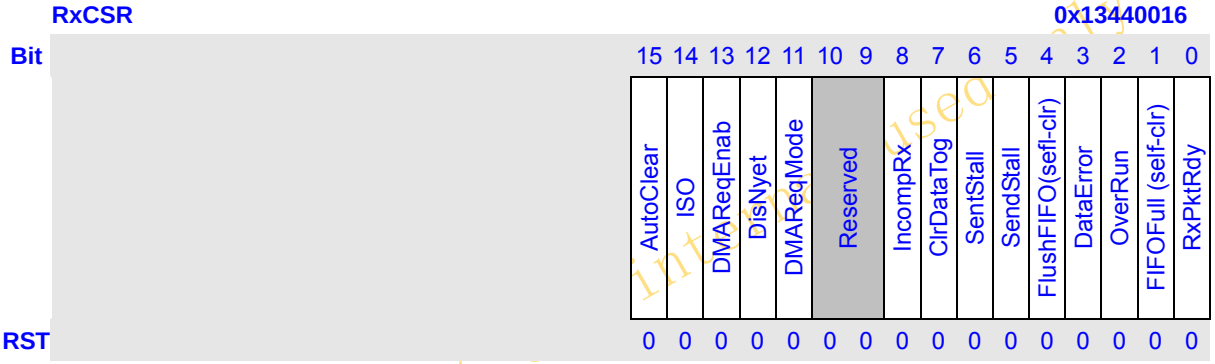
10:0	Maximum Payload/transaction	Maximum payload transmitted in a single transaction.	RW	R

6.5.8 RxCSR

RxCSR is an 16-bit register that provides control and status bits for transfers through the currently-selected Rx endpoint. There is an RxCSR register for each configured Rx endpoint (not including Endpoint 0).

NOTE: The interpretation of the register depends on whether the MUSBHDRC is acting as a peripheral or as a host. Users should also be aware that the value returned when the register is read reflects the status attained e.g. as a result of writing to the register.

Peripheral Mode:



Bits	Name	Description	CPU	USB
15	AutoClear	If the CPU sets this bit then the RxPktRdy bit will be automatically cleared when a packet of RxMaxP bytes has been unloaded from the Rx FIFO. When packets of less than the maximum packet size are unloaded, RxPktRdy will have to be cleared manually. **NOTE:** *Should not be set for highbandwidth* Isochronous endpoints.	RW	R
14	ISO	The CPU sets this bit to enable the Rx endpoint for Isochronous transfers, and clears it to enable the Rx endpoint for Bulk/Interrupt transfers.	RW	R
13	DMAReqEnab	The CPU sets this bit to enable the DMA request for the Rx endpoint.	RW	R

12	DisNyet	The CPU sets this bit to disable the sending of NYET handshakes. When set, all successfully received Rx packets are ACK'd including at the point at which the FIFO becomes full. **NOTE:** *This bit only has any effect in High-speed mode, in which mode it should be set for all Interrupt endpoints.*	RW	R
11	DMAReqMode	The CPU sets this bit to select DMA. Request Mode 1 and clears it to select DMA. Request Mode 0.	RW	R
10:9	Reserved	Unused, always return zero.	R	R
8	IncompRx	This bit is set in a high-bandwidth Isochronous transfer if the packet in the Rx FIFO is incomplete because parts of the data were not received. It is cleared when RxPktRdy is cleared. **NOTE:** *In anything* other than a high-bandwidth Isochronous transfer, this bit will always return 0.	R	Set
7	ClrDataTog	The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.	R	R/clr
6	SentStall	This bit is set when a STALL handshake is transmitted. The CPU should clear this bit.	R/clr	Set
5	SendStall	The CPU writes a 1 to this bit to issue a STALL handshake. The CPU clears this bit to terminate the stall condition. **NOTE:** *This bit has no effect where the endpoint is being used for Isochronous transfers.*	RW	R
4	FlushFIFO	The CPU writes a 1 to this bit to flush the next packet to be read from the endpoint Rx FIFO. The FIFO pointer is reset and the RxPktRdy bit (below) is cleared. **NOTE:** FlushFIFO has no effect unless RxPktRdy is set. Also note that, if the FIFO is double-buffered, FlushFIFO may need to be set twice to completely clear the FIFO.	Set	R

3	DataError	This bit is set when RxPktRdy is set if the data packet has a CRC or bit-stuff error. It is cleared when RxPktRdy is cleared. **NOTE:** *This bit is only valid when the endpoint is operating in ISO mode. In Bulk mode, it always returns zero.*	R	Set
2	OverRun	This bit is set if an OUT packet cannot be loaded into the Rx FIFO. The CPU should clear this bit. **NOTE:** *This bit is only valid when the endpoint is operating in ISO mode. In Bulk mode, it always returns zero.*	R/clr	Set
1	FIFOFull	This bit is set when no more packets can be loaded into the Rx FIFO.	R	Set
0	RxPktRdy	This bit is set when a data packet has been received. The CPU should clear this bit when the packet has been unloaded from the Rx FIFO. An interrupt is generated when the bit is set.	R/clr	Set

Host Mode:

RxCSR **0x13440016**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	AutoClear	AutoReq	DMAReqEnab	DisNyet	DMAReqMode	Reserved		IncompRx	ClrDataTog	RxStall	ReqPkt	FlushFIFO(sefl-clr)	DataError/NAKTimeout	Error	FIFOFull (self-clr)	RxPktRdy
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
15	AutoClear	If the CPU sets this bit then the RxPktRdy bit will be automatically cleared when a packet of RxMaxP bytes has been unloaded from the Rx FIFO. When packets of less than the maximum packet size are unloaded, RxPktRdy will have to be cleared manually. **NOTE:** *Should not be set for highbandwidth Isochronous endpoints.*	RW	R

14	AutoReq	If the CPU sets this bit, the ReqPkt bit will be automatically set when the RxPktRdy bit is cleared.	RW	R
13	DMAReqEnab	The CPU sets this bit to enable the DMA request for the Rx endpoint.	RW	R
12	DisNyet	The CPU sets this bit to disable the sending of NYET handshakes. When set, all successfully received Rx packets are ACK'd including at the point at which the FIFO becomes full. **NOTE:** *This bit only has any effect in High-speed mode, in which mode it should be set for all Interrupt endpoints.*	RW	R
11	DMAReqMode	The CPU sets this bit to select DMA. Request Mode 1 and clears it to select DMA. Request Mode 0.	RW	R
10:9	Reserved	Unused, always return 0.	R	R
8	IncompRx	This bit will be set in a high-bandwidth Isochronous transfer if the packet received is incomplete. It will be cleared when RxPktRdy is cleared. **NOTE:** *If USB protocols are followed correctly, this bit should never be set. The bit becoming set indicates a failure of the associated Peripheral device to behave correctly. (In anything other than a high-bandwidth Isochronous transfer, this bit will always return 0.)*	R	Set
7	ClrDataTog	The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.	Set	R/clr
6	RxStall	When a STALL handshake is received, this bit is set and an interrupt is generated. The CPU should clear this bit.	R/clr	Set
5	ReqPkt	The CPU writes a 1 to this bit to request an IN transaction. It is cleared when RxPktRdy is set.	RW	RW

4	FlushFIFO	The CPU writes a 1 to this bit to flush the next packet to be read from the endpoint Rx FIFO. The FIFO pointer is reset and the RxPktRdy bit (below) is cleared. **NOTE:** FlushFIFO has no effect unless RxPktRdy is set. Also note that, if the FIFO is double-buffered, FlushFIFO may need to be set twice to completely clear the FIFO.	Set	R
3	DataError/NAKTimeout	When operating in ISO mode, this bit is set when RxPktRdy is set if the data packet has a CRC or bit-stuff error and cleared when RxPktRdy is cleared. In Bulk mode, this bit will be set when the Rx endpoint is halted following the receipt of NAK responses for longer than the time set as the NAK Limit by the RxInterval register. The CPU should clear this bit to allow the endpoint to continue.	R/clr	Set
2	Error	The USB sets this bit when 3 attempts have been made to receive a packet and no data packet has been received. The CPU should clear this bit. An interrupt is generated when the bit is set. **NOTE:** *This bit is only valid when the Tx endpoint is operating in Bulk or Interrupt mode. In ISO mode, it always returns zero.*	R/clr	Set
1	FIFOFull	This bit is set when no more packets can be loaded into the Rx FIFO.	R	Set
0	RxPktRdy	This bit is set when a data packet has been received. The CPU should clear this bit when the packet has been unloaded from the Rx FIFO. An interrupt is generated when the bit is set.	R/clr	Set

6.5.9 RxCount

RxCount is a 13-bit read-only register that holds the number of received data bytes in the packet in the Rx FIFO.

NOTE: The value returned changes as the FIFO is unloaded and is only valid while RxPktRdy (RxCSR.D0) is set.

RxCount 0x13440018

Bit	12	11	10	9	8	7	6	5	4	3	2	1	0
	Endpoint Rx Count												
RST	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
12:0	Endpoint Rx Count	Number of received data bytes in the packet in the Rx FIFO.	R	W

6.5.10 TxType (Host Mode Only)

TxType is a 6-bit register that should be written with the endpoint number to be targeted by the endpoint in the lower 4 bits, and the transaction protocol to use for the currently-selected Tx endpoint in the upper 2 bits. There is a TxType register for each configured Tx endpoint (except Endpoint 0).

TxType 0x1344001a

Bit	5	4	3	2	1	0
	Protocol		Target Endpoint Number			
RST	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
5:4	Protocol	The CPU should set this to select the required protocol for the Tx endpoint. 00: Illegal 01: Isochronous 10: Bulk 11: Interrupt	RW	R
3:0	Target Endpoint Number	The CPU should set this value to the endpoint number contained in the Tx endpoint descriptor returned to the MUSBHDRC during device enumeration.	RW	R

6.5.11 TxInterval (Host Mode Only)

TxInterval is an 8-bit register that, for Interrupt and Isochronous transfers, defines the polling interval for the currently-selected Tx endpoint. For Bulk endpoints, this register sets the number of frames/microframes after which the endpoint should timeout on receiving a stream of NAK responses.

There is a TxInterval register for each configured Tx endpoint (except Endpoint 0).

In each case the value that is set defines a number of frames/microframes (High Speed transfers), as follows:

Transfer Type	Speed	Valid Values (m)	Interpretation
Interrupt	LS or FS	1-255	Polling interval is *m* frames
	HS	1-16	Polling interval is $2_{(m-1)}$ microframes
Isochronous	FS or HS	1-16	Polling interval is $2_{(m-1)}$ frames/microframes
Bulk	FS or HS	2-16	NAK Limit is $2_{(m-1)}$ frames/microframes **NOTE:** A value of 0 or 1 disables the NAK timeout function.

TxInterval **0x1344001b**

Bit	7	6	5	4	3	2	1	0
	Tx Polling Interval/ NAK Limit (m)							
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
7:0	Tx Polling Interval/NAK Limit	For interrupt and isochronous transfers, defines the polling interval for the currently selected Tx endpoint. For Bulk endpoints, this register sets the number of frames/microframes after which the endpoint should timeout on receiving a stream of NAK responses.	RW	R

6.5.12 RxType (Host Mode Only)

RxType is a 6-bit register that should be written with the endpoint number to be targeted by the endpoint in the lower 4 bits, and the transaction protocol to use for the currently-selected Rx endpoint in the upper 2 bits. There is an RxType register for each configured Rx endpoint (except Endpoint 0).

RxType **0x1344001c**

Bit	5	4	3	2	1	0
	Protocol		Target Endpoint Number			
RST	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
5:4	Protocol	The CPU should set this to select the required protocol for the Rx endpoint. 00: Illegal 01: Isochronous 10: Bulk 11: Interrupt	RW	R
3:0	Target Endpoint Number	The CPU should set this value to the endpoint number contained in the Rx endpoint descriptor returned to the MUSBHDRC during device enumeration.	RW	R

6.5.13 RxInterval

RxInterval is an 8-bit register that, for Interrupt and Isochronous transfers, defines the polling interval for the currently-selected Rx endpoint. For Bulk endpoints, this register sets the number of frames/microframes after which the endpoint should timeout on receiving a stream of NAK responses. There is a RxInterval register for each configured Rx endpoint (except Endpoint 0). In each case the value that is set defines a number of frames/microframes (High Speed transfers), as follows:

Transfer Type	Speed	Valid Values (m)	Interpretation
Interrupt	LS or FS	1-255	Polling interval is *m* frames
	HS	1-16	Polling interval is $2^{(m-1)}$ microframes
Isochronous	FS or HS	1-16	Polling interval is $2^{(m-1)}$ frames/microframes
Bulk	FS or HS	2-16	NAK Limit is $2^{(m-1)}$ frames/microframes **NOTE:** A value of 0 or 1 disables the NAK timeout function.

RxInterval **0x1344001d**

Bit	7	6	5	4	3	2	1	0
	Rx Polling Interval/ NAK Limit (m)							
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
7:0	Rx Polling Interval/NAK Limit	For interrupt and isochronous transfers, defines the polling interval for the currently selected Tx endpoint. For Bulk endpoints, this register sets the number of frames/microframes after which the endpoint should timeout on receiving a stream of NAK responses.	RW	R

6.5.14 FifoSize

FIFOSize is an 8-bit Read-Only register that returns the sizes of the FIFOs associated with the selected additional Tx/Rx endpoints. The lower nibble encodes the size of the selected Tx endpoint FIFO; the upper nibble encodes the size of the selected Rx endpoint FIFO. Values of 3 – 13 correspond to a FIFO size of 2^n bytes (8 – 8192 bytes). If an endpoint has not been configured, a value of 0 will be displayed. Where the Tx and Rx endpoints share the same FIFO, the Rx FIFO size will be encoded as 0xF.

NOTE: The register only has this interpretation when the Index register is set to select one of Endpoints 1 – 15 and Dynamic Sizing is not selected. It has a special interpretation when the Index register is set to select Endpoint 0 (see Section 5.3.3), while the result returned is not valid where Dynamic FIFO sizing is used. (Index register set to select Endpoints 1 – 15 only)

FifoSize **0x1344001f**

Bit	7	6	5	4	3	2	1	0
	Rx FIFO Size				Tx FIFO Size			
RST	?	?	?	?	?	?	?	?

Bits	Name	Description	CPU	USB
7:4	Rx FIFO Size	the sizes of the FIFOs associated with the selected additional Rx endpoints.	R	R
3:0	Tx FIFO Size	the sizes of the FIFOs associated with the selected additional Tx endpoints.	R	R

6.5.15 FIFOx

This address range provides 16 addresses for CPU access to the FIFOs for each endpoint. Writing to these addresses loads data into the TxFIFO for the corresponding endpoint. Reading from these addresses unloads data from the RxFIFO for the corresponding endpoint.

The address range is 20h – 5Fh and the FIFOs are located on 32-bit double-word boundaries (Endpoint 0 at 20h, Endpoint 1 at 24h ... Endpoint 15 at 5Ch).

6.6 Additional Multipoint Control / Status Registers

The following subsections detail additional control and status registers that are only valid when the multipoint option is enabled in the configuration GUI. If the multipoint option is not enabled these reregisters should not be accessed.

6.6.1 TxFuncAddr / RxFuncAddr

NOTE: REQUIRED IN HOST MODE!

TxFuncAddr and RxFuncAddr are 7-bit read/write registers that record the address of the target function that is to be accessed through the associated endpoint (EP*n*). TxFuncAddr needs to be defined for each TX endpoint that is used; RxFuncAddr needs to be defined for each Rx endpoint that is used.

NOTE: TxFuncAddr must be defined for Endpoint 0. The RxFuncAddr register does not exist on EP0.

TxFuncAddr/RxFuncAddr **0x13440080+8*n / 0x13440084+8*n**

Bit	6	5	4	3	2	1	0
	Addr of Target Function						
RST	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
6:0	Addr of Target Function	read/write registers that record the address of the target function that is to be accessed through the associated endpoint.	RW	R

6.6.2 TxHubAddr/RxHubAddr

NOTE: RELEVANT IN HOST MODE ONLY!

TxHubAddr and RxHubAddr are 8-bit read/write registers which, like TxHubPort and RxHubPort, only need to be written where a full- or low-speed device is connected to TX/Rx Endpoint EP*n* via a high-speed USB 2.0 hub which carries out the necessary transaction translation to convert between high-speed transmission and full-/low-speed transmission. In such circumstances:

- the lower 7 bits should record the address of this USB 2.0 hub.
- the top bit should record whether the hub has multiple transaction translators (set to '0' if single transaction translator; set to '1' if multiple transaction translators).

NOTE: If Endpoint 0 is connected to a hub, then TxHubAddr must be defined for EP0. The RxHubAddr register does not exist on EP0.

TxHubAddr/RxHubAddr **0x13440082+8*n / 0x13440086+8*n**

Bit	7	6	5	4	3	2	1	0
	Multi-Translators	HubAddr						
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
7	Multi-Translators	0: single transaction translator 1: multiple transaction translators	RW	R
6:0	HubAddr	Hub Address.	RW	R

6.6.3 TxHubPort / RxHubPort

NOTE: RELEVANT IN HOST MODE ONLY!

TxHubPort and RxHubPort only need to be written where a full- or low-speed device is connected to TX/Rx Endpoint EP*n* via a high-speed USB 2.0 hub which carries out the necessary transaction translation. In such circumstances, these 7-bit read/write registers need to be used to record the port of that USB 2.0 hub through which the target associated with the endpoint is accessed.

NOTE: If Endpoint 0 is connected to a hub, then TxHubPort must be defined. The RxHubPort register does not exist on EP0.

TxHubPort/RxHubPort **0x13440083+8*n / 0x13440087+8*n**

Bit	6	5	4	3	2	1	0
	Hub Port						
RST	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
6:0	Hub Port	record the port of that USB 2.0 hub through which the target associated with the endpoint is accessed.	RW	R

6.7 Additional Control/Status Registers

6.7.1 VControl

NOTE: WRITE ONLY!

VControl is a UTMI+ PHY Vendor register that may optionally be included in the core when the core is configured. Its size is also configurable and may be up to 32 bits. The structure of the register is up to the system designer, though users should note that the UTMI+ specification defines a 4-bit VControl register.

Vcontrol **0x13440068**

Bit	3	2	1	0
	Vcontrol			
RST	?	?	?	?

Bits	Name	Description	CPU	USB
3:0	VControl	UTMI+ PHY Vendor registers.	RW	-

6.7.2 VStatus

NOTE: READ ONLY!

VStatus is a UTMI+ PHY Vendor register that may optionally be included in the core when the core is configured. Its size is also configurable and may be up to 32 bits. The structure of the register is up to the system designer, though users should note that the UTMI+ specification defines an 8-bit VStatus register.

VStatus **0x1344006b**

Bit	7	6	5	4	3	2	1	0
	VStatus							
RST	?	?	?	?	?	?	?	?

Bits	Name	Description	CPU	USB
7:0	VStatus	UTMI+ PHY Vendor registers.	RW	-

6.7.3 Hwvers

HWVers register is a 16-bit read-only register that returns information about the version of the RTL from which the core hardware was generated, in particular the RTL version number (v*xx.yyy*).

Hwvers **0x1344006c**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RC	xx					yyy									
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	CPU	USB
15	RC	Set to '1' if RTL used from a Release Candidate rather than from a full release of the core.	R	R
14:10	xx	Major Version Number. (Range 0 – 31)	R	R
9:0	yyy	Minor Version Number. (Range 0 – 999)	R	R

6.8 Additional Configuration Registers

6.8.1 EPInfo

This 8-bit read-only register allows read-back of the number of TX and Rx endpoints included in the design.

EPInfo **0x13440078**

	7	6	5	4	3	2	1	0
	RxEdps				TxEdps			
RST	?	?	?	?	?	?	?	?

Bits	Name	Description	CPU	USB
7:4	RxEdps	The number of Rx endpoints implemented in the design.	R	R
3:0	TxEdps	The number of TX endpoints implemented in the design.	R	R

6.8.2 RAMInfo

This 8-bit read-only register provides information about the width of the RAM.

RAMInfo **0x13440079**

Bit	7	6	5	4	3	2	1	0
	DMAChans				RAMBits			
RST	?	?	?	?	?	?	?	?

Bits	Name	Description	CPU	USB
7:4	DMAChans	The number of DMA channels implemented in the design.	R	R
3:0	RAMBits	The width of the RAM address bus.	R	R

6.8.3 LinkInfo

This 8-bit register allows some delays to be specified.

LinkInfo								0x1344007a
Bit	7	6	5	4	3	2	1	0
	WTCon				WTID			
	0	1	0	1	1	1	0	0

Bits	Name	Description	CPU	USB
7:4	WTCon	Sets the wait to be applied to allow for the user's connect/disconnect filter in units of 533.3ns. (The default setting corresponds to 2.667μs)	RW	R
3:0	WTID	Sets the delay to be applied from IDPULLUP being asserted to IDDIG being considered valid in units of 4.369ms. (The default setting corresponds to 52.43ms)	RW	R

6.8.4 VPLen

This 8-bit register sets the duration of the VBus pulsing charge.

VPlen								0x1344007b
Bit	7	6	5	4	3	2	1	0
	VPLen							
RST	0	0	1	1	1	1	0	0

Bits	Name	Description	CPU	USB
7:0	VPLen	Sets the duration of the VBus pulsing charge in units of 546.1 μs. (The default setting corresponds to 32.77ms)	RW	R

6.8.5 HS_EOF1

This 8-bit register sets the minimum time gap that is to be allowed between the start of the last transaction and the EOF for High-speed transactions.

HS_EOF1								0x1344007c
Bit	7	6	5	4	3	2	1	0
	HS_EOF1							
RST	1	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
7:0	HS_EOF1	Sets for High-speed transactions the time before EOF to stop beginning new transactions, in units of 133.3ns. (The default setting corresponds to 17.07μs)	RW	R

6.8.6 FS_EOF1

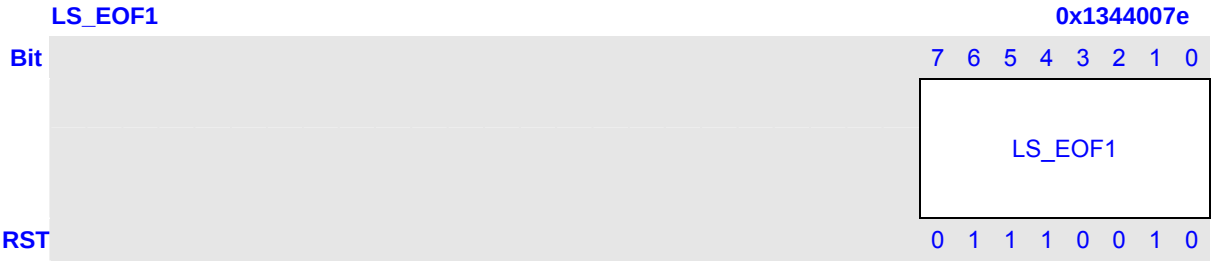
This 8-bit register sets the minimum time gap that is to be allowed between the start of the last transaction and the EOF for Full-speed transactions.

FS_EOF1								0x1344007d
Bit	7	6	5	4	3	2	1	0
	FS_EOF1							
RST	0	1	1	1	0	1	1	1

Bits	Name	Description	CPU	USB
7:0	FS_EOF1	Sets for Full-speed transactions the time before EOF to stop beginning new transactions, in units of 533.3ns. (The default setting corresponds to 63.46μs)	RW	R

6.8.7 LS_EOF1

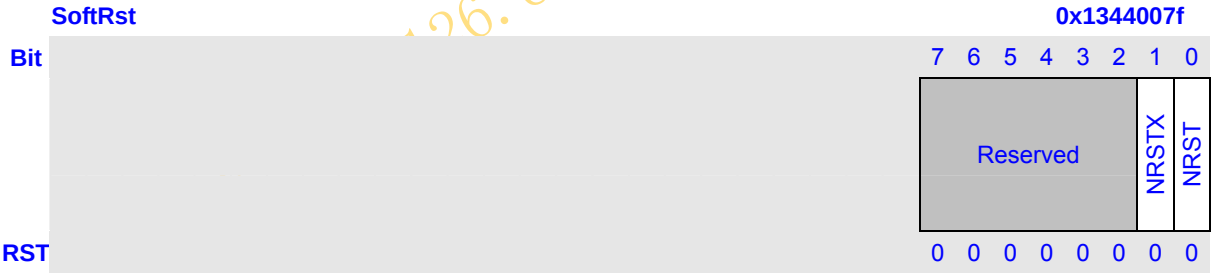
This 8-bit register sets the minimum time gap that is to be allowed between the start of the last transaction and the EOF for Low-speed transactions.



Bits	Name	Description	CPU	USB
7:0	LS_EOF1	Sets for Low-speed transactions the time before EOF to stop beginning new transactions, in units of 1.067μs. (The default setting corresponds to 121.6μs)	RW	R

6.8.8 SoftRst

This 8-bit register will assert LOW the output reset signals NRSTO and NRSTOX. This register is self clearing and will be reset by the input NRST.



Bits	Name	Description	CPU	USB
7:2	reserved	Unused, always returns zero.	-	-
1	NRSTX	The default value of this bit is 1'b0; When a 1 is written to this bit, the output NRSTXO will be asserted (LOW) within a minimum delay of 7 cycles of the CLK input. The output NRSTXO will be asynchronously asserted and synchronously de-asserted with respect to XCLK. This register is self clearing and will be reset by the input NRST.	RW	R

0	NRST	The default value of this bit is 1'b0; When a 1 is written to this bit, the output NRSTO will be asserted (LOW) within a minimum delay of 7 cycles of the CLK input. The output NRSTO will be asynchronously asserted and synchronously de-asserted with respect to CLK. This register is self clearing and will be reset by the input NRST.	RW	R

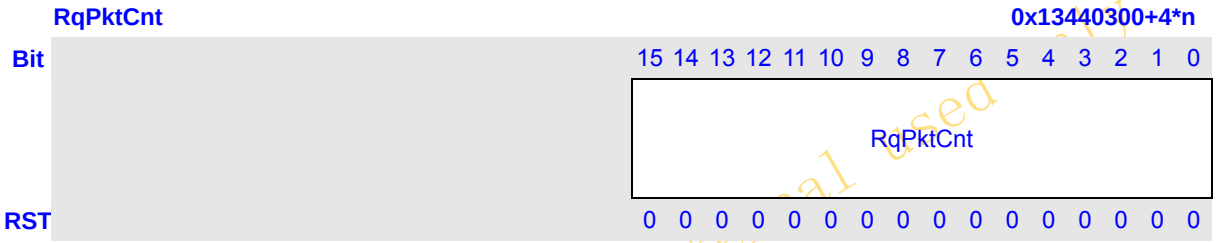
6.9 Extended Registers

6.9.1 RqPktCnt

NOTE: HOST MODE ONLY!

For each Rx Endpoint 1 – 15, the MUSBMHDRC provides a 16-bit RqPktCount register. This read/write register is used in Host mode to specify the number of packets that are to be transferred in a block transfer of one or more Bulk packets of length MaxP to Rx Endpoint *n*. The core uses the value recorded in this register to determine the number of requests to issue where the AutoReq option (included in the RxCSR register) has been set.

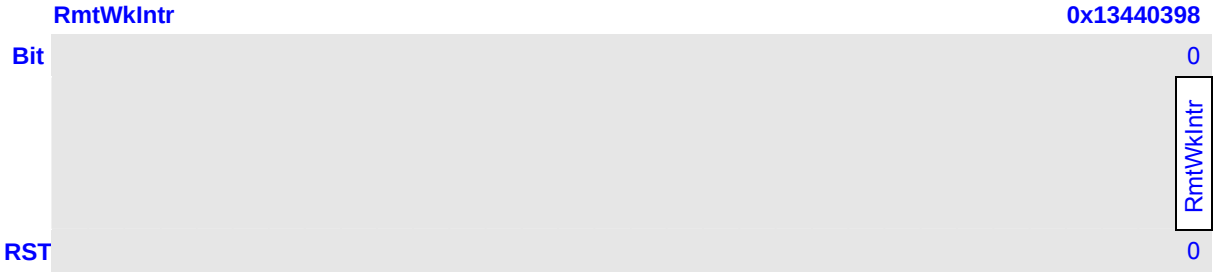
NOTE: Multiple packets combined into a single bulk packet within the FIFO count as one packet.



Bits	Name	Description	CPU	USB
15:0	RqPktCnt	Sets the number of packets of size MaxP that are to be transferred in a block transfer. *Only used in Host mode when AutoReq is set. Has no effect in Peripheral mode or when AutoReq is not set.*	RW	RW

6.9.2 RmtWkIntr

Usb remote wakeup interrupt. It will be generated when there's non idle signaling on the bus during system hibernate.

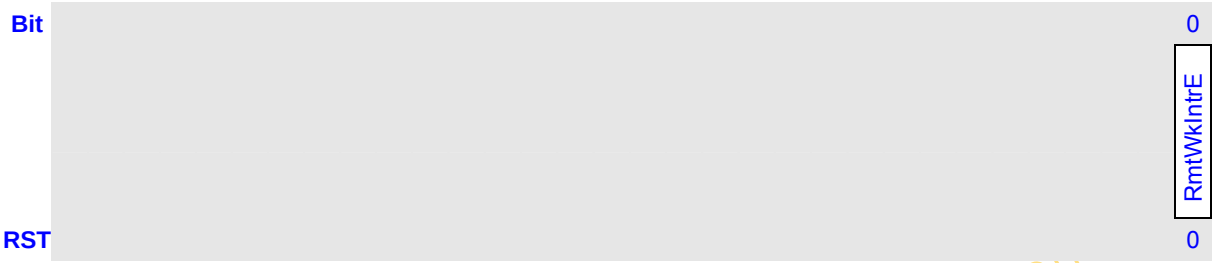


Bits	Name	Description	CPU	USB
0	RmtWkIntr	Usb remote wakeup interrupt. Self cleared after read.	RW	S

6.9.3 RmtWkIntrE

Usb remote wakeup interrupt enable.

RmtWkIntrE 0x1344039C



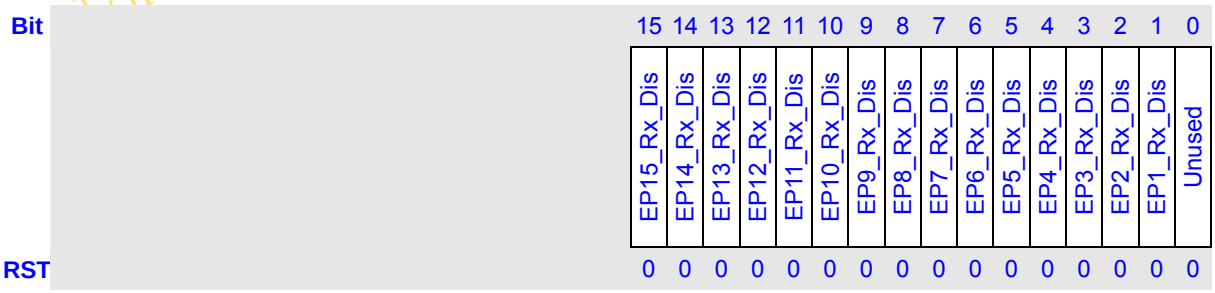
Bits	Name	Description	CPU	USB
0	RmtWkIntrE	Usb remote wakeup interrupt enable. High active.	RW	R

6.9.4 RxDPktBufDis

Rx DPktBufDis is a 16-bit register that indicates which of the Rx endpoints have disabled the double packet buffer functionality described in section 8.4.2.2 of the MUSBMHDRC Product Specification.

NOTE: Bits relating to endpoints that have not been configured may be asserted by writing a '1' their respective register, however the disable bit will have no observable effect.

RxDPktBufDis 0x13440340



Bits	Name	Description	CPU	USB
15	EP15_Rx_Dis	Rx Double Packet Buffer Disable for Endpoint 15.	RW	R
14	EP14_Rx_Dis	Rx Double Packet Buffer Disable for Endpoint 14.	RW	R
13	EP13_Rx_Dis	Rx Double Packet Buffer Disable for Endpoint 13.	RW	R
12	EP12_Rx_Dis	Rx Double Packet Buffer Disable for Endpoint 12.	RW	R
11	EP11_Rx_Dis	Rx Double Packet Buffer Disable for Endpoint 11.	RW	R

10	EP10_Rx_Dis	Rx Double Packet Buffer Disable for Endpoint 10.	RW	R
9	EP9_Rx_Dis	Rx Double Packet Buffer Disable for Endpoint 9.	RW	R
8	EP8_Rx_Dis	Rx Double Packet Buffer Disable for Endpoint 8.	RW	R
7	EP7_Rx_Dis	Rx Double Packet Buffer Disable for Endpoint 7.	RW	R
6	EP6_Rx_Dis	Rx Double Packet Buffer Disable for Endpoint 6.	RW	R
5	EP5_Rx_Dis	Rx Double Packet Buffer Disable for Endpoint 5.	RW	R
4	EP4_Rx_Dis	Rx Double Packet Buffer Disable for Endpoint 4.	RW	R
3	EP3_Rx_Dis	Rx Double Packet Buffer Disable for Endpoint 3.	RW	R
2	EP2_Rx_Dis	Rx Double Packet Buffer Disable for Endpoint 2.	RW	R
1	EP1_Rx_Dis	Rx Double Packet Buffer Disable for Endpoint 1.	RW	R
0	Unused	Reserved.	R	R

6.9.5 TxDPktBufDis

Tx DPktBufDis is a 16-bit register that indicates which of the TX endpoints have disabled the double packet buffer functionality described in section 8.4.1.2 of the MUSBMHDRC Product Specification.

NOTE: Bits relating to endpoints that have not been configured may be asserted by writing a '1' their respective register; however the disable bit will have no observable effect.

TxDPktBufDis **0x13440342**

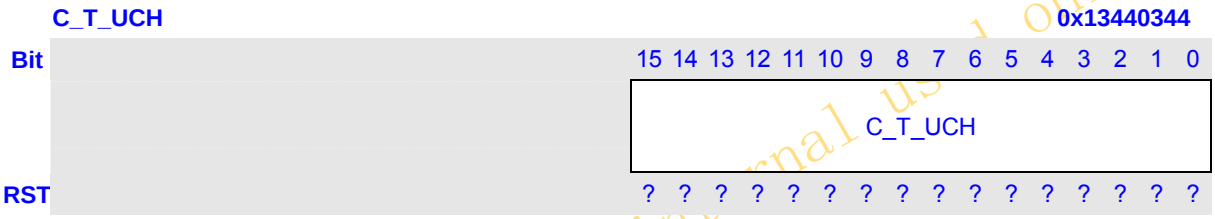
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EP15_Tx_Dis	EP14_Tx_Dis	EP13_Tx_Dis	EP12_Tx_Dis	EP11_Tx_Dis	EP10_Tx_Dis	EP9_Tx_Dis	EP8_Tx_Dis	EP7_Tx_Dis	EP6_Tx_Dis	EP5_Tx_Dis	EP4_Tx_Dis	EP3_Tx_Dis	EP2_Tx_Dis	EP1_Tx_Dis	Unused
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
15	EP15_Tx_Dis	Tx Double Packet Buffer Disable for Endpoint 15.	RW	R
14	EP14_Tx_Dis	Tx Double Packet Buffer Disable for Endpoint 14.	RW	R
13	EP13_Tx_Dis	Tx Double Packet Buffer Disable for Endpoint 13.	RW	R
12	EP12_Tx_Dis	Tx Double Packet Buffer Disable for Endpoint 12.	RW	R
11	EP11_Tx_Dis	Tx Double Packet Buffer Disable for Endpoint 11.	RW	R
10	EP10_Tx_Dis	Tx Double Packet Buffer Disable for Endpoint 10.	RW	R
9	EP9_Tx_Dis	Tx Double Packet Buffer Disable for Endpoint 9.	RW	R
8	EP8_Tx_Dis	Tx Double Packet Buffer Disable for Endpoint 8.	RW	R
7	EP7_Tx_Dis	Tx Double Packet Buffer Disable for Endpoint 7.	RW	R
6	EP6_Tx_Dis	Tx Double Packet Buffer Disable for Endpoint 6.	RW	R
5	EP5_Tx_Dis	Tx Double Packet Buffer Disable for Endpoint 5.	RW	R
4	EP4_Tx_Dis	Tx Double Packet Buffer Disable for Endpoint 4.	RW	R

3	EP3_Tx_Dis	Tx Double Packet Buffer Disable for Endpoint 3.	RW	R
2	EP2_Tx_Dis	Tx Double Packet Buffer Disable for Endpoint 2.	RW	R
1	EP1_Tx_Dis	Tx Double Packet Buffer Disable for Endpoint 1.	RW	R
0	Unused	Reserved.	R	R

6.9.6 C_T_UCH

This register sets the chirp timeout. This number when multiplied by 4 represents the number of XCLK cycles before the timeout occurs. That is, if XCLK is 30MHz, this number represents the number of 133ns time intervals before the timeout occurs. If XCLK is 60MHz, this number represents the number of 67ns time intervals before the timeout occurs. Although this bit is written by the host in the CLK domain, the counter that utilizes this value is in the XCLK domain. No time domain crossing is provided as the value in this register is a static. The default value is the value of the compiler directive of the same name located in the configuration file musbmhdrc_cfg.v.



Bits	Name	Description	CPU	USB
15:0	C_T_UCH	Configurable Chirp Timeout timer; The default value is determined by compiler directive in musbhsfc_xcfg.v file. The default value is 203Ah if the host PHY data width is 16 bits (XCLK is 30MHz) and 4074h if the PHY data width is 8 bits (XCLK is 60Mhz) corresponding to a delay of 1.1ms.	RW	-

6.9.7 C_T_HHSRTN

This register sets the delay from the end of High Speed resume signaling (acting as a Host) to enable the UTM normal operating mode. This number when multiplied by 4 represents the number of XCLK cycles before the timeout occurs. That is, if XCLK is 30MHz, this number represents the number of 33.3ns time intervals before the timeout occurs. If XCLK is 60MHz, this number represents the number of 16.7ns time intervals before the timeout occurs. Although this bit is written by the host in the CLK domain, the counter that utilizes this value is in the XCLK domain. No time domain crossing is provided as the value in this register is a static. The default value is the value of the compiler directive of the same name located in the configuration file musbmhdrc_cfg.v.

C_T_HHSRTN 0x13440346

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	C_T_HHSRTN															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	CPU	USB
15:0	C_T_HHSRTN	The delay from the end of High Speed resume signaling to enabling UTM normal operating mode. The default value is determined by compiler directive in musbhsfc_xcfg.v file. The default value is 2F3h if the host PHY data width is 16 bits (XCLK is 30MHz) and 5E6h if the PHY data width is 8 bits (XCLK is 60Mhz) corresponding to a delay of 100us.	RW	-

6.9.8 C_T_HSBT

Per USB 2.0, Section 7.1.19.2, a high-speed host or device expecting a response to a transmission must not timeout the transaction if the interpacket delay is less than 736 bit times, and it must timeout the transaction if no signaling is seen within 816 bit times. This register represents the value to be added to the mimumum high speed timeout period of 736 bit times. The timeout period can be increased in increments of 64 high speed bit times (133 ns). There are 16 possible values. By default, the adder is 0 thus setting the high speed timeout to its minimum value. Use of this register will allow the high speed timeout to be set to values that are greater the maximum specified in USB 2.0 making the MUSBMHDRC non-compliant.

C_T_HSBT 0x13440348

Bit	3	2	1	0
	C_T_HSBT			
RST	0	0	0	0

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th><th>CPU</th><th>USB</th></tr>
<tr><td>3:0</td><td>C_T_HSBT</td><td>The value added to the minimum High Speed Timeout period (736 bit times) in increments of 64 High Speed bit times. This allows the turn around timeout period to be set to 16 possible values as follows:
<table>
<tr><th>Register Value</th><th>HS Turnaround Timeout (HS Bit times)</th><th>HS Turnaround Timeout (us)</th></tr>
<tr><td>0</td><td>736</td><td>1.534</td></tr>
<tr><td>1</td><td>800</td><td>1.667</td></tr>
<tr><td>2</td><td>864</td><td>1.801</td></tr>
<tr><td>3</td><td>928</td><td>1.934</td></tr>
<tr><td>4</td><td>992</td><td>2.067</td></tr>
<tr><td>5</td><td>1056</td><td>2.201</td></tr>
<tr><td>6</td><td>1120</td><td>2.334</td></tr>
<tr><td>7</td><td>1184</td><td>2.467</td></tr>
<tr><td>8</td><td>1248</td><td>2.601</td></tr>
<tr><td>9</td><td>1312</td><td>2.734</td></tr>
<tr><td>10</td><td>1376</td><td>2.868</td></tr>
<tr><td>11</td><td>1440</td><td>3.001</td></tr>
<tr><td>12</td><td>1504</td><td>3.134</td></tr>
<tr><td>13</td><td>1568</td><td>3.268</td></tr>
<tr><td>14</td><td>1632</td><td>3.401</td></tr>
<tr><td>15</td><td>1696</td><td>3.534</td></tr>
</table></td><td>RW</td><td>R</td></tr>
</table>

6.10 DMA Registers

6.10.1 DMA_INTR

This register provides an interrupt for each DMA channel. This interrupt register is cleared when read. When any bit of this register is set, the output DMA_NINT is asserted low. Events that cause interrupts to be set is described in section 17 (The optional DMA Controller description). Bits in this register will only be set if the DMA Interrupt Enable bit for the corresponding channel is enabled (register DMA_CNTL.D3).

DMA_INTR **0x13440200**

Bit	7	6	5	4	3	2	1	0
	CH8_INT	CH7_INT	CH6_INT	CH5_INT	CH4_INT	CH3_INT	CH2_INT	CH1_INT
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
7	CH8_INT	Channel 8 DMA Interrupt.	RW	Set
6	CH7_INT	Channel 7 DMA Interrupt.	RW	Set
5	CH6_INT	Channel 6 DMA Interrupt.	RW	Set
4	CH5_INT	Channel 5 DMA Interrupt.	RW	Set
3	CH4_INT	Channel 4 DMA Interrupt.	RW	Set
2	CH3_INT	Channel 3 DMA Interrupt.	R	Set
1	CH2_INT	Channel 2 DMA Interrupt.	R	Set
0	CH1_INT	Channel 1 DMA Interrupt.	R	Set

6.10.2 DMA_CNTL

This register is only available if the MUSBMHDRC is configured to use at least one internal DMA channel. This register provides the DMA transfer control for each channel. The enabling, transfer direction, transfer mode, the DMA burst modes are all controlled by this register.

DMA_CNTL **0x13440204+(n-1)*0x10**

Bit	10	9	8	7	6	5	4	3	2	1	0
	DMA_BRSTM		DMA_ERR	DMAEP				DMAIE	DMAMODE	DMA_DIR	DMA_EN
RST	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
10:9	DMA_BRSTM	Burst Mode. 00 : (Burst Mode 0 : Bursts of unspecified length) 01: (Burst Mode 1 : INCR4 or unspecified length) 10: (Burst Mode 2 : INCR8, INCR4 or unspecified length) 11: (Burst Mode 3 : INCR16, INCR8, INCR4 or unspecified length)	RW	R
8	DMA_ERR	Bus Error Bit. Indicates that a bus error has been observed on the input. AHB_HRESPM[1:0]. This bit is cleared by software.	RW	RW
7:4	DMAEP	The endpoint number this channel is assigned to.	RW	R
3	DMAIE	DMA Interrupt Enable.	RW	R
2	DMAMODE	This bit selects the DMA Transfer Mode. 0: DMA Mode0 Transfer 1: DMA Mode1 Transfer	RW	R
1	DMA_DIR	This bit selects the DMA Transfer Direction. 0: DMA Write (RX Endpoint) 1: DMa Read (TX Endpoint)	RW	R
0	DMA_ENAB	This bit enables the DMA transfer and will cause the transfer to begin.	RW	R

6.10.3 DMA_ADDR

This register identifies the current memory address of the corresponding DMA channel. The Initial memory address written to this register must have a value such that its modulo 4 value is equal to 0. That is, DMA_ADDR[1:0] must be equal to 2'b00. The lower two bits of this register are read only and cannot be set by software. As the DMA transfer progresses, the memory address will increment as bytes are transferred.

DMA_ADDR **0x13440208+(n-1)*0x10**

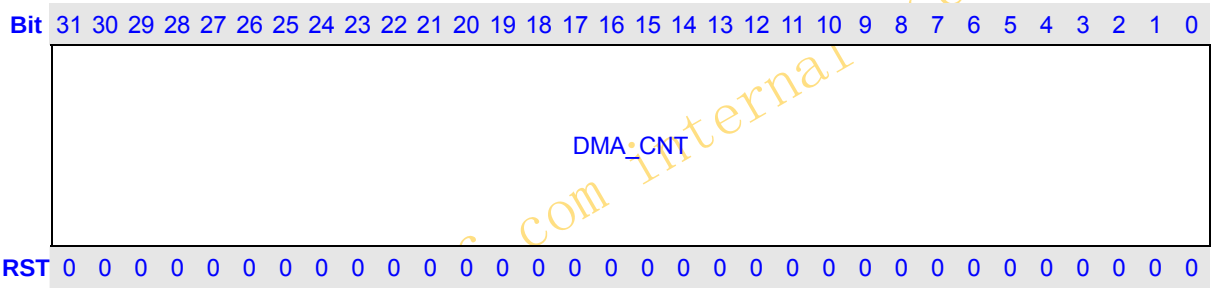
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DMA_ADDR																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	CPU	USB
31:0	DMA_ADDR	The DMA memory address. **Note that the initial memory address written to this register must have a value such that it's modulo 4 value is equal to 0. That is, DMA_ADDR[1:0] must be equal to 2'b00. The lower two bits of this register are read only and cannot be set by software.**	RW	RW

6.10.4 DMA_COUNT

This register identifies the current DMA count of the transfer. Software will set the initial count of the transfer which identifies the entire transfer length. As the count progresses this count is decremented as bytes are transferred.

DMA_CNT **0x1344020c+(n-1)*0x10**

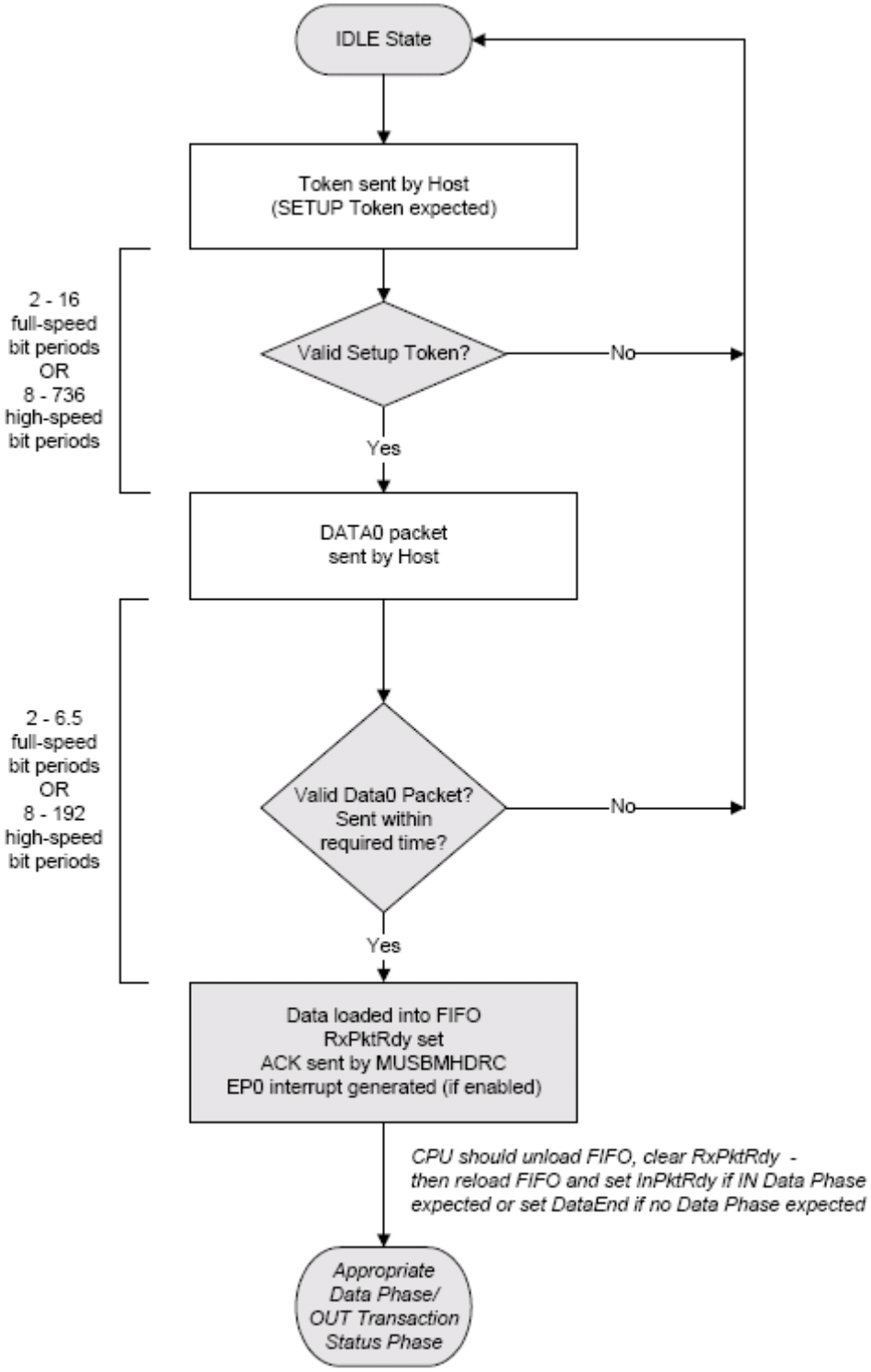


31:0	DMA_CNT	The DMA memory address for the corresponding DMA channel. **NOTE:** If DMA is enabled with a count of 0, the bus will not be requested and a DMA interrupt will be generated.	RW	RW

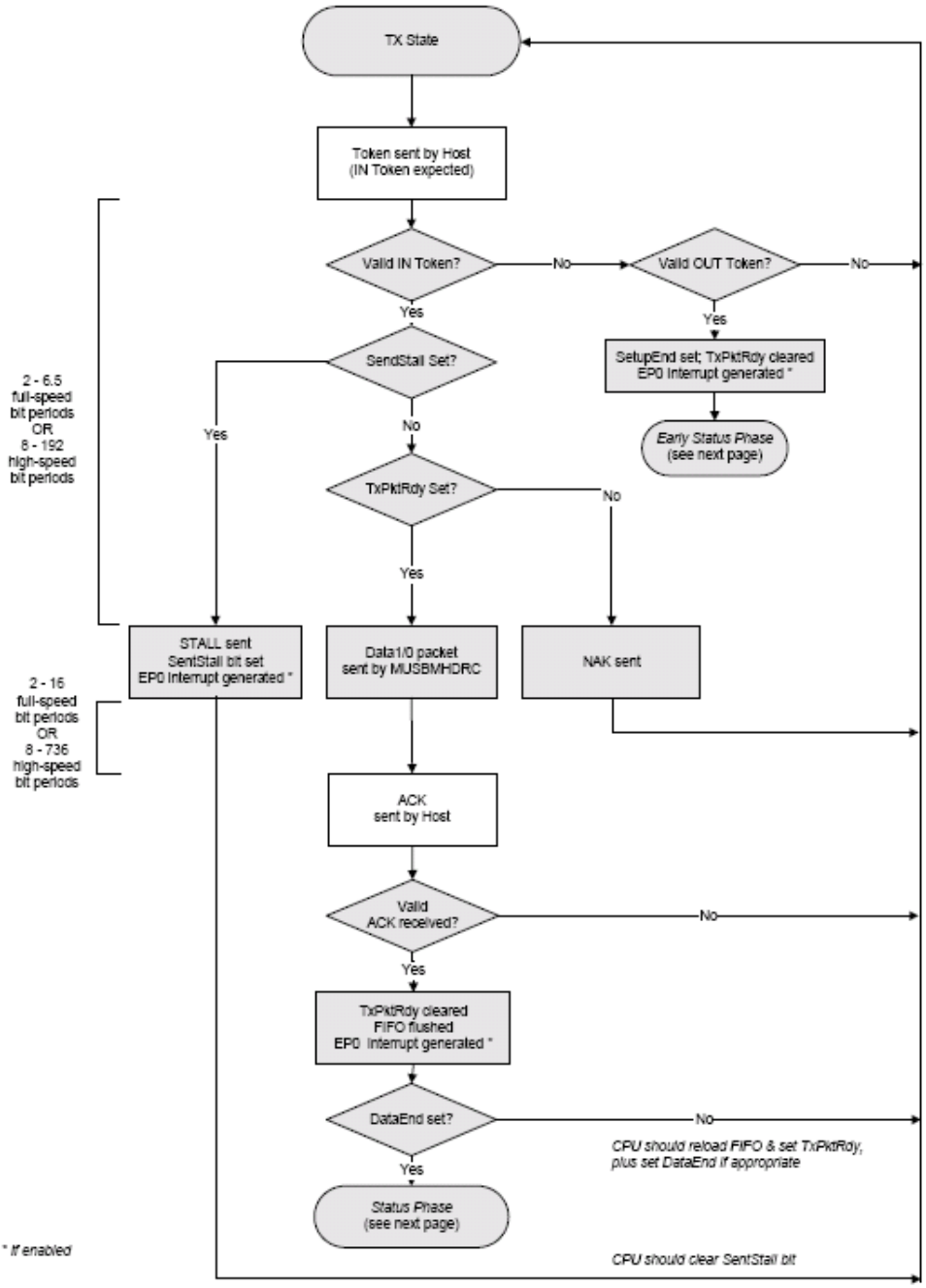
6.11 Transaction flows as a peripheral

6.11.1 Control transactions

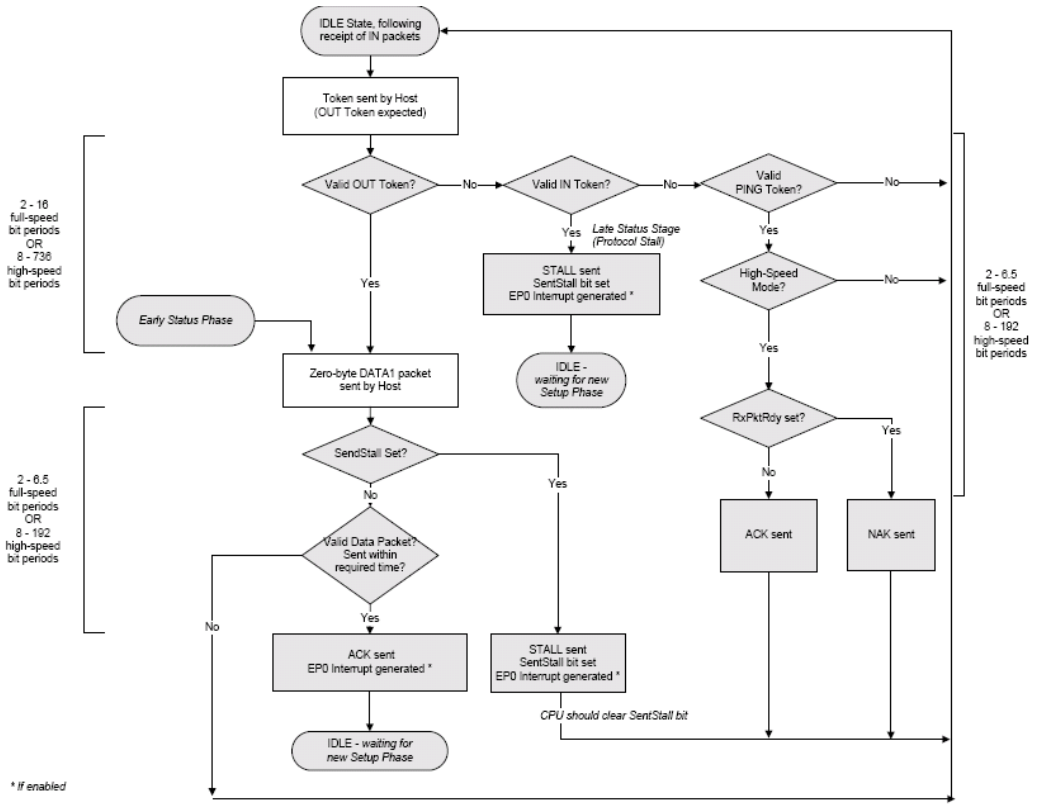
6.11.1.1 Setup phase



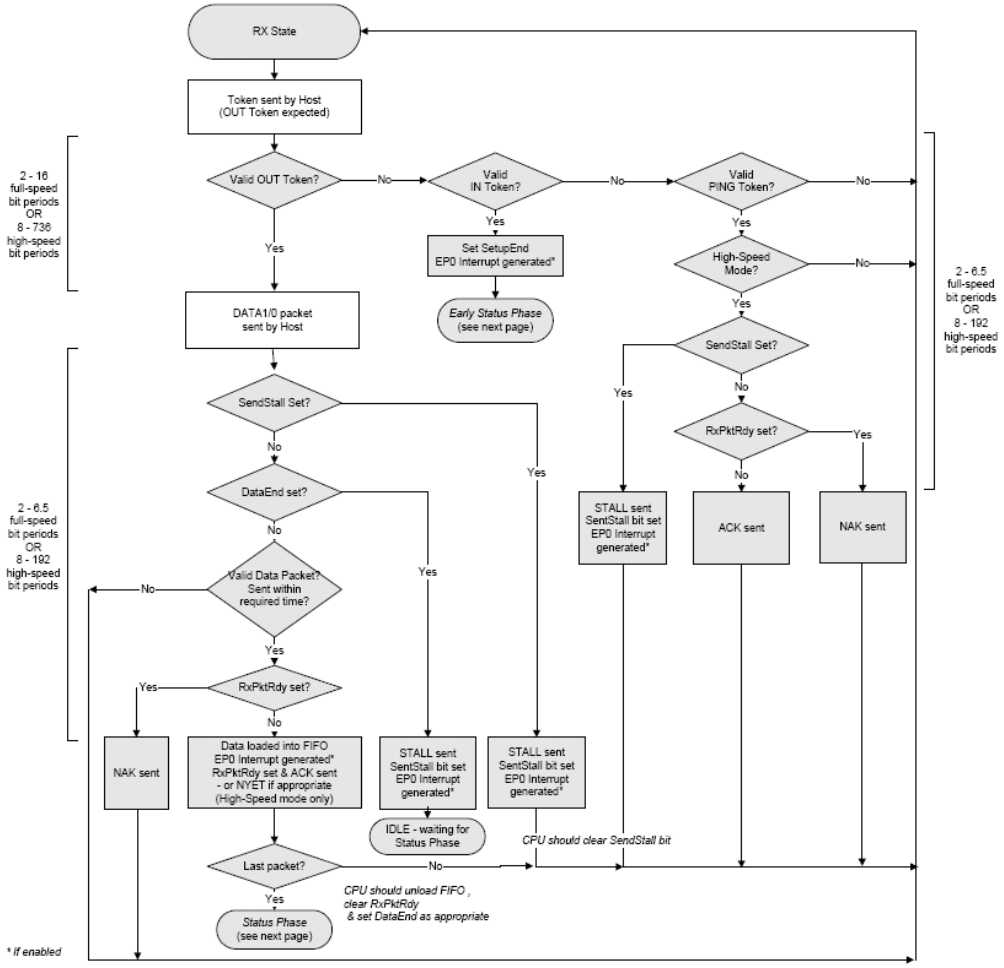
6.11.1.2 In data phase



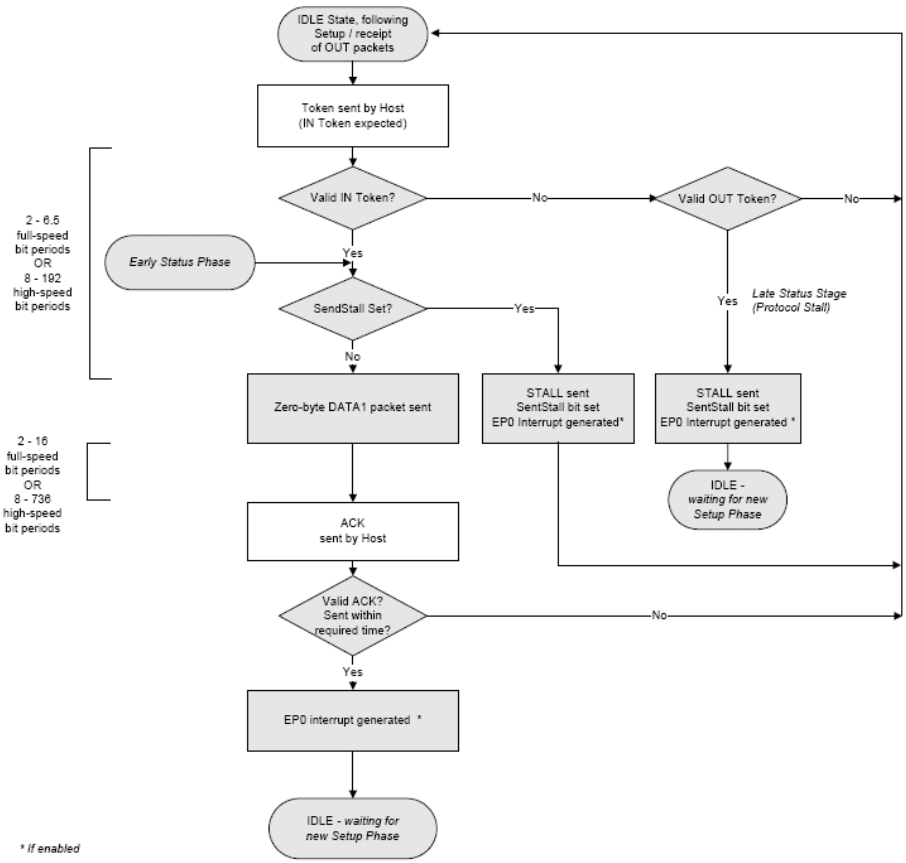
6.11.1.3 Following the status phase



6.11.1.4 Out data phase

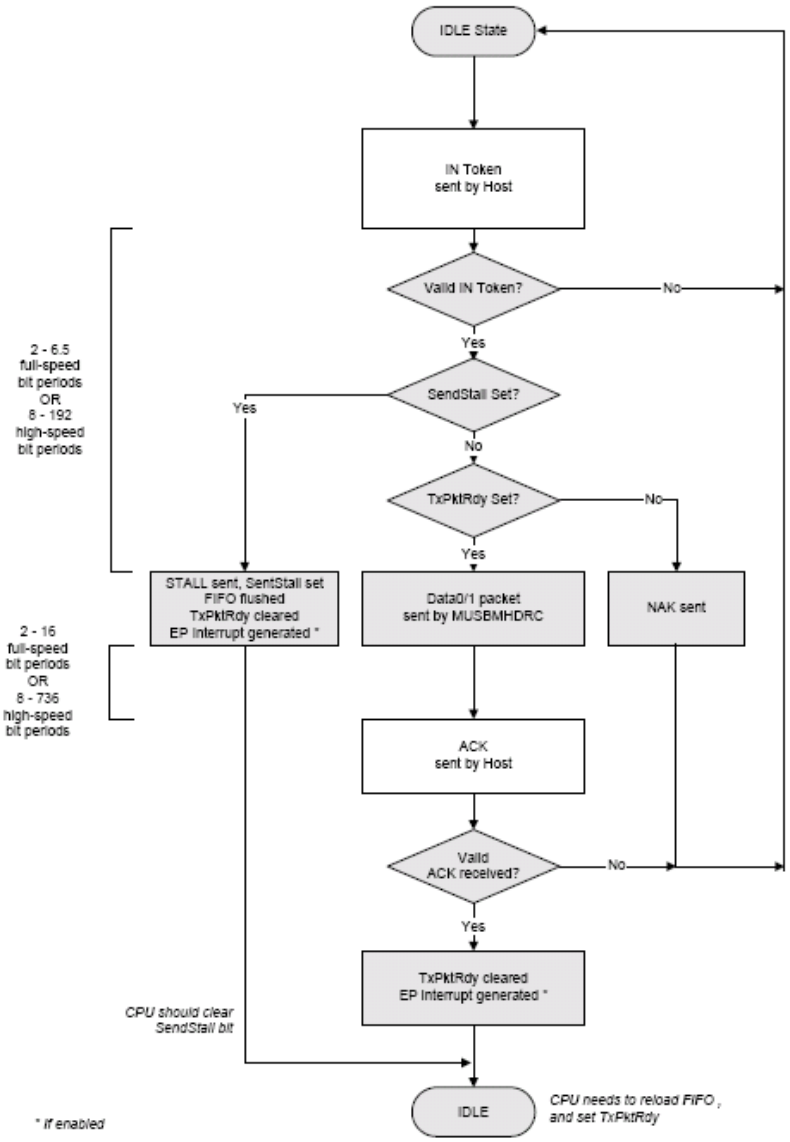


6.11.1.5 Following the status phase

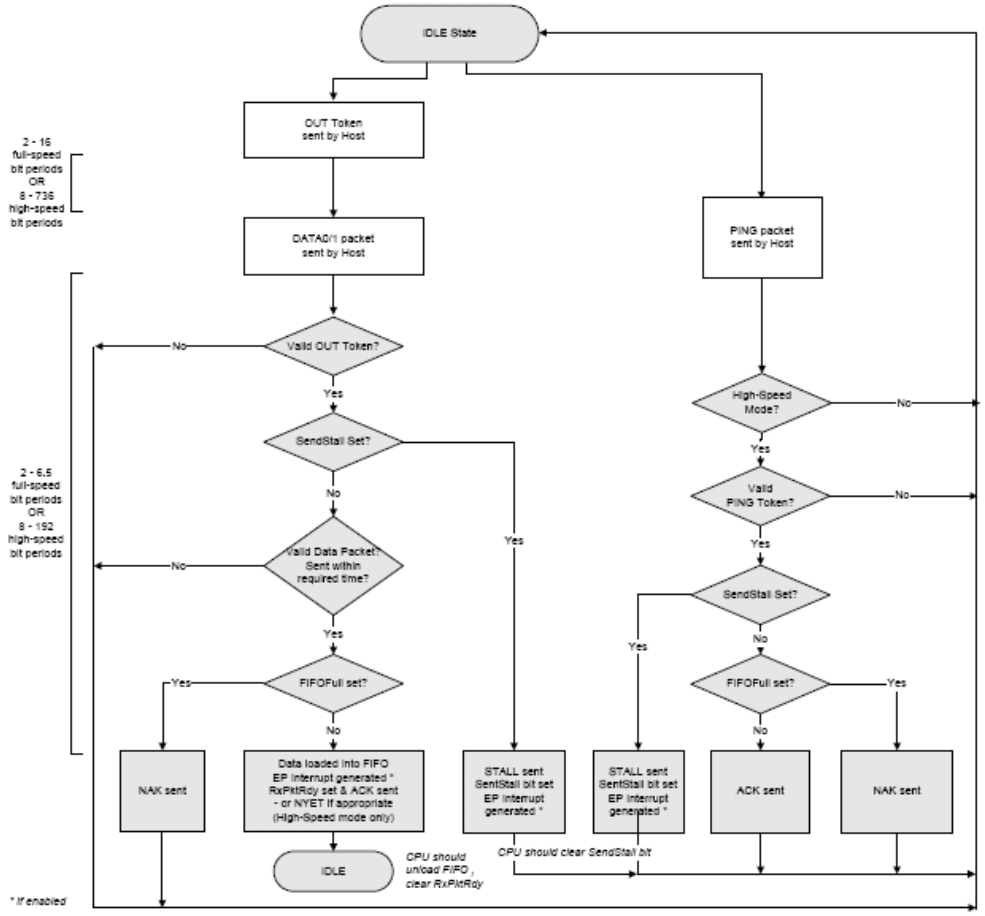


6.11.2 Bulk/Low-bandwidth interrupt transactions

6.11.2.1 In transaction

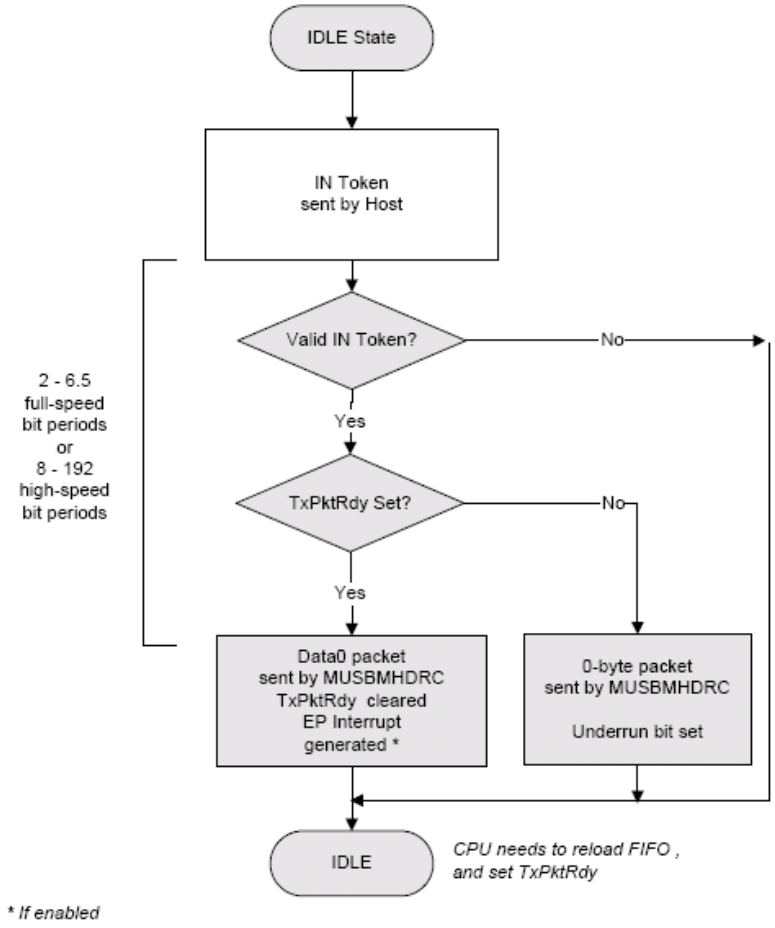


6.11.2.2 Out transaction

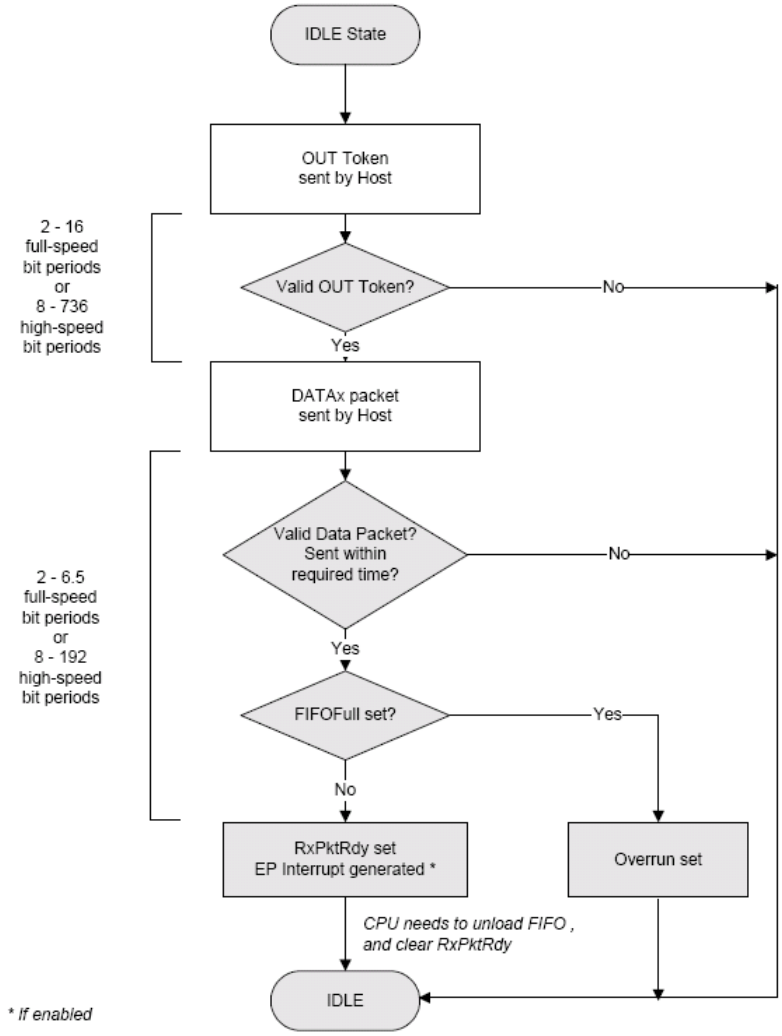


6.11.3 Full-speed/Low-bandwidth isochronous transactions

6.11.3.1 In transaction

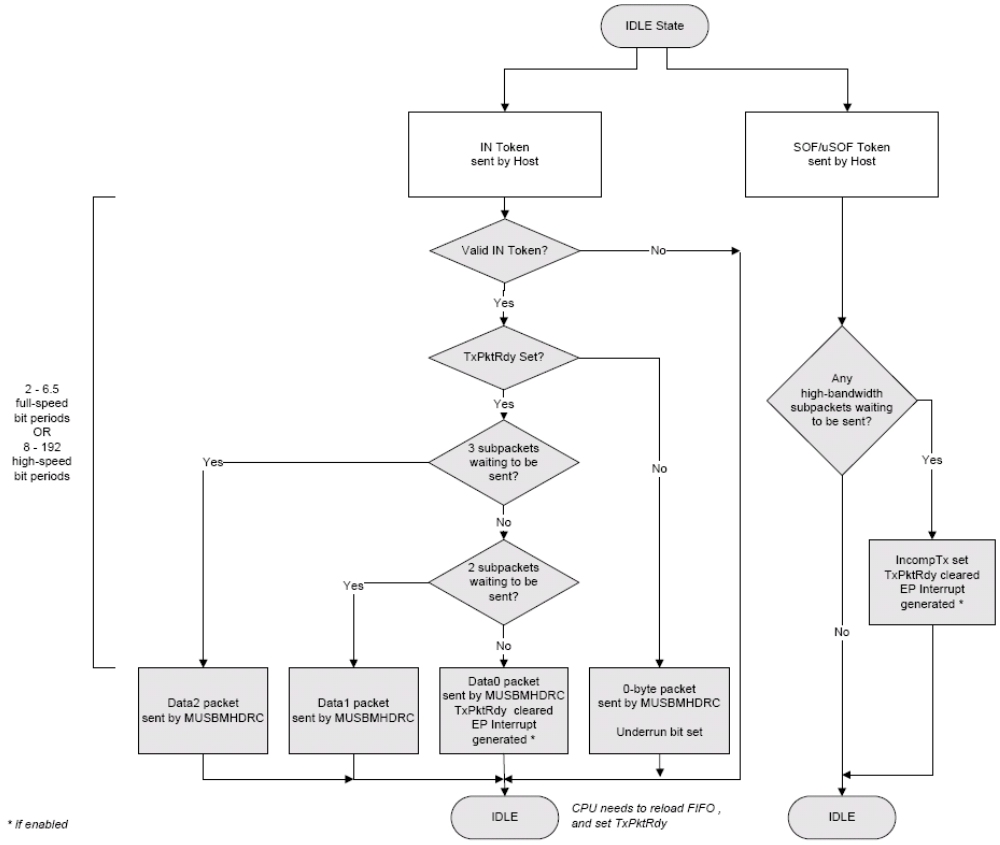


6.11.3.2 Out transaction

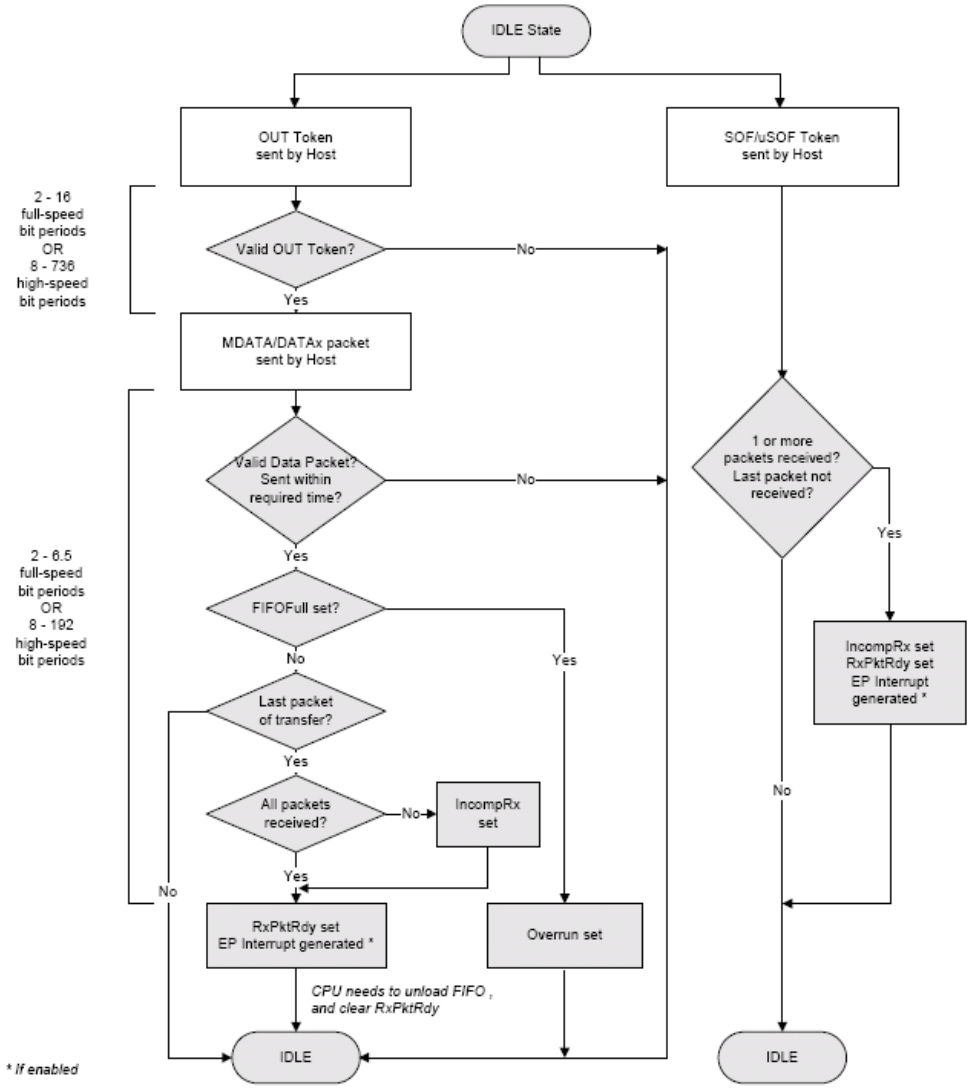


6.11.4 High-bandwidth transactions (Isochronous and interrupt)

6.11.4.1 In transaction



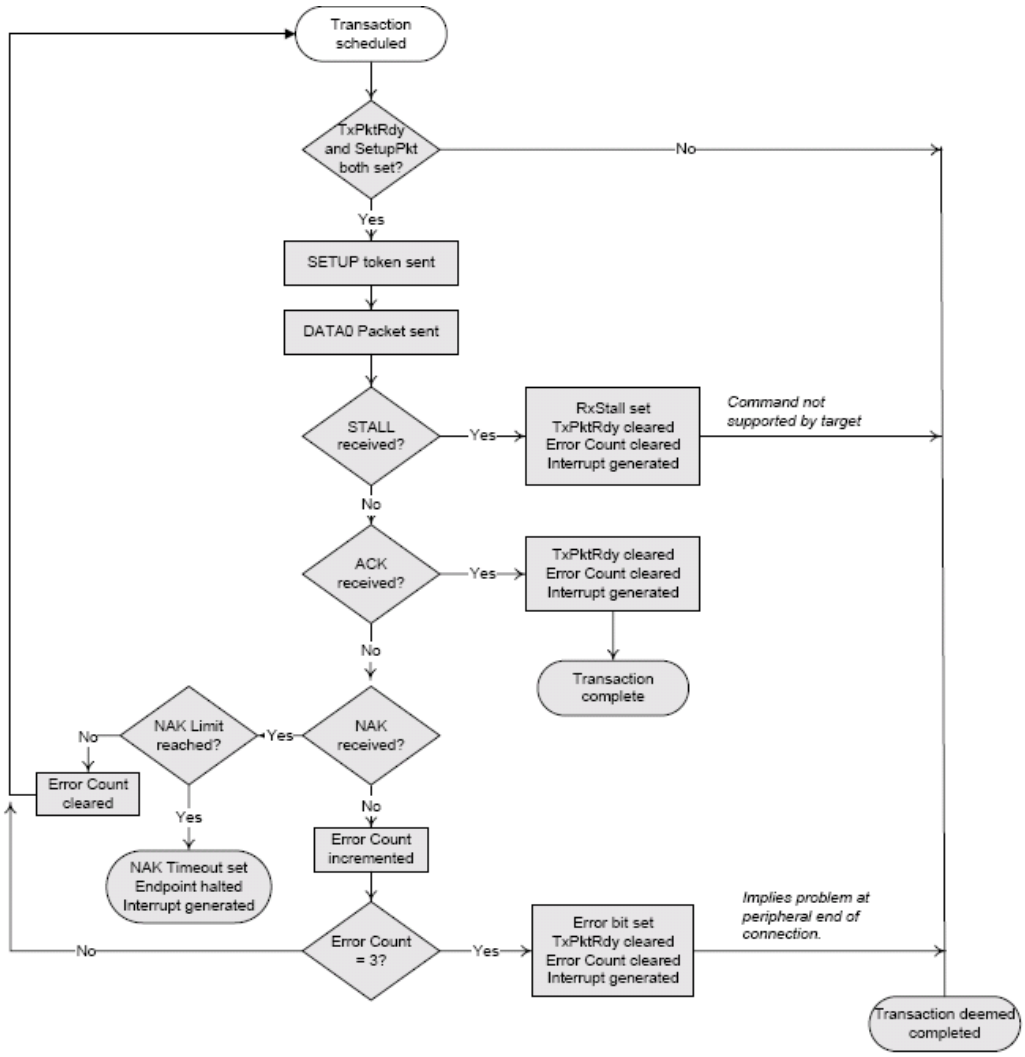
6.11.4.2 Out transaction



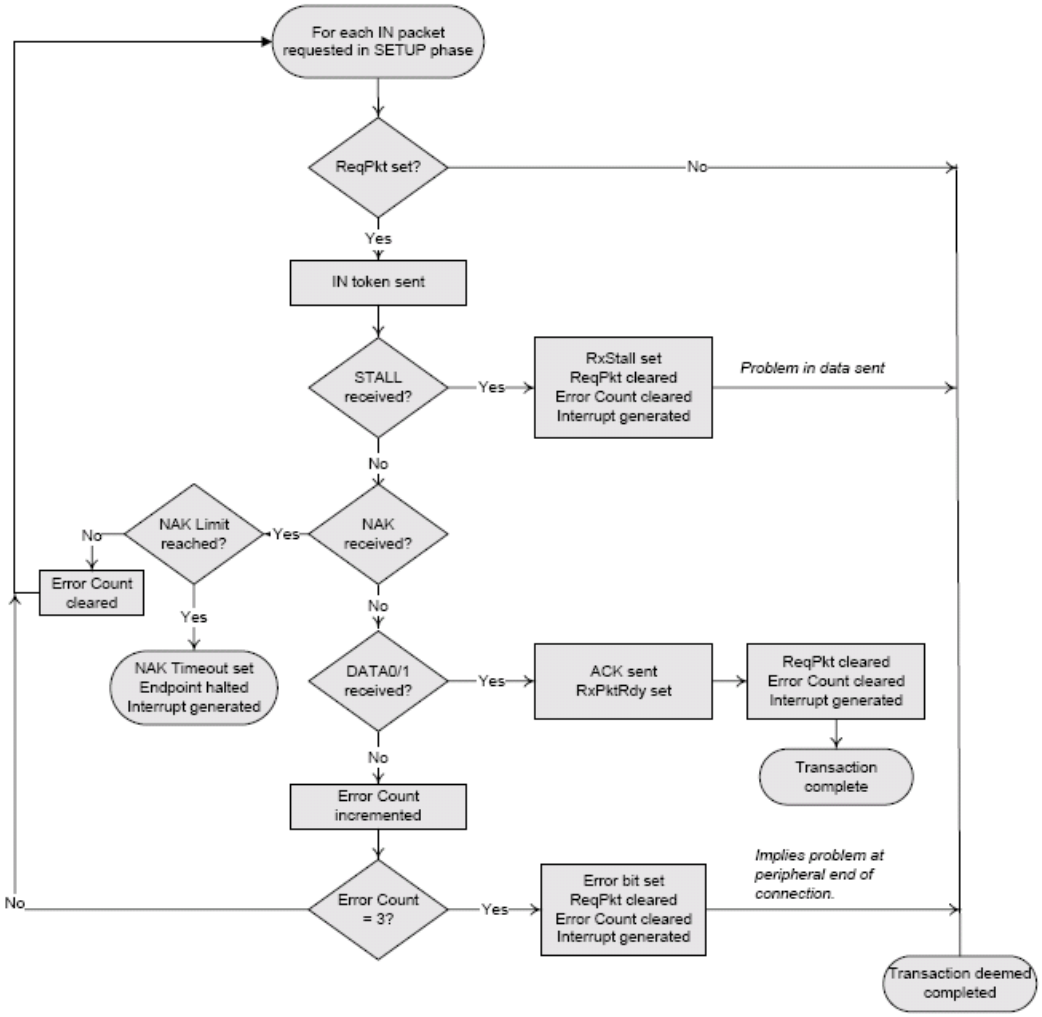
6.12 Transaction flows as a host

6.12.1 Control transactions

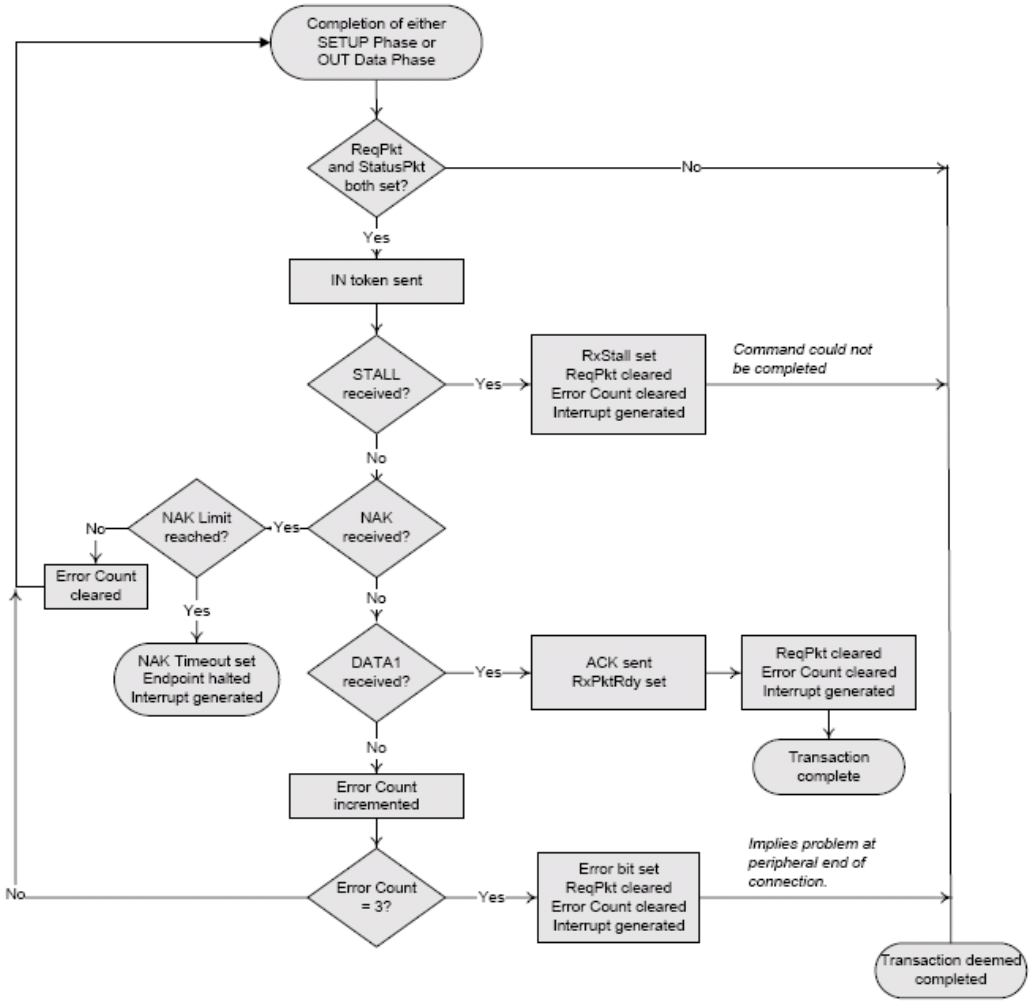
6.12.1.1 Setup phase



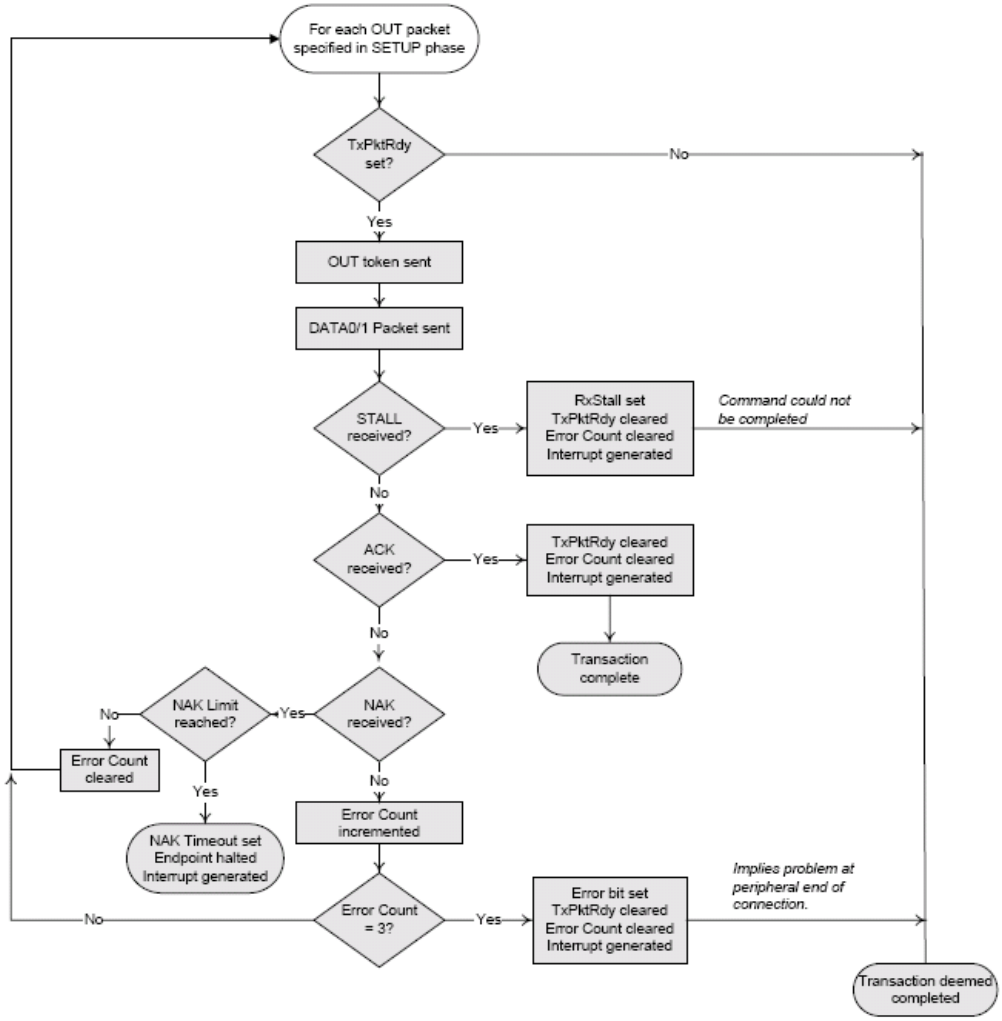
6.12.1.2 In data phase



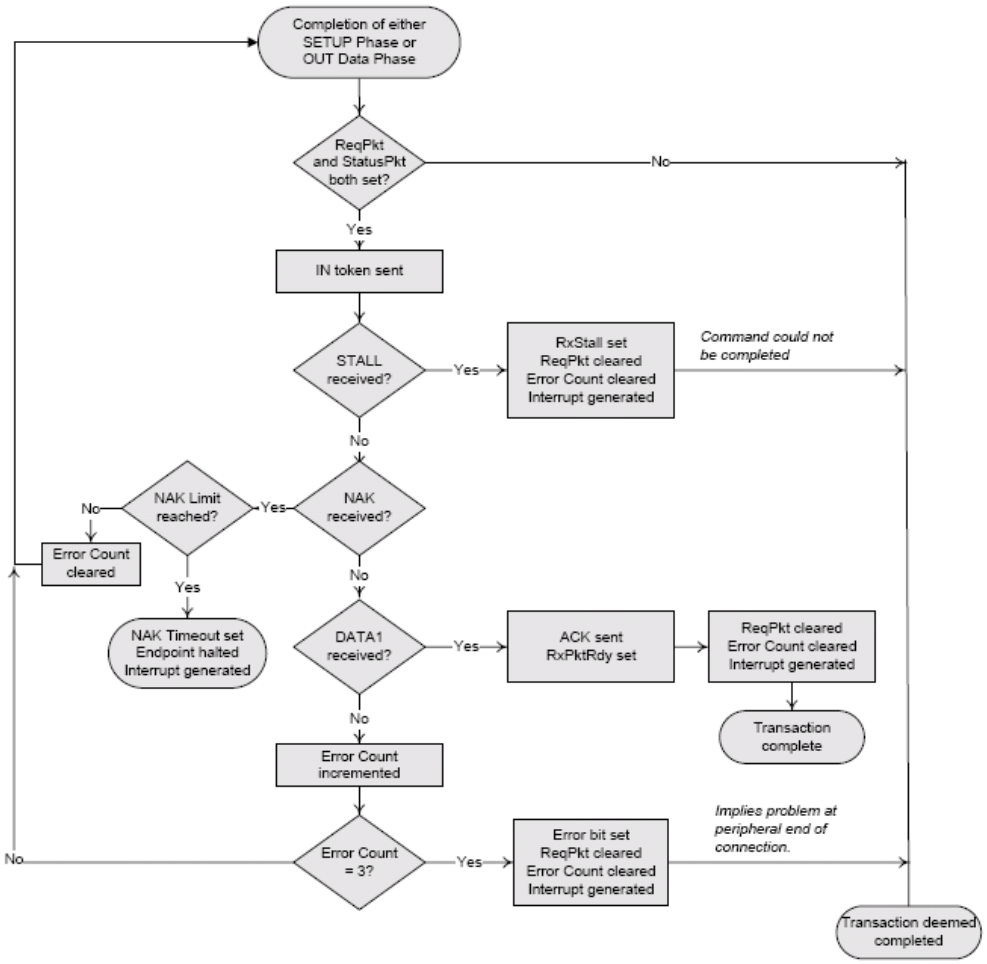
6.12.1.3 Following the status phase



6.12.1.4 Out data phase

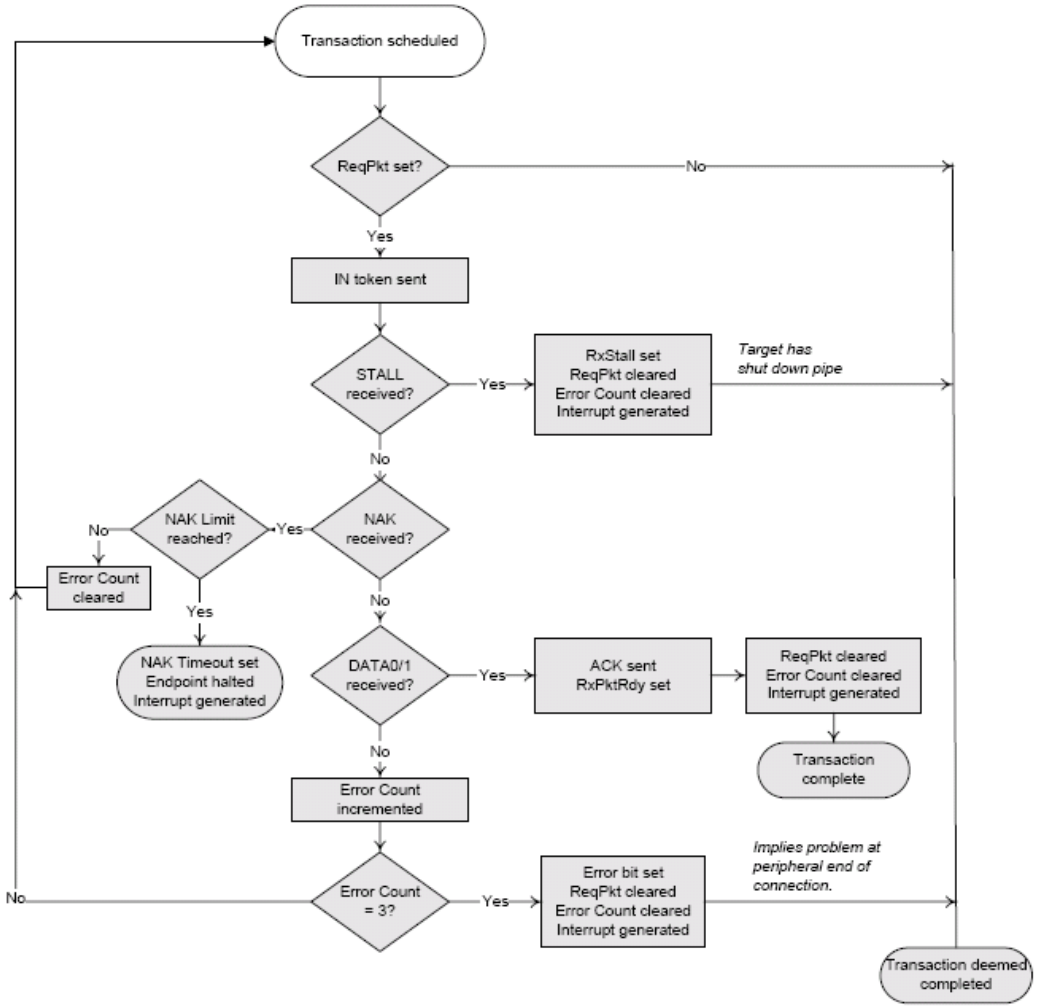


6.12.1.5 Following the status phase

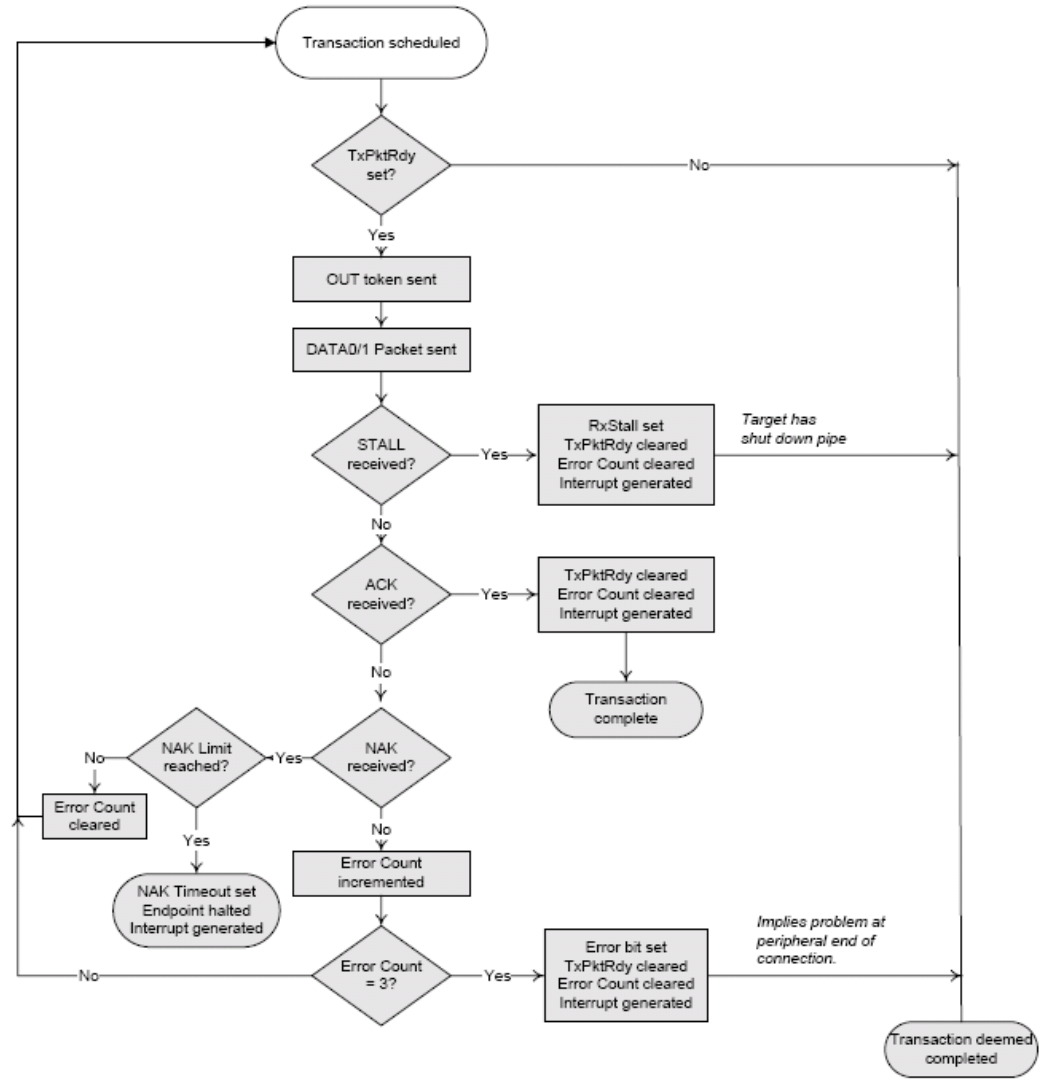


6.12.2 Bulk/Low-bandwidth interrupt transactions

6.12.2.1 In transaction

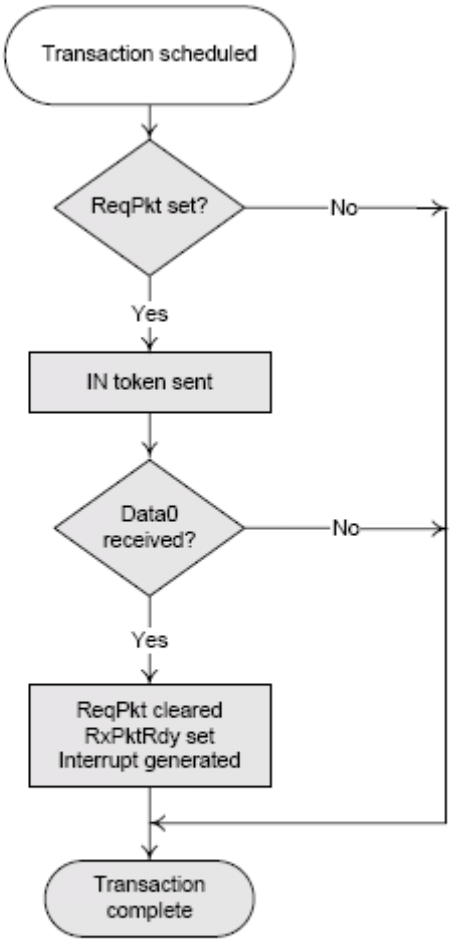


6.12.2.2 Out transaction

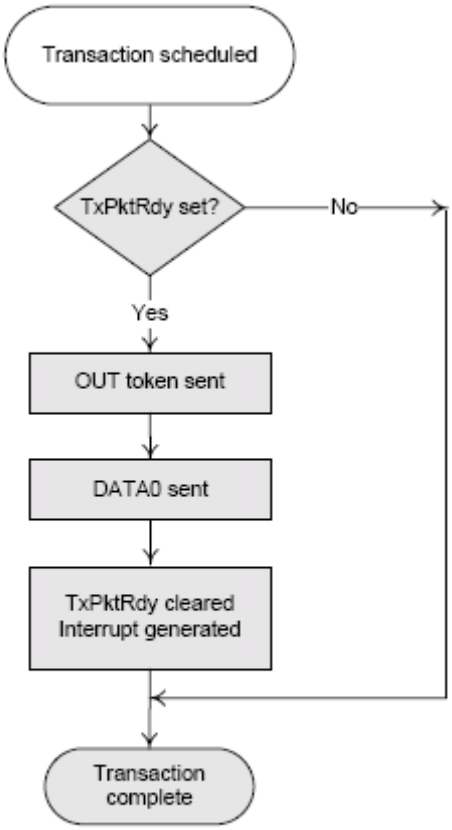


6.12.3 Full-speed/Low-bandwidth isochronous transactions

6.12.3.1 In transaction

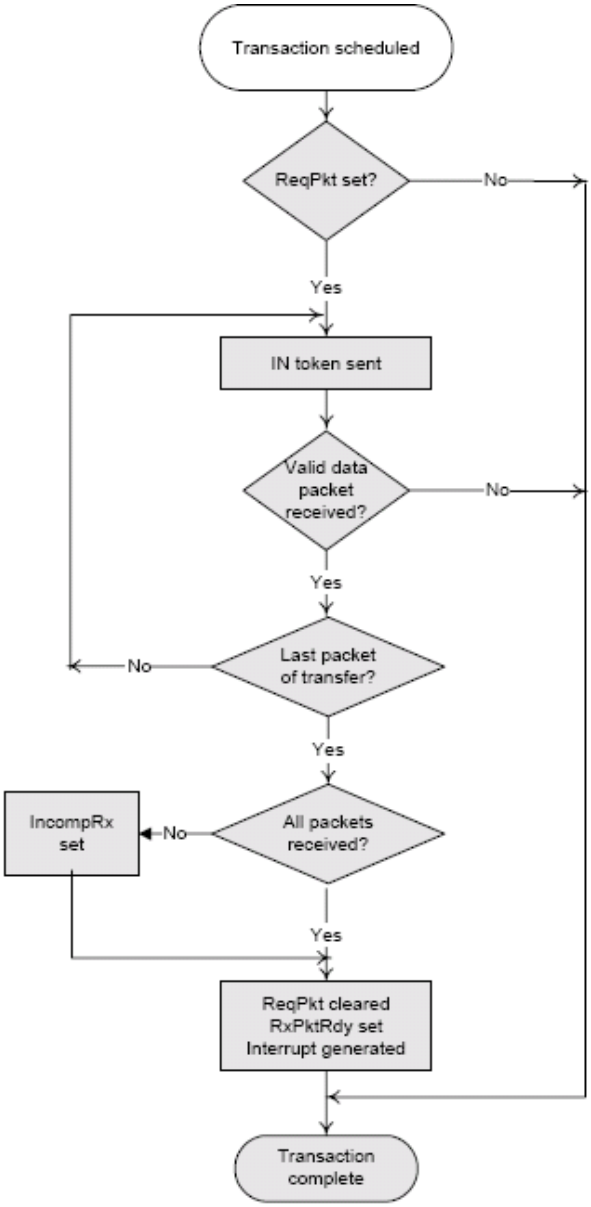


6.12.3.2 Out transaction

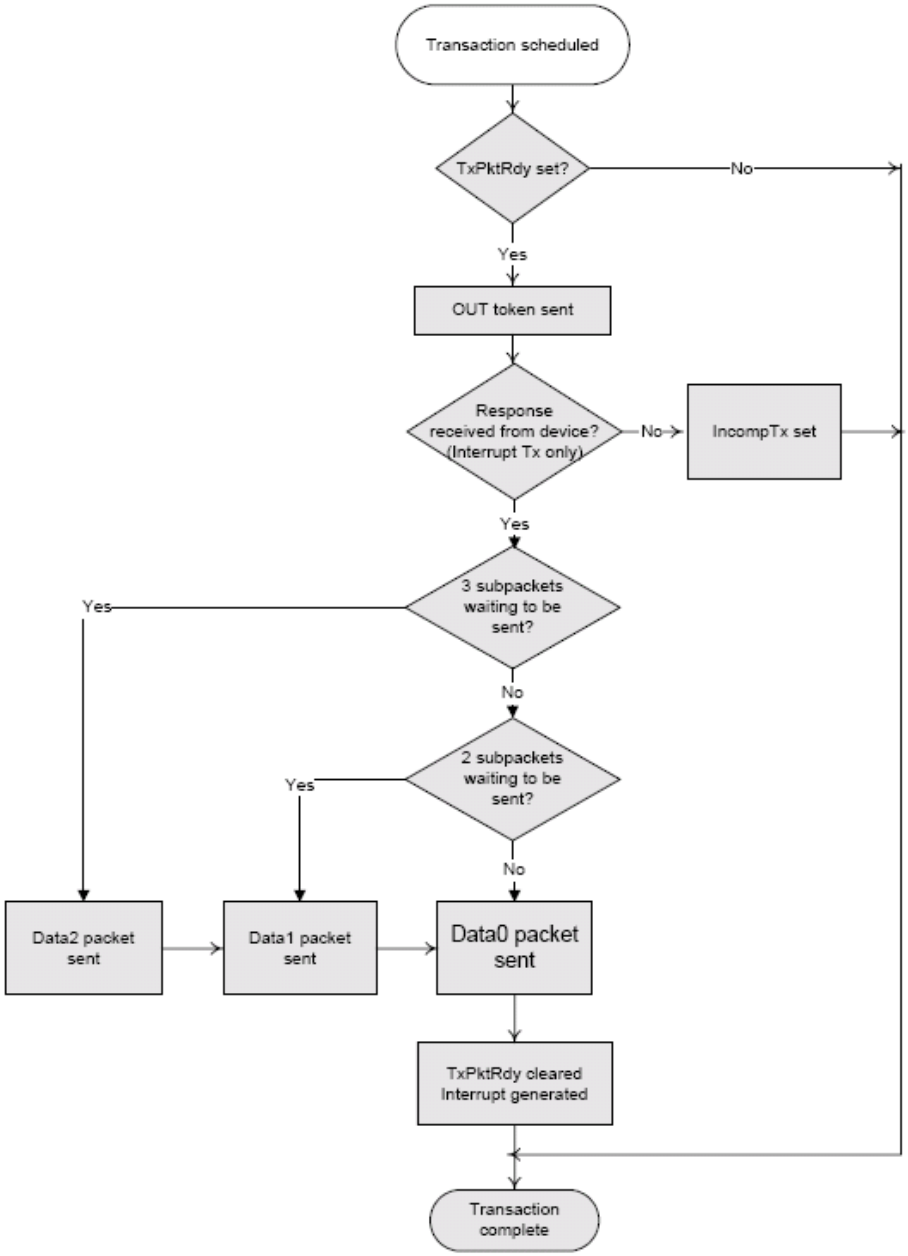


6.12.4 High-bandwidth transactions (isochronous and interrupt)

6.12.4.1 In transaction

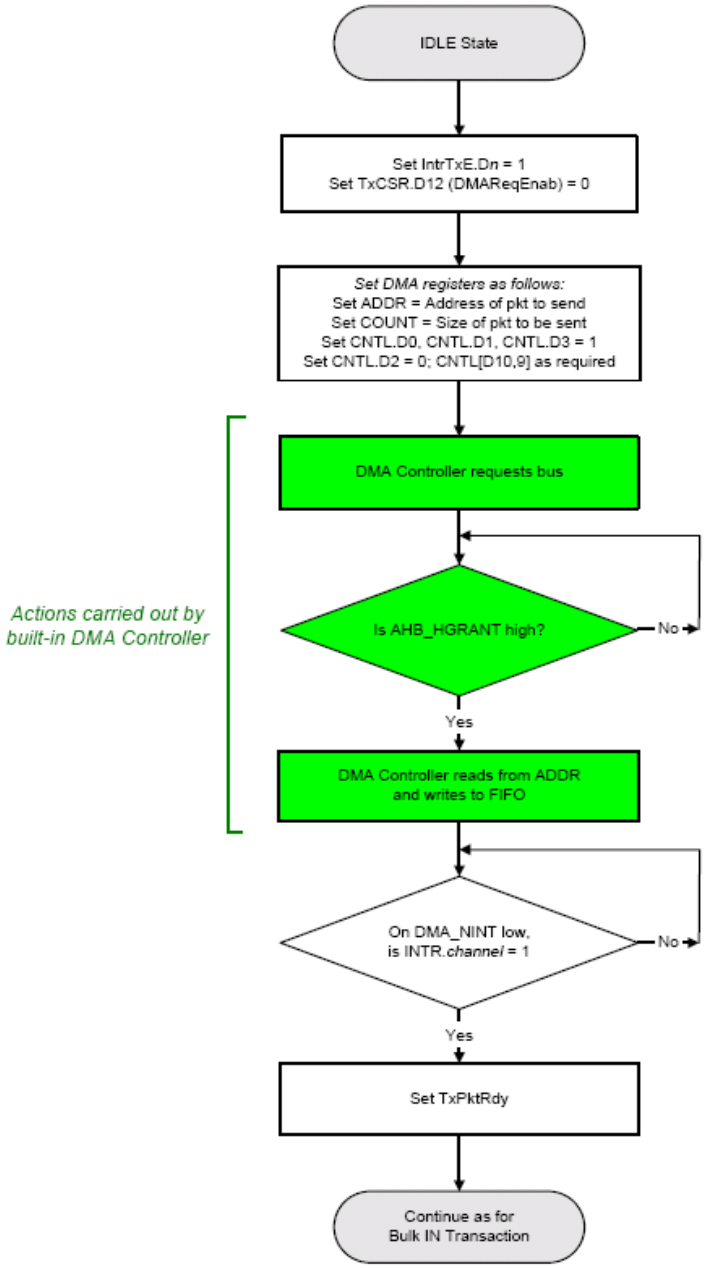


6.12.4.2 Out transaction

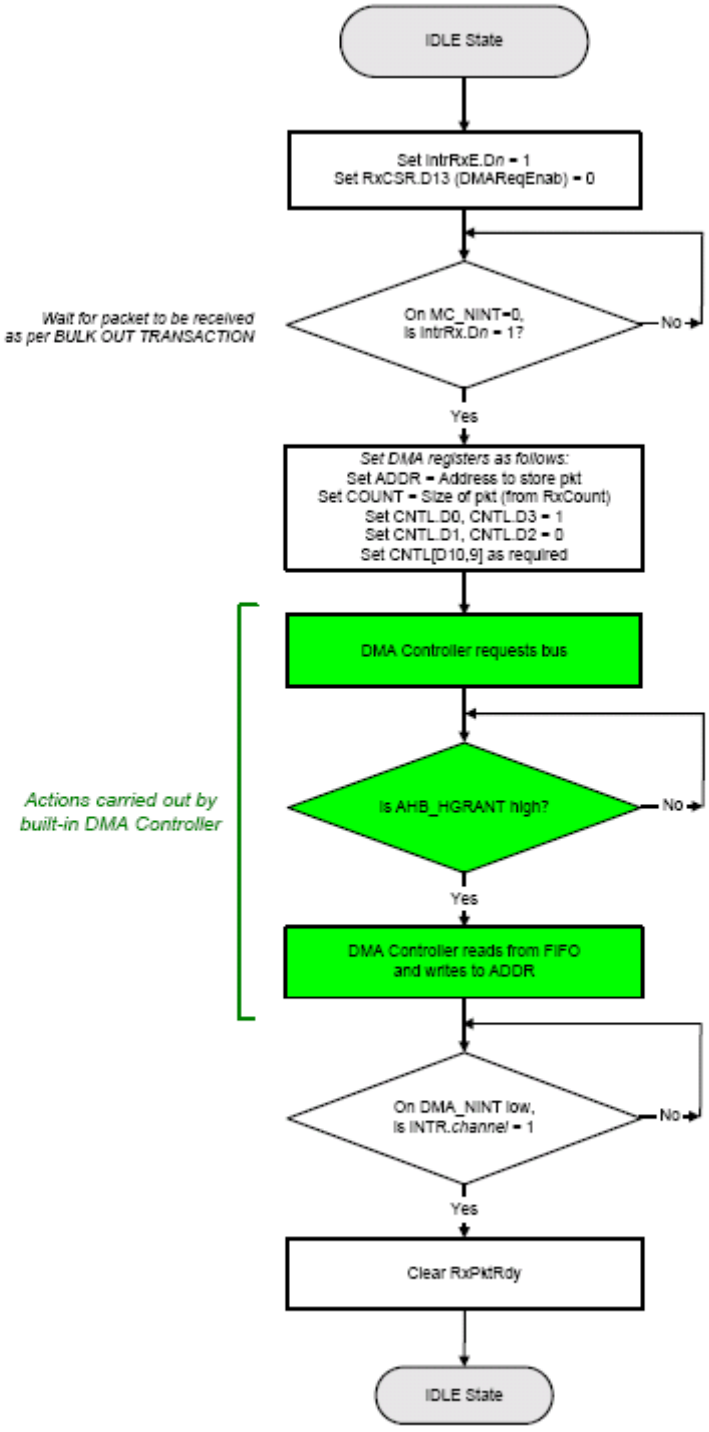


6.13 DMA operations

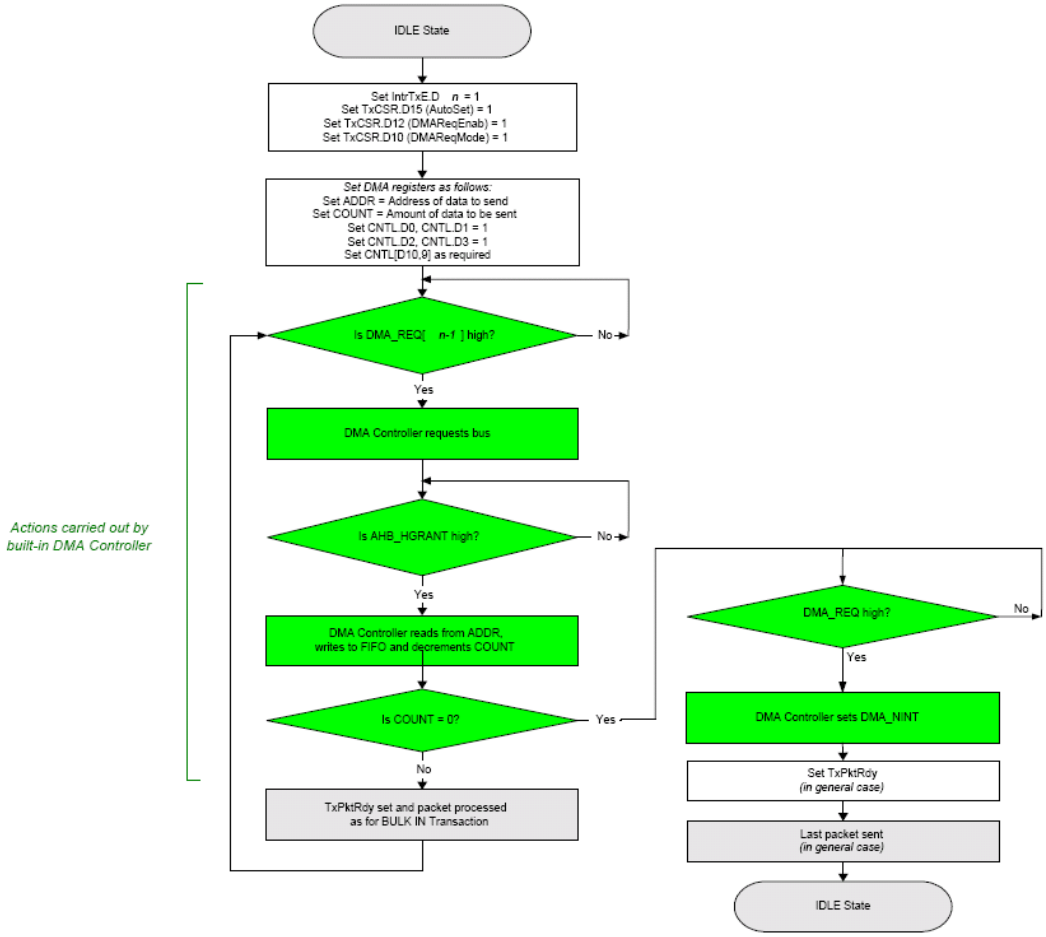
6.13.1 Single packet tx



6.13.2 Single packet rx

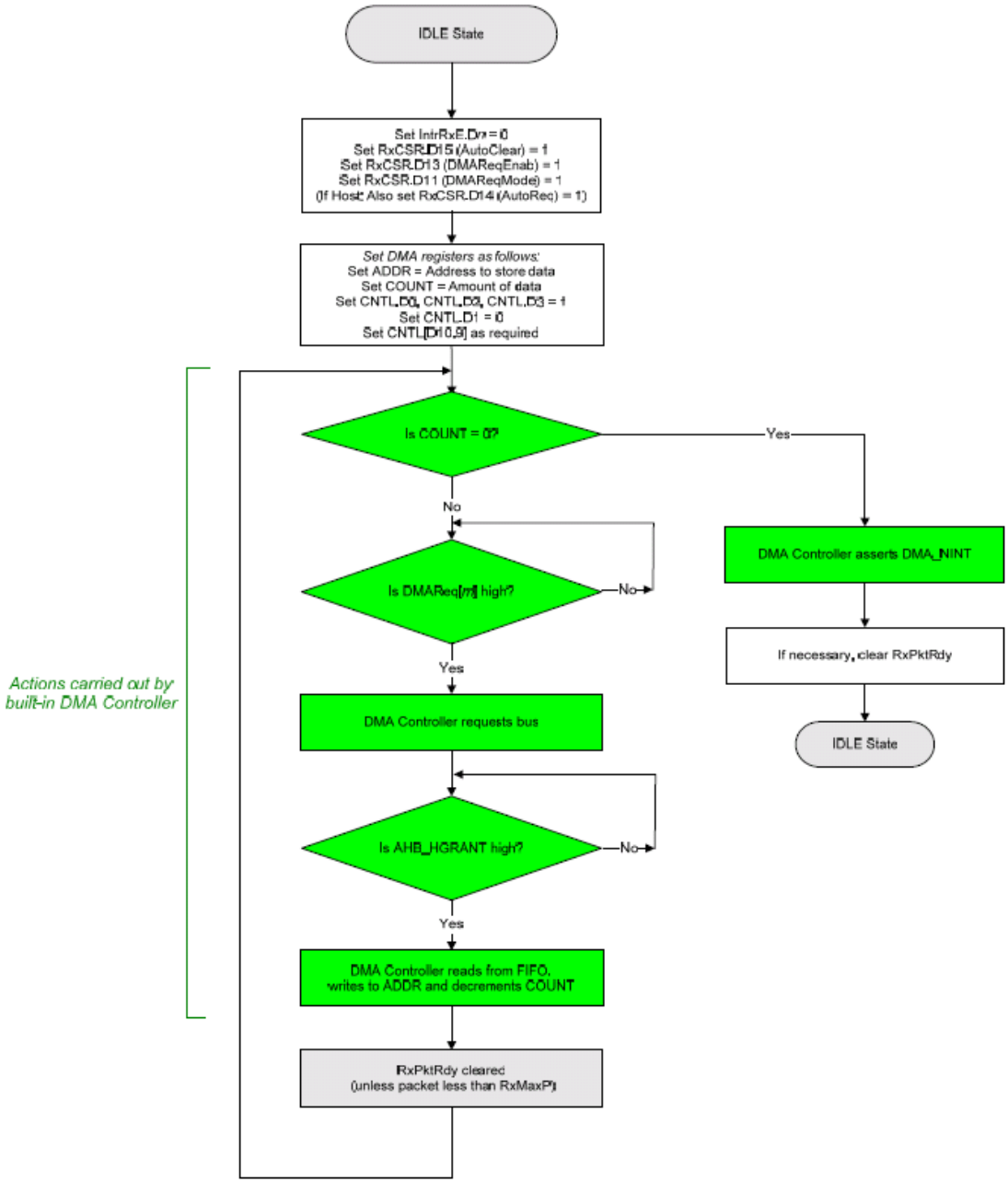


6.13.3 Multiple packet tx

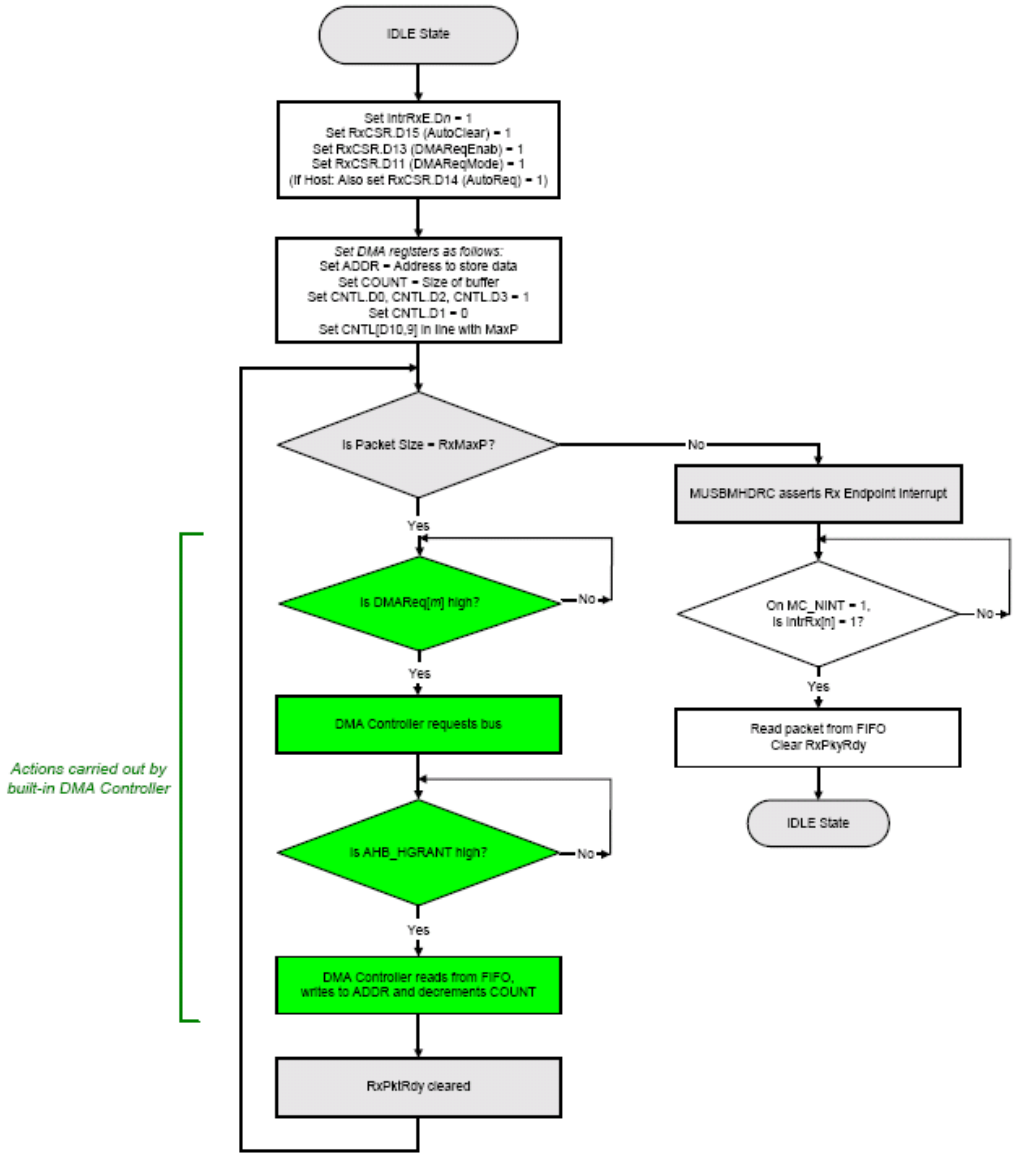


6.13.4 Multiple packet rx

If Size of Data Block Known:



If Size of Data Block Not Known:



7 MMC/SD CE-ATA Controller

7.1 Overview

The MultiMediaCard (MMC) is a universal low cost data storage and communication media that is designed to cover a wide area of applications such as electronic toys, organizers, PDAs, smart phones, and so on.

The Secure Digital (SD) card is an evolution of MMC, It is specifically designed to meet the security, capacity, performance, and environmental requirements inherent in newly emerging audio and video consumer electronic devices. The physical form factor, pin assignment, and data transfer protocol are forward compatible with the MultiMediaCard with some additions. An SD card can be categorized as SD memory or SD I/O card, commonly known as SDIO. A memory card invokes a copyright protection mechanism that complies with the security of the SDMI standard and is faster and capable of higher memory capacity. The SDIO card provides high-speed data I/O with low-power consumption for mobile electronic devices.

For CE-ATA detail protocol , please referred to WWW.CE-ATA.ORG.

Features of the MSC Controller include the following:

- Fully compatible with the *MMC System Specification version 4.2*
- Fully compatible with the *SD Memory Card Specification 2.0* and *SD I/O Specification 1.0* with 1 command channel and 4 data channels
- Consumer Electronics Advanced Transport Architecture (CE-ATA – version 1.1)
- 20-80 Mbps maximum data rate
- Support MMC data width 1bit ,4bit and 8bit
- Built-in programmable frequency divider for MMC/SD bus
- Maskable hardware interrupt for SDIO interrupt, internal status and FIFO status
- 32-entry x 32-bit built-in data FIFO
- Multi-SD function support including multiple I/O and combined I/O and memory
- IRQ supported enable card to interrupt MMC/SD controller
- Single or multi block access to the card including erase operation
- Stream access to the MMC card
- Supports SDIO read wait, interrupt detection during 1-bit or 4-bit access
- Supports CE-ATA digital protocol commands
- Support Command Completion Signal and interrupt to CPU
- Command Completion Signal disable feature
- The maximum block length is 4096bytes

7.2 Block Diagram

MSC Controller Block Diagram

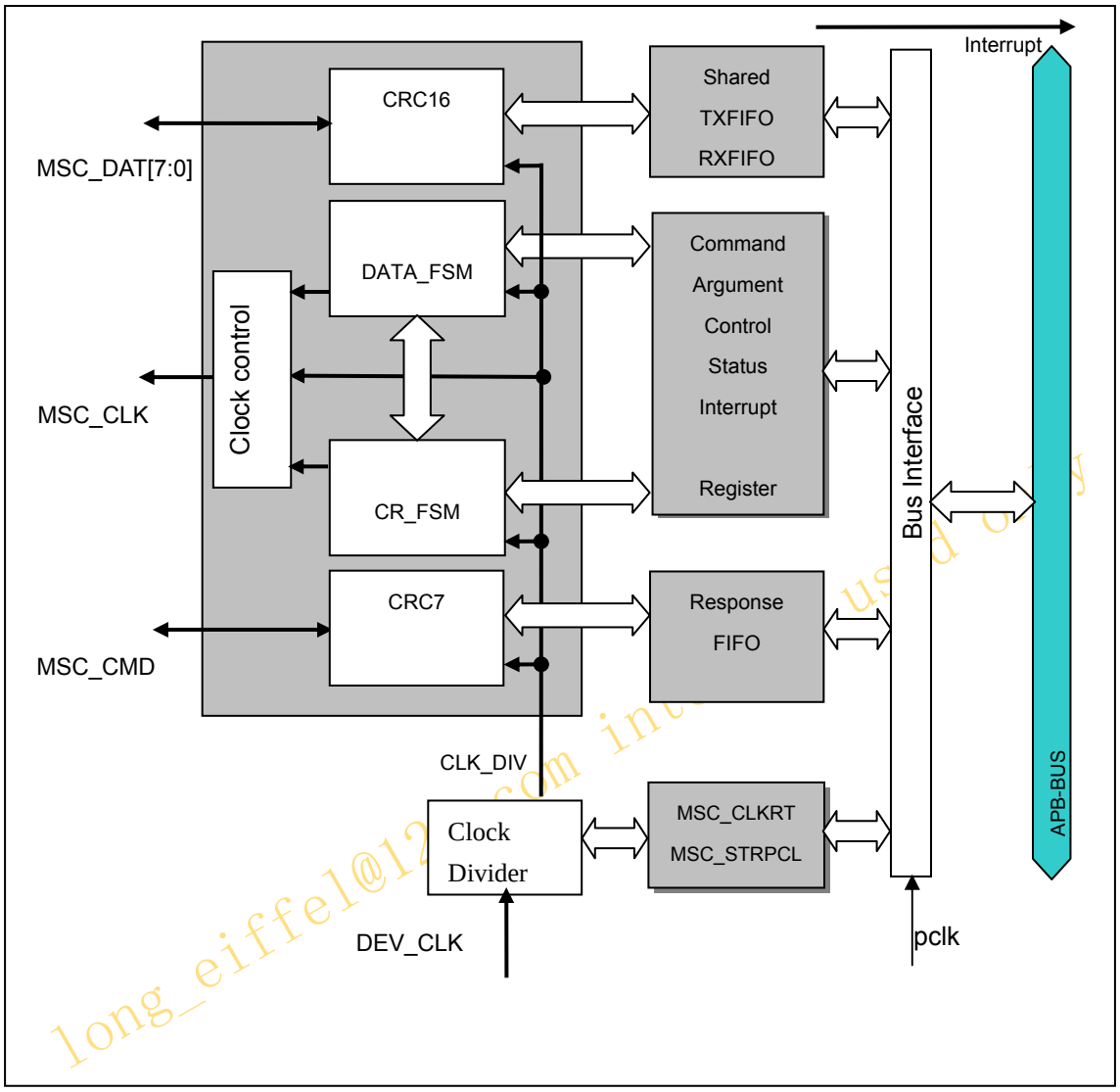


Figure 7-1 MMC/SD CE-ATA Controller Block Diagram

7.3 MMC/SD Controller Signal I/O Description

MSC and the card communication over the CMD and DATA line is base on command and data bit streams which are initiated by a start bit and terminated by a stop bit.

Command: a command is a token, which starts an operation. A command is sent from MSC either to a single card (addressed command) or to all connected cards (broadcast command). A command is transferred serially on the CMD line. Each command token is preceded by a start bit (‘0’) and succeeded by an end bit (‘1’). The total length is 48 bits and protected by CRC bits.

Table 7-1 Command Token Format

Bit position	**47**	**46**	**[45 : 40]**	**[39 : 8]**	**[7 : 1]**	**0**
Width (bits)	1	1	6	32	7	1
Value	0	1	X	X	x	1
Description	Start bit	Transmission bit	Command index	argument	CRC7	End bit

Response: a response is a token which is sent from an addressed card, or (synchronously) from all connected cards, to MSC as an answer to a previously received command. A response is transferred serially on the CMD line. Response tokens have varies coding schemes depending on their content.

Data: data can be transferred from the card to MSC or vice versa. Data is transferred via the data line. Data transfers to/from the SD Memory Card are done in blocks. Data blocks always succeeded by CRC bits. Single and multiple block operations are defined. Note that the Multiple Block operation mode is better for faster write operation. A multiple block transmission is terminated when a stop command follows on the CMD line. Data transfer can be configured by the MSC to use single or multiple data lines.

Table 7-2 MMC/SD Data Token Format

Description	**Start bit**	**Data**	**CRC16**	**End bit**
Stream Data	0	X	no CRC	1
Block Data	0	X	X	1

7.4 Register Description

The MMC-SD-CE_ATA controller is controlled by a set of registers that the application configures before every operation. The Table 7-3 lists all the MSC registers.

MSC0 Base Address：0x10021000
MSC1 Base Address：0x10022000
MSC2 Base Address：0x10023000

Table 7-3 MMC/SD Controller Registers Description

Name	RW	Reset Value	Address	Access Size
MSC_CTRL	W	0x0000	Base + 0x000	16
MSC_STAT	R	0x00000040	Base + 0x004	32
MSC_CLKRT	RW	0x0000	Base + 0x008	16
MSC_CMDAT	RW	0x00000000	Base + 0x00C	32
MSC_RESTO	RW	0x40	Base + 0x010	16
MSC_RDTO	RW	0xFFFF	Base + 0x014	32
MSC_BLKLEN	RW	0x0000	Base + 0x018	16
MSC_NOB	RW	0x0000	Base + 0x01C	16
MSC_SNOB	R	0x????	Base + 0x020	16
MSC_IMASK	RW	0x00FF	Base + 0x024	32
MSC_IREG	RW	0x0000	Base + 0x028	16
MSC_CMD	RW	0x00	Base + 0x02C	8
MSC_ARG	RW	0x00000000	Base + 0x030	32
MSC_RES	R	0x????	Base + 0x034	16
MSC_RXFIFO	R	0x????????	Base + 0x038	32
MSC_TXFIFO	W	0x????????	Base + 0x03C	32
MSC_LPM	RW	0x00000000	Base + 0x040	32

7.4.1 MMC/SD Control Register (MSC_CTRL)

MSC_CTRL **Base + 0x000**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SEND_CCSD	SEND_AD_CCSD	Reserved						EXIT_MULTIPLE	EXIT_TRANSFER	START_READWAIT	STOP_READWAIT	RESET	START_OP	STRPCL	
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15	SEND_CCSD	0: clear bit 1: Send Command Completion Signal Disable (CCSD) to CE_ATA device when set, host sends CCSD to CE_ATA device. Software set the bit only if current command is expecting CCS and interrupts are enabled in CE_ATA devices. Once the CCSD pattern is sent to device, host automatically clears the SEND_CCSD bit.	W
14	SEND_AS_CCSD	0: clear bit 1: send internally generated stop after sending CCSD to CE_ATA device When set, host automatically sends internally-generated STOP command(CMD12) to CE_ATA device. After sending CMD12, Auto Command Done (ACD) is set and generates interrupt to CPU. After sending the CCSD, controller automatically clears the SEND_AS_CCSD bit.	W
13:8	Reserved	Writing has no effect, read as zero.	R
7	EXIT_MULTIPLE	If CMD12 or CMD52 (I/O abort) is to be sent to terminate multiple block read/write in advance, set this bit to 1. 0: No effect 1: Exit from multiple block read/write	W
6	EXIT_TRANSFER	**Only used for SDIO suspend/resume and MMC stream read.** For SDIO, after suspend is accepted, set this bit with 1. For MMC, after the data of the expected number are received, set this bit with 1. 0: No effect 1: Exit from multiple block read/write after suspend is accepted, or exit from stream read	W
5	START_READWAIT	Only used for SDIO ReadWait. Start the ReadWait cycle. 0: No effect 1: Start ReadWait	W
4	STOP_READWAIT	Only used for SDIO ReadWait. Stop the ReadWait cycle. 0: No effect 1: Start ReadWait	W
3	RESET	Resets the MMC/SD controller. 0: No effect 1: Reset the MMC/SD controller	W
2	START_OP	This bit is used to start the new operation. When starting the clock, this bit can be 1. When stopping the clock, this bit can only be 0. 0: Do nothing 1: Start the new operation	W

1:0	CLOCK_CONTROL	These bits are used to start or stop clock. 00: Do nothing 01: Stop MMC/SD clock 10: Start MMC/SD clock 11: Reserved	W

7.4.2 MSC Status Register (MSC_STAT)

MSC_STAT **Base + 0x004**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	AUTO_CMD_DONE	Reserved															IS_RESETTING	SDIO_INT_ACTIVE	PRG_DONE	DATA_TRAN_DONE	END_CMD_RES	DATA_FIFO_AFULL	IS_READWAIT	CLK_EN	DATA_FIFO_FULL	DATA_FIFO_EMPTY	CRC_RES_ERR	CRC_READ_ERROR	CRC_WRITE_ERROR		TIME_OUT_RES	TIME_OUTREAD
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

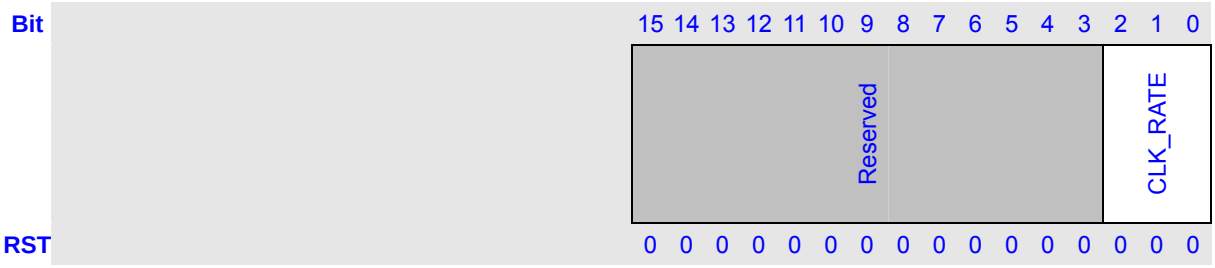
Bits	Name	Description	RW
31	AUTO_CMD_DONE	Indicate that the stop command (CMD12) that is internally generated by controller has finished.	R
30:16	Reserved	Writing has no effect, read as zero.	R
15	IS_RESETTING	MSC is resetting after power up or MSC_STRPCL[RESET] is written with 1. 0: Reset has been finished 1: Reset has not been finished	R
14	SDIO_INT_ACTIVE	Indicates whether an interrupt is detected at the SD I/O card. A separate acknowledge command to the card is required to clear this interrupt. 0: No interrupt detected 1: The interrupt from SDIO is detected	R
13	PRG_DONE	Indicates whether card has finished programming. 0: Card has not finished programming and is busy 1: Card has finished programming and is not busy	R
12	DATA_TRAN_DONE	Indicates whether data transmission to card has completed. 0: Data transmission to card has not completed 1: Data transmission to card has completed	R
11	END_CMD_RES	End command-response sequence or command sequence.	R

		0: Command and response/no-response sequence has not completed 1: Command and response/no-response sequence has completed	
10	DATA_FIFO_AFULL	Indicates whether data FIFO is almost full (The number of words >= 15). For reading data from card, use this bit. 0: Data FIFO is not full 1: Data FIFO is full	R
9	IS_READWAIT	Indicates whether SDIO card has entered ReadWait State. 0: Card has not entered ReadWait 1: Card has entered ReadWait	R
8	CLK_EN	Clock enabled. 0: Clock is off 1: Clock is on	R
7	DATA_FIFO_FULL	Indicates whether data FIFO is full. For reading data from card, do not use this bit, because it almost keeps to be 0. 0: Data FIFO is not full 1: Data FIFO is full	R
6	DATA_FIFO_EMPTY	Indicates whether data FIFO is empty. 0: Data FIFO is not empty 1: Data FIFO is empty	R
5	CRC_RES_ERR	Response CRC error. 0: No error on the response CRC 1: CRC error occurred on the response	R
4	CRC_READ_ERROR	CRC read error. 0: No error on received data 1: CRC error occurred on received data	R
3:2	CRC_WRITE_ERROR	CRC write error. 00: No error on transmission of data 01: Card observed erroneous transmission of data 10: No CRC status is sent back 11: Reserved	R
1	TIME_OUT_RES	Response time out. 0: Card response has not timed out 1: Card response has time out	R
0	TIME_OUT_READ	Read time out. 0: Card read data has not timed out 1: Card read data has timed out	R

7.4.3 MSC Clock Rate Register (MSC_CLKRT)

The MSC_CLKRT register specifies the frequency division of the MMC/SD bus clock. The software is responsible for setting this register.

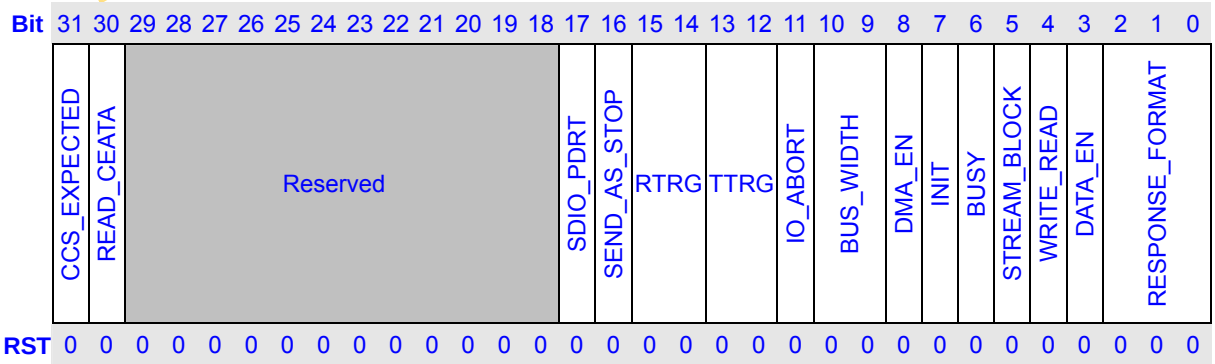
MSC_CLKRT **Base + 0x008**



Bits	Name	Description	RW
15:3	Reserved	Writing has no effect, read as zero.	R
2:0	CLK_RATE	Clock rate. 000: CLK_DIV 001: 1/2 of CLK_DIV 010: 1/4 of CLK_DIV 011: 1/8 of CLK_DIV 100: 1/16 of CLK_DIV 101: 1/32 of CLK_DIV 110: 1/64 of CLK_DIV 111: 1/128 of CLK_DIV	WR

7.4.4 MMC/SD Command and Data Control Register (MSC_CMDAT)

MSC_CMDAT **Base + 0x00C**



Bits	Name	Description	RW
31	CCS_EXPECTED	0: interrupts are not enabled in CE-ATA device, or commands does not expect CCS from device 1: interrupts are enabled in CE_ATA device, or RW_BLK	RW

		command expects command completion signal from device If the command expects Command Completion Signal (CCS) from the device, the software should set the control bit. It is auto cleared 0 by hardware.	
30	READ_CEATA	0: host is not performing read access (RW_BLK or RW_REG) towards CE_ATA device 1: host id performing read access (RW_BLK or RW_REG) towards CE_ATA device Software should set the bit to indicate that CE_ATA device is being accessed for read transfer. The bit is used to disable read data timeout indication while performing CE_ATA read transfers. It is auto cleared 0 by hardware.	RW
29:18	Reserved	Writing has no effect, read as zero.	R
17	SDIO_PRDT	Determine whether SDIO interrupt is 2 cycle or extend more cycle when data block last is transferred. 0: more cycle (like single block) 1: exact 2 cycle	RW
16	SEND_AS_STOP	0: no stop command sent at end of data transfer 1: send stop command at end of data transfer when stop command has finished, it is auto cleared 0 by hardware.	RW
15:14	RTRG	These bits set the receive FIFO half-empty threshold value, when the number of transmit FIFO >= threshold value , RXFIFO_RD_REQ will be set to 1. 00 : more than or equal to 8 01: more than or equal to 16 10: more than or equal to 24 11: reserved	RW
13:12	TTRG	These bits set the transmit FIFO half-empty threshold value, when the number of transmit FIFO < threshold value , TXFIFO_WR_REQ will be set to 1. 00 : less than 8 01: less than 16 10: less than 24 11: reserved	RW
11	STOP_ABORT	Specifies the current command is used to abort data transfer. 0: Nothing 1: The current command is used to abort transfer it is auto cleared 0 by hardware.	WR
10:9	BUS_WIDTH	Specifies the width of the data bus. 00: 1-bit	WR

		01: Reserved 10: 4-bit 11: 8bit	
8	DMA_EN	DMA mode enables. When DMA mode is used, this bit is also a mask on RXFIFO_RD_REQ and TXFIFO_WR_REQ interrupts. 0: Program I/O 1: DMA mode	WR
7	INIT	80 initialization clocks. 0: Do not precede command sequence with 80 clocks 1: Precede command sequence with 80 clocks	W
6	BUSY	Specifies whether a busy signal is expected after the current command. This bit is for no data command/response transactions only. 0: Not expect a busy signal 1: Expects a busy signal. If the response is R1b, then set it	WR
5	STREAM_BLOCK	Stream mode. 0: Data transfer of the current command sequence is not in stream mode 1: Data transfer of the current command sequence is in stream mode	WR
4	WRITE_READ	Specifies that the data transfer of the current command is a read or write operation. 0: Specifies that the data transfer of the current command is a read operation 1: Specifies that the data transfer of the current command is a write operation	WR
3	DATA_EN	Specifies whether the current command includes a data transfer. It is also used to reset RX_FIFO and TX_FIFO. 0: No data transfer with current command 1: Has data transfer with current command. It is also used to reset RX_FIFO and TX_FIFO	WR
2:0	RESPONSE_FORMAT	These bit specify the response format for the current command. 000: No response 001: Format R1 and R1b 010: Format R2 011: Format R3 100: Format R4 101: Format R5 110: Format R6 111: Format R7	WR

7.4.5 MMC/SD Response Time Out Register (MSC_RESTO)

MSC_RESTO **Base + 0x010**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								RES_TO							
RST	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bits	Name	Description	RW
15:8	Reserved	Writing has no effect, read as zero.	R
7:0	RES_TO	Specifies the number of MSC_CLK clock counts between the command and when the MMC/SD controller turns on the time-out error for the received response. The default value is 64.	WR

7.4.6 MMC/SD Read Time Out Register (MSC_RDTO)

MSC_RDTO **Base + 0x014**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	READ_TO																															
RST	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	RW
31:0	READ_TO	Specifies the number of clocks between the command and when the MMC/SD host controller turns on the time-out error for the received data. The unit is MSC_CLK.	WR

7.4.7 MMC/SD Block Length Register (MSC_BLKLEN)

MSC_BLKLEN **Base + 0x018**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	BLK_LEN															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:0	BLK_LEN	Specifies the number of bytes in a block, and is normally set to 0x200	WR

		for MMC/SD data transactions. The value Specified in the cards CSD.	

7.4.8 MSC/SD Number of Block Register (MSC_NOB)

MSC_NOB **Base + 0x01C**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NOB															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:0	NOB	Specifies the number of blocks in a data transfer. One block is a possibility.	WR

7.4.9 MMC/SD Number of Successfully-transferred Blocks Register (MSC_SNOB)

In block mode, the MSC_SNOB register records the number of successfully transferred blocks. If the last block has CRC error, this register also summaries it. It is used to query blocks for multiple block transfer.

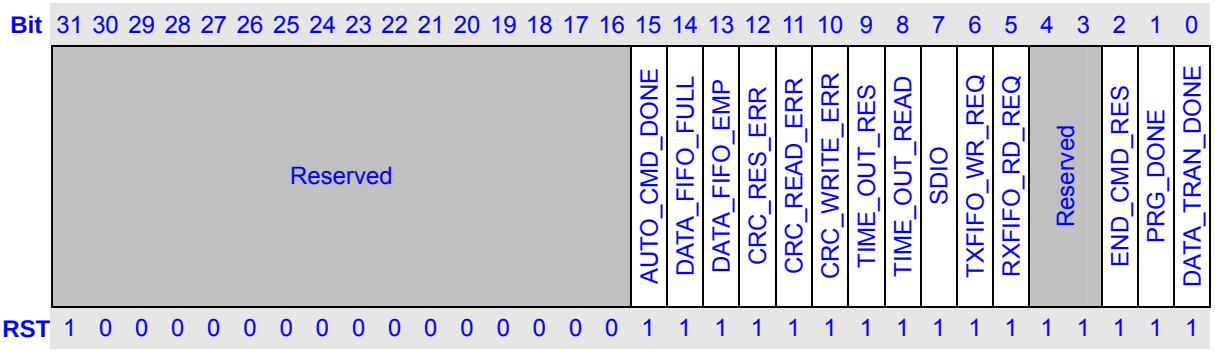
MSC_SNOB **Base + 0x020**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MSC_SNOB															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
15:0	MSC_SNOB	Specify the number of successfully transferred blocks for a multiple block transfer.	R

7.4.10 MMC/SD Interrupt Mask Register (MSC_IMASK)

MSC_IMASK **Base + 0x024**



Bits	Name	Description	RW
31:16	Reserved	Writing has no effect, read as zero.	R
15	AUTO_CMD_DONE	Mask the interrupt Auto Cmd Done (ACD). 0: Not masked 1: Masked	RW
14	DATA_FIFO_FULL	0: Not masked 1: Masked	RW
13	DATA_FIFO_EMP	0: Not masked 1: Masked	RW
12	CRC_RES_ERR	0: Not masked 1: Masked	RW
11	CRC_READ_ERR	0: Not masked 1: Masked	RW
10	CRC_WRITE_ERR	0: Not masked 1: Masked	RW
9	TIME_OUT_RES	0: Not masked 1: Masked	RW
8	TIME_OUT_READ	0: Not masked 1: Masked	RW
7	SDIO	Mask the interrupt from the SD I/O card. 0: Not masked 1: Masked	WR
6	TXFIFO_WR_REQ	Mask the Transmit FIFO write request interrupt. 0: Not masked 1: Masked	WR
5	RXFIFO_RD_REQ	Mask the Receive FIFO read request interrupt. 0: Not masked 1: Masked	WR
4:3	Reserved	Writing has no effect, read as zero.	R
2	END_CMD_RES	Mask the End command response interrupt.	WR

		0: Not masked 1: Masked	
1	PRG_DONE	Mask the Programming done interrupt. 0: Not masked 1: Masked	WR
0	DATA_TRAN_DONE	Mask the Data transfer done interrupt. 0: Not masked 1: Masked	WR

7.4.11 MMC/SD Interrupt Register (MSC_IREG)

The MSC_IREG register shows the currently requested interrupt. The FIFO request interrupts, TXFIFO_WR_REQ, and RXFIFO_RD_REQ are masked off with the DMA_EN bit in the MSC_CMDAT register. The software is responsible for monitoring these bit in program I/O mode.

MSC_IREG **Base + 0x028**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	AUTO_CMD_DONE	DATA_FIFO_FULL	DATA_FIFO_EMP	CRC_RES_ERR	CRC_READ_ERR	CRC_WRITE_ERR	TIME_OUT_RES	TIME_OUT_READ	SDIO	TXFIFO_WR_REQ	RXFIFO_RD_REQ	Reserved		END_CMD_RES	PRG_DONE	DATA_TRAN_DONE
RST	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15	AUTO_CMD_DONE	indicate Auto Cmd Done (ACD) interrupt. 0: the interrupt is not detected 1: the interrupt is detected	RW
14	DATA_FIFO_FULL	Indicate data FIFO is full interrupt. 0: the interrupt is not detected 1: the interrupt is detected	R
13	DATA_FIFO_EMP	Indicate data FIFO is empty interrupt. 0: the interrupt is not detected 1: the interrupt is detected	R
12	CRC_RES_ERR	Indicate response CRC error interrupt. 0: the interrupt is not detected 1: the interrupt is detected	RW
11	CRC_READ_ERR	Indicate CRC read error interrupt. 0: the interrupt is not detected 1: the interrupt is detected	RW
10	CRC_WRITE_ERR	Indicate CRC write error interrupt.	RW

		0: the interrupt is not detected 1: the interrupt is detected	
9	TIME_OUT_RES	Indicate response time out interrupt. 0: the interrupt is not detected 1: the interrupt is detected	RW
8	TIME_OUT_READ	Indicate read time out interrupt. 0: the interrupt is not detected 1: the interrupt is detected	RW
7	SDIO	Indicates whether the interrupt from SDIO is detected. 0: The interrupt from SDIO is not detected 1: The interrupt from SDIO is detected	R
6	TXFIFO_WR_REQ	Transmit FIFO write request. Set if data FIFO becomes half empty. (the number of words is < 8) 0: No Request for data Write to MSC_TXFIFO 1: Request for data write to MSC_TXFIFO	R
5	RXFIFO_RD_REQ	Receive FIFO read request. Set if data FIFO becomes half full (the number of words is >= 8) or the entries in data FIFO are the last read data. 0: No Request for data read from MSC_RXFIFO 1: Request for data read from MSC_RXFIFO	R
4:3	Reserved	Writing has no effect, read as zero.	R
2	END_CMD_RES	Indicates whether the command/response sequence has been finished. 0: The command/response sequence has not been finished 1: The command/response sequence has been finished Write 1 to clear.	WR
1	PRG_DONE	Indicates whether card has finished programming. 0: Card has not finished programming and is busy 1: Card has finished programming and is no longer busy Write 1 to clear.	WR
0	DATA_TRAN_DONE	Indicates whether data transfer is done. Note that for stream read/write, only when CMD12 (STOP_TRANS) has been sent, is this bit set. 0: Data transfer is not complete 1: Data transfer has completed or an error has occurred Write 1 to clear.	WR

7.4.12 MMC/SD Command Index Register (MSC_CMD)

MSC_CMD — Base + 0x02C

Bit	7	6	5	4	3	2	1	0
	Reserved		CMD_INDEX					
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:6	Reserved	Writing has no effect, read as zero.	R
5:0	CMD_INDEX	Specifies the command index to be executed.	WR

7.4.13 MMC/SD Command Argument Register (MSC_ARG)

MSC_ARG — Base + 0x030

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ARG																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:0	ARG	Specifies the argument for the current command.	WR

7.4.14 MMC/SD Response FIFO Register (MSC_RES)

The read-only MMC/SD Response FIFO register (RES_FIFO) holds the response sent back to the MMC/SD controller after every command. The size of this FIFO is 8 x 16-bit. The RES FIFO does not contain the 7-bit CRC for the response. The Status for CRC checking and response time-out status is in the status register, MSC_STAT.

The first halt-word read from the response FIFO is the most significant halt-word of the received response.

MSC_RES — Base + 0x034

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
15:0	DATA	Contains the response to every command that is sent by the MMC/SD controller. The size of this FIFO register is 8 x 16-bit.	R

7.4.15 MMC/SD Receive Data FIFO Register (MSC_RXFIFO)

The MSC_RXFIFO is used to read the data from a card. It is read-only to the software, and is read on 32-bit boundary. The size of this FIFO is 16 x 32-bit.

MSC_RXFIFO **Base + 0x038**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA																															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:0	DATA	One word of read data. The size of this FIFO is 16 x 32-bit.	R

7.4.16 MMC/SD Transmit Data FIFO Register (MSC_TXFIFO)

The MSC_TXFIFO is used to write the data to a card. It is write-only to the software, and is written on 32-bit boundary. The size of this FIFO is 16 x 32-bit.

MSC_TXFIFO **Base + 0x03C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA																															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:0	DATA	One word of write data. The size of this FIFO is 16 x 32-bit.	W

7.4.17 MMC/SD Low Power Mode Register (MSC_LPM)

The MSC_LPM is used to control whether MSC controller enters Low-Power Mode.

MSC_LPM **Base + 0x040**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	HSPEED	Reserved																														LPM
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:1	HSPEED	0: controller send CMD and data at clock falling 1: controller send CMD and data at clock rising	RW
30:1	Reserved	Writing has no effect, read as zero.	R
0	LPM	0 : Non –Low Power Mode 1: Low-Power Mode. Stop clock when card in idle (should be normally set to only MMC and SD cards. For SDIO cards, if interrupts must be detected, clock should not be stopped) When software sets the bit, MSC clock can auto be stopped. **NOTE:** when set the bit, the start_clock and stop clock can be not use.	RW

7.5 MMC/SD Functional Description

All communication between system and cards is controlled by the MSC. The MSC sends commands of two type: broadcast and addressed (point-to-point) commands.

Broadcast commands are intended for all cards, command like “Go_Idle_State”, “Send_Op_Cond”, “All_send_CID” and “Set_relative_Addr” are using way of broadcasting. During Broadcast mode, all cards are in open-drain mode, to avoid bus contention.

After Broadcast commands “Set_relative_Addr” issue, cards are enter standby mode, and Addressed command will be used from now on, in this mode, CMD/DAT will return to push-pull mode, to have maximum driving for maximum operation frequency.

The MMC and the SD are similar product. Besides the 4x bandwidth and the built-in encryption, they are being programmed similarly.

The MMC/SD controller (MSC) is the interface between the software and the MMC/SD bus. It is responsible for the timing and protocol between the software and the MMC/SD bus. It consists of control and status registers, a 16-bit response FIFO that is 8 entries deep, and one 32-bit receive/transmit data FIFOs that are 16 entries deep. The registers and FIFOs are accessible by the software.

MSC also enable minimal data latency by buffering data and generating and checking CRCs.

7.5.1 MSC Reset

The MMC/SD controller (MSC) can be reset by a hardware reset or software reset. All registers and FIFO controls are set to their default values after any reset.

7.5.2 MSC Card Reset

The command Go_Idle_State, CMD0 is the software reset command for MMC and SD Memory Card, and sets each card into Idle State regardless of the current card state; while in SDIO card, CMD52 is used to write IO reset in CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

7.5.3 Voltage Validation

All cards shall be able to establish communication with the host using any operation voltage in the maximal allowed voltage range specified in this standard. However, the support minimum and maximum values for Vdd are defined in Operation Conditions register (OCR) and many not cover the whole range. Cards that store the CID and CSD data in the payload memory would be able to communicate these information only under data transfer Vdd conditions. That means if host and card

have non compatible Vdd ranges, the card will not be able to complete the identification cycle, nor to send CSD data.

Therefore, a special command Send_Op_cont (CMD1 for MMC), SD_Send_Op_Cont (CMD41 for SD Memory) and IO_Send_Op_Cont (CMD5 for SDIO) are designed to provide a mechanism to identify and reject cards which do not match the Vdd range desired by the host. This is accomplished by the host sending the required Vdd voltage window as the operand of this command. Cards which can not perform data transfer in the specified range must discard themselves from further bus operations and go into Inactive State. By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the Inactive State. This query should be used if the host is able to select a common voltage range or if a notification to the application of non usable cards in the stack is desired.

7.5.4 Card Registry

Card registry on MCC and SD card are different.

For SD card, Identification process start at clock rate Fod, while CMD line output drives are push-pull drivers instead of open-drain. After the bus is activated the host will request the cards to send their valid operation conditions. The response to ACMD41 is the operation condition register of the card. The same command shall be send to all of the new cards in the system. Incompatible cards are sent into Inactive State. The host then issue the command All_Send_CID (CMD2) to each card and get its unique card identification (CID) number. Card that is unidentified, that is, which is in Ready State, send its CID number as the response. After the CID was sent by the card it goes into Identification State. Thereafter, the host issues Send_Relative_Addr (CMD3) asks the card to publish a new relative card address (RCA), which is shorter that CID and which will be used to address the card in the future data transfer mode. Once the RCA is received the card state changes to the Stand-by State. At this point, if the host wants that the card will have another RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card. The host repeats the identification process, that is, the cycles with CMD2 and CMD3 for each card in the system.

In MMC, the host starts the card identification process in open-drain mode with the identification clock rate Fod. The open drain driver stages on the CMD line allow parallel card operation during card identification. After the bus is active the host will request the cards to send their valid operation conditions (CMD1). The response to CMD1 is the ‘wired or’ operation on the condition restrictions of all cards in the system. Incompatible cards are sent into Inactive State. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards, that is, those which are in Ready State, simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bitstream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods stop sending their CID immediately and must wait for the next identification cycle. Since CID is unique for each card, only one card can be successfully send its full CID to the host. This card then

goes into Identification State. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign to this card a relative card address (RCA). Once the RCA is received the card state changes to the Stand-by State, and the card does not react to further identification cycles, and its output switches from open-drain to push-pull. The host repeat the process, which is CM2 and CMD3, until the host receive time-out condition to recognize completion of the identification process.

7.5.5 Card Access

7.5.5.1 Block Access, Block Write and Block Read

During block write (CMD24-27) one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. A card supporting block write shall always be able to accept a block of data defined by WRITE_BL_LEN. If the CRC fails, the card shall indicate the failure on the DAT line; the transferred data will be discarded and not written, and all further transmitted blocks (in multiple block write mode) will be ignored.

Programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card will report an error and not change any register contents. Some cards may require long and unpredictable times to write a block of data. After receiving a block of data and completing the CRC check, the card will begin writing and hold the DAT line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) at any time, and the card will respond with its status. The status bit READY_FOR_DATA indicates whether the card can accept new data or whether the write process is still in progress). The host may deselect the card by issuing CMD7 (to select a different card) which will displace the card into the Disconnect State and release the DAT line without interrupting the write operation. When reselecting the card, it will reactivate busy indication by pulling DAT to low if programming is still in progress and the write buffer is unavailable.

Block read is similar to stream read, except the basic unit of data transfer is a block whose maximizes is defined in the CSD (READ_BL_LEN). If READ_BL_PARTIAL is set, smaller blocks whose starting and ending address are entirely contained within one physical block (as defined by READ_BL_LEN) may also be transmitted. Unlike stream read, a CRC is appended to the end of each block ensuring data transfer integrity. CMD17 (READ_SINGLE_BLOCK) initiates a block read and after completing the transfer, the card returns to the Transfer state. CMD18 (READ_MULTIPLE_BLOCK) starts a transfer of several consecutive blocks. Blocks will be continuously transferred until a stop command is issued. If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed, the card shall detect a block misalignment at the beginning of the first mis-aligned block, set the ADDRESS_ERROR error bit in the status register, abort transmission and wait in the Data State for a stop command.

7.5.5.2 Stream Access, Stream Write and Stream Read (MMC Only)

Stream write (CMD20) starts the data transfer from the host to the card beginning from the starting address until the host issues a stop command. Since the amount of data to be transferred is not determined in advance, CRC can not be used. If the end of the memory range is reached while sending data and no stop command has been sent by the host, all further transferred data is discarded.

There is a stream oriented data transfer controlled by READ_DAT_UNTIL_STOP (CMD11). This command instructs the card to send its payload, starting at a specified address, until the host sends a STOP_TRANSMISSION command (CMD12). The stop command has execution delay due to the serial command transmission. The data transfer stops after the end bit of the stop command. If the end of the memory range is reached while sending data and no stop command has been sent yet by the host, the contents of the further transferred payload is undefined.

7.5.5.3 Erase, Group Erase and Sector Erase (MMC Only)

It is desirable to erase many sectors simultaneously in order to enhance the data throughput. Identification of these sectors is accomplished with the TAG_* commands. Either an arbitrary set of sectors within a single erase group, or an arbitrary selection of erase groups may be erase at one time, but not both together. That is, the unit of measure for determining an erase is either a sector or an erase group. If a set of sectors must be erased, all selected sectors must lie within the same erase group. To facilitate selection, a first command with the starting address is followed by a second command with the final address, and all sectors (or groups) within this range will be selected for erase.

7.5.5.4 Wide Bus Selection/Deselection

Wide Bus (4 bit bus width) operation mode may be selected / deselected using ACMD6. The default bus width after power up or GO_IDLE (CMD0) is 1 bit bus width. ACMD6 command is valid in ‘trans state’ only. That means the bus width may be changed only after a card was selected (CMD7).

7.5.6 Protection Management

Three write protect methods are supported in the host for Cards, Card internal write protect (Card’s responsibility), Mechanical write protect switch (Host responsibility only) and Password protection card lock operation.

7.5.6.1 Card Internal Write Protection

Card data may be protected against either erase or write. The entire card may be permanently write protected by the manufacturer or content provider by setting the permanent or temporary write protect bits in the CSD. For cards which support write protection of groups of sectors by setting the WP_GRP_SIZE sectors as specified in the CSD), and the write protection may be changed by the application. The SET_WRITE_PROT command sets the write protection of the addressed

write-protect group, and the CLR_WRITE_PROT command clears the write protection of the addressed write-protect group.

The SEND_WRITE_PROT command is similar to a single block read command. The card shall send a data block containing 32 write protection bits (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits. The address field in the write protect commands is a group address in byte units. The card will ignore all LSB's below the group size.

7.5.6.2 Mechanical write protect switch

A mechanical sliding tablet on the side of the card will be used by the user to indicate that a given card is write protected or not. If the sliding tablet is positioned in such a way that the window is open that means the card is write protected. If the window is close the card is not write protected.

A proper, matched, switch on the socket side will indicated to the host that the card is write protected or not. It is the responsibility of the host to protect the card. The position of the write protect switch is un-known to the internal circuitry of the card.

7.5.6.3 Password Protect

The password protection feature enables the host to lock a card while providing a password, which later will be used for unlocking the card. The password and its size is kept in an 128-bit PWD and 8-bit PWD_LEN registers, respectively. These registers are non-volatile so that a power cycle will not erase them.

Locked cards respond to (and execute) all commands in the basic command class (class 0) and "lock card" command class. Thus the host is allowed to reset, initialize, select, query for status, etc., but not to access data on the card. If the password was previously set (the value of PWD_LEN is not 0) will be locked automatically after power on. Similar to the existing CSD and CID register write commands the lock/unlock command is available in "trans_state" only. This means that it does not include an address argument and the card must be selected before using it. The card lock/unlock command has the structure and bus transaction type of a regular single block write command. The transferred data block includes all the required information of the command (password setting mode, PWD itself, card lock/unlock etc.). The following table describes the structure of the command data block.

Table 7-4 Command Data Block Structure

Byte #	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Rsv	Rsv	Rsv	Rsv	ERASE	LOCK_UNLOCK	CLR_PWD	SET_PWD
1	PWDS_LEN							
2	Password Data							
…								

PWDS_LEN + 1	

ERASE – 1 Defines Forced Erase Operation (all other bits shall be 0) and only the command byte is sent.

LOCK/UNLOCK – 1=Locks the card. 0=Unlock the card (note that it is valid to set this bit together with SET_PWD but it is not allowed to set it together with CLR_PWD).

CLR_PWD – 1=Clears PWD.

SET_PWD – 1=Set new password to PWD.

PWD_LEN – Defines the following password length (in bytes).

PWD – The password (new or currently used depending on the command).

The data block size shall be defined by the host before it send the card lock/unlock command. This will allow different password sizes.

The following paragraphs define the various lock/unlock command sequences:

Lock command sequences:

1 Setting the Password.

a Select a card (CMD7), if not previously selected already.

b Define the block length (CMD16), given by the 8bit card lock/unlock mode, the 8 bits password size (in bytes), and the number of bytes of the new password. In case that a password replacement is done, then the block size shall consider that both passwords, the old and the new one, are sent with the command.

c Send Card Lock/Unlock command with the appropriate data block size on the data line including 16-bit CRC. The data block shall indicate the mode (SET_PWD), the length (PWD_LEN) and the password itself. In case that a password replacement is done, then the length value (PWD_LEN) shall include both passwords, the old and the new one, and the PWD field shall include the old password (currently used) followed by the new password.

d In case that the sent old password is not correct (not equal in size and content) then LOCK_UNLOCK_FAILED error bit will be set in the status register and the old password does not change. In case that PWD matches the sent old password then the given new password and its size will be saved in the PWD and PWD_LEN fields, respectively.

NOTE:

the password length register (PWD_LEN) indicates if a password is currently set. When it equals 0 there is no password set. If the value of PWD_LEN is not equal to zero the card will lock itself after power up. It is possible to lock the card immediately in the current power session by setting the LOCK/UNLOCK bit (while setting the password) or sending additional command for card lock.

2 Reset the password.

a Select a card (CMD7), if not previously selected already.

b Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.

c Send the card lock/unlock command with the appropriate data block size on the data line including 16-bit CRC. The data block shall indicate the mode CLR_PWD, the length (PWD_LEN) and the password (PWD) itself (LOCK/UNLOCK bit is don't care). If the PWD and PWD_LEN is set to 0. If the password is not correct then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

3 Locking a card.

a Select a card (CMD7), if not previously selected already.

b Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of currently used password.

c Send the card lock/unlock command with the appropriate data block size on the data line including 16-bit CRC. The data block shall indicate the mode LOCK, the length (PWD_LEN) and the password (PWD) itself.

If the PWD content equals to the sent password then the card will be locked and the card-locked status bit will be set in the status register. If the password is not correct then LOCK_UNLOCK_FAILED error bit will be set in the status register.

NOTE:

it is possible to set the password and to lock the card in the same sequence. In such case the host shall perform all the required steps for setting the password (as described above) including the bit LOCK set while the new password command is sent. If the password was previously set (PWD_LEN is not 0), then the card will be locked automatically after power on reset. An attempt to lock a locked card or to lock a card that does not have a password will fail and the LOCK_UNLOCK_FAILED error bit will be set in the status register.

Unlock command sequences:

1 Unlocking the card.

a Select a card (CMD7), if not previously selected already.

b Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.

c Send the card lock/unlock command with the appropriate data block size on the data line including 16-bit CRC. The data block shall indicate the mode UNLOCK, the length (PWD_LEN) and the password (PWD) itself.

If the PWD content equals to the sent password then the card will be unlocked and the card-locked status bit will be cleared in the status register. If the password is not correct then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

NOTE:

the unlocking is done only for the current power session. As long as the PWD is not

cleared the card will be locked automatically on the next power up. The only way to unlock the card is by clearing the password. An attempt to unlock an unlocked card will fail and LOCK_UNLOCK_FAILED error bit will be set in the status register.

2 Forcing Erase.

In case that the user forgot the password (the PWD content) it is possible to erase all the card data content along with the PWD content. This operation is called Forced Erase.

a Select a card (CMD7), if not previously selected already.

b Define the block length (CMD16) to 1 byte (8bit card lock/unlock command). Send the card lock/unlock command with the appropriate data block of one byte on the data line including 16-bit CRC. The data block shall indicate the mode ERASE (the ERASE bit shall be the only bit set).

If the ERASE bit is not the only bit in the data field then the LCOK_UNLOCK_FAILED error bit will be set in the status register and the erase request is rejected. If the command was accepted then ALL THE CARD CONTENT WILL BE ERASED including the PWD and PWD_LEN register content and the locked card will get unlocked.

An attempt to force erase on an unlocked card will fail and LOCK_UNLOCK_FAILED error bit will be set in the status register.

7.5.7 Card Status

The response format R1 contains a 32-bit field named card status. This field is intended to transmit the card's status information (which may be stored in a local status register) to the host. If not specified otherwise, the status entries are always related to the previous issued command.

Table below defines the different entries of the status. The type and clear condition fields in the table are abbreviate as follows:

Type:

E: Error bit.

S: Status bit..

R: Detected and set for the actual command response.

X: Detected and set during command execution. The host must poll the card by issuing the status command in order to read these bits.

Clear Condition:

A: According to the card current state.

B: Always related to the previous command. Reception of a valid command will clear it (with a delay of one command).

C: Clear by read.

Table 7-5 Card Status Description

Bits	Identifier	Type	Description	Clear Condition
31	OUT_OF_RAGE	E R	The command's argument was out of the allowed range for this card. 0: No Error 1: Error	C
30	ADDRESS_ERROR	E R X	A misaligned address which did not match the block length was used in the command. 0: No Error 1: Error	C
29	BLOCK_LEN_ERROR	E R	The transferred block length is not allowed for this, or the number of transferred bytes does not match the block length. 0: No Error 1: Error	C
28	ERASE_SEQ_ERROR	E R	An error in the sequence of erase commands occurred. 0: No Error 1: Error	C
27	ERASE_PARAM	E X	An invalid selection of sectors or groups for erase occurred. 0: No Error 1: Error	C
26	WP_VIOLATION	E R X	Attempt to program a write protected block. 0: No Protected 1: Protected	C
25	CARD_IS_LOCKED	S X	When set, signals that the card is locked by the host. 0: Card unlocked 1: Card locked	A
24	LOCK_UNLOCK_FAILED	E R X	Set when a sequence or password error has been detected in lock/unlock card command or if there was an attempt to access a locked card. 0: No Error 1: Error	C

23	COM_CRC_ERROR	E R	The CRC check of the previous command failed. 0: No Error 1: Error	B
22	ILLEGAL_COMMAND	E R	Command not legal for the card state. 0: No Error 1: Error	B
21	CARD_ECC_FAILED	E X	Card internal ECC was applied but failed to correct the data. 0: normal 1: failure	C
20	CC_ERROR	E R X	Internal card controller error. 0: No Error 1: Error	C
19	ERROR	E R X	A general or an unknown error occurred during the operation. 0: No Error 1: Error	C
18	UNDERRUN	E X	The card could not sustain data transfer in stream read mode. 0: No Error 1: Error	C
17	OVERRUN	E X	The card could not sustain data programming in stream write mode. 0: No Error 1: Error	C
16	CID/CSD_OVERWRITE	E R X	Can be either one of the following errors. 0: No Error 1: Error	C
15	WP_ERASE_SKIP	S X	Only partial address space was erased due to existing write protected blocks. 0: No Protected 1 : Protected	C
14	CARD_ECC_DISABLED	S X	The command has been executed without using the internal ECC. 0: enabled 1: disabled	A

13	ERASE_RESET	S R	An erase sequence was cleared before executing because an out of erase sequence command was received. 0: normal 1: set	C
12:9	CURRENT_STATE	S X	The state of the card when receiving the command. If the command execution causes a state change, it will be visible to the host in the response to the next command. The four bits are interpreted as binary coded number between 0 and 15. 0: idle 1: ready 2: ident 3: stby 4: tran 5: data 6: rcv 7: prg 8 : dis (9 – 15) : rsv	B
8	READY_FOR_DATA	S X	Corresponds to buffer empty signaling on the bus. 0: No Ready 1: Ready	A
7:6	Reserved	-	-	-
5	APP_CMD	S R	The card will expect ACMD, or indication that the command has been interpreted as ACMD. 0: Disable 1: Enable	C
4:0	Reserved	-	-	-

7.5.8 SD Status

The SD status contains status bits that are related to the SD card proprietary features and may be used for future application specific usage. The size of the SD status is one data block of 512bit. The content of this register is transmitted to the Host over the DAT bus along with 16-bit CRC. The SD status is sent to the host over the DAT bus if ACMD13 is sent (CMD55 followed with CMD13). ACMD13 can be sent to a card only in ‘tran_state’ (card is selected). SD status structure is described in below.

The same abbreviation for *type* and *clear condition* were used as for the Card Status above.

Table 7-6 SD Status Structure

Bits	Identifier	Type	Description	Clear Condition
511:510	DAT_BUS_WIDTH	S R	Shows the currently defined data bus width that was defined by SET_BUS_WIDTH command. 00: 1 (default) 01: Reserved 10: 4 bit width 11: Reserved	A
509	SECURED_MODE	S R	Card is in Secured Mode of operation. 0: Not in the Mode 10: In the mode	A
508:496	Reserved.			
495:480	SD_CARD_TYPE	S R	All 0, is SD Memory cards.	A
479:448	SIZE_OF_PROTECTED_AREA	S R	Size of protected area.	A
447:312	Reserved.			
311:0	Reserved for manufacturer.			

7.5.9 SDIO

I/O access differs from memory in that the registers can be written and read individually and directly without a FAT file structure or the concept of blocks (although block access is supported). These registers allow access to the IO data, control of the IO function, and report on status or transfer I/O data to and from the host.

Each SDIO card may have from 1 to 7 functions plus one memory function built into it. A function is a self contained I/O device. I/O functions may be identical or completely different from each other. All I/O functions are organized as a collection of registers, and there is a maximum of 131,072 registers possible for each I/O function.

7.5.9.1 SDIO Interrupts

In order to allow the SDIO card to interrupt the host, and interrupt function is added to a pin on the SD interface. Pin number 8 which is used as DAT[1] when operating in the 4 bit SD mode is used to signal the card's interrupt to the host. The use of interrupt is optional for each card or function within a card. The SDIO interrupt is "level sensitive", that is, the interrupt line must be held active (low) until it is either recognized and acted upon by the host or de-asserted due to the end of the Interrupt Period.

Once the host has serviced the interrupt, it is cleared via an IO write to the appropriate bit in the CCCR. The interrupt output of all SDIO cards is active low. This host controller provides pull-up resistors on all data lines DAT[3:0].

As Pin 8 of the card is shared between the IRQ and DAT[1] use in the 4 bit SD mode, and interrupt shall only be sent by the card and recognized by the host during a specific time. The time that a low on Pin 8 will be recognized as an interrupt is defined as the Interrupt Period.

The host here will only sample the level of Pin 8 (DAT[1]/IRQ) into the interrupt detector during the Interrupt Period. At all other times, the host will ignore the level on Pin 8. Note that the Interrupt Period is applicable for both memory and IO operations. The definition of the Interrupt Period is different for operations with single block and multiple block data transfer.

7.5.9.2 SDIO Suspend/Resume

Within a multi-function SDIO or a Combo (Mix IO and Memory) card, there are multiple devices (I/O and memory) that must share access to the SD bus. In order to allow the sharing of access to the host among multiple devices, SDIO and combo cards can implement the optional concept of suspend/resume. In a card supports suspend/resume, the host may temporarily halt a data transfer operation to one function or memory (suspend) in order to free the bus for a higher priority transfer to a different function of memory. Once this higher-priority transfer is complete, the original transfer is re-started where it left off (resume). The host controller here is supported by all IO functions except zero, and the memory of a combo card, and can suspend multiple transactions and resume them in any order desired. IO function zero does not support suspend/resume.

The procedure used to perform the Suspend/Resume operation on the SD bus is:

- The host determines which function currently used the DAT[] line(s).
- The host requests the lower priority or slower transaction to suspend.
- The host checks for the transaction suspension to complete.
- The host begins the higher priority transaction.
- The host waits for the completion of the higher priority transaction.
- The host restores the suspended transaction.

7.5.9.3 SDIO Read Wait

The optional Read Wait (RW) operation is defined only for the SD 1-bit and 4-bit modes. The read wait operation allows a host to signal a card that it is doing a read multiple (CMD53) operation to temporarily stall the data transfer while allowing the host to send commands to any function within the SDIO device. To determine if a card supports the Read Wait protocol, the host must test capability bits in CCCR. The timing for Read Wait is base on the Interrupt Period.

7.5.10 Clock Control

The software should guarantee that the card identification process starts in open-drain mode with the clock rate fod (0 ~ 400khz). In addition, the software should also make the card into interrupt mode with fod (only for MMC). The commands that require fod are CMD0, CMD1, CMD2, CMD3, CMD5, CMD40 and ACMD41. In data transfer mode, the MSC controller can operate card with clock rate fpp (0 ~ 25Mhz).

7.5.11 Application Specified Command Handling

The MultiMediaCard/SD system is designed to provide a standard interface for a variety applications types. In this environment it is anticipate that there will be a need for specific customers/applications features. To enable a common way of implementing these features, two types of generic commands are defined in the standard: Application Specific Command, ACMD, and General Command, GEN_CMD.

GEN_CMD, this command, when received by the card, will cause the card to interpret the following command as an application specific command, ACMD. The ACMD has the same structure as of regular MultiMediaCard standard commands and it may have the same CMD number. The card will recognize it as ACMD by the fact that it appears after APP_CMD.

The only effect of the APP_CMD is that if the command index of the, immediately, following command has an ACMD overloading, the none standard version will used. If, as an example, a card has a definition for ACMD13 but not for ACMD7 then, if received immediately after APP_CMD command, Command 13 will be interpreted as the non standard ACMD13 but, command 7 as the standard CMD7.

In order to use one of the manufacturer specific ACMD's the host will:

1 Send APP_CMD. The response will have the APP_CMD bit (new status bit) set signaling to the host that ACMD is now expected.
2 Send the required ACMD. The response will have the APP_CMD bit set, indicating that the accepted command was interpreted as ACMD. If a non-ACMD is sent then it will be respected by the card as normal MultiMediaCard command and the APP_CMD bit in the Card Status stays clear.

If a non valid command is sent (neither ACMD nor CMD) then it will be handled as a standard MultiMediaCard illegal command error.

The bus transaction of the GEN_CMD is the same as the single block read or write commands (CMD24 or CMD17). The difference is that the argument denotes the direction of the data transfer (rather than the address) and the data block is not a memory payload data but has a vendor specific format and meaning.

The card shall be selected ('tran_state') before sending CMD56. The data block size is the

BLOCK_LEN that was defined with CMD16. The response to CMD56 will be R1b (card status + busy indication).

7.6 MMC/SD Controller Operation

7.6.1 Data FIFOs

The controller FIFOs for the response tokens, received data, and transmitted data are MSC_RES, MSC_RXFIFO, and MSC_TXFIFO, respectively. These FIFOs are accessible by the software and are described in the following paragraphs.

7.6.1.1 Response FIFO (MSC_RES)

The response FIFO, MSC_RES, contains the response received from an MMC/SD card after a command is sent from the controller. MSC_RES is a read-only, 16-bit, and 8-entry deep FIFO.

The FIFO will hold all possible response lengths. Responses that are only one byte long are located on the LSB of the 16-bit entry in the FIFO. The first half-word read from the response FIFO is the most significant half-word of the received response. For example, if the response format is R1, then the response read from RES_FIFO is bit [47:32], bit[31:16], bit[15:0] and in the third half-word only the low 8-bit is effective response [15:8] and the high 8-bit is ignored. If the response format is R2, then the response read from MSC_RES is bit [135:8] and needs reading 8 times.

The FIFO does not contain the response CRC. The status of the CRC check is in the status register, MSC_STAT.

7.6.1.2 Receive/Transmit Data FIFO (MSC_RXFIFO/MSC_TXFIFO)

The receive data FIFO and transmit data FIFO share one 16-entry x 32-bit FIFO, because at one time data are only received or are only transmitted. If it is used to receive data, it is called MSC_RXFIFO and read-only. If it is used to transmit data, it is called MSC_TXFIFO and write-only.

Data FIFO and its controls are cleared to a starting state after a system reset or at the beginning of the operations which include data transfer. (MSC_CMDAT[DATA_EN] == 1)

If at any time MSC_RXFIFO becomes full and the data transmission is not complete, the controller turns the MSC_CLK off to prevent any overflows. When the clock is off, data transmission from the card stops until the clock is turned back on. After MSC_RXFIFO is not full, the controller turns the clock on to continue data transmission. The full status of the FIFO is registered in the MSC_STAT [DATA_FIFO_FULL] bit.

If at any time MSC_TXFIFO becomes empty and the data transmission is not complete, the controller turns the MSC_CLK off to prevent any underrun. When the clock is off, data transmission to the card stops until the clock is turned back on. When MSC_TXFIFO is no longer empty, the controller automatically restarts the clock. The empty status of the FIFO is registered in the MSC_STAT [DATA_FIFO_EMPTY] bit.

The FIFO is readable on word (32-bit) boundaries. The max read/written number is 16 words. The controller can correctly process big-endian and little-endian data.

Because at the beginning of the operation which include data transfer (MSC_CMDAT [DATA_EN] == 1), Data FIFO and its controls are cleared, software should guarantee data in FIFO have been read/written before beginning a new command.

7.6.2 DMA and Program I/O

Software may communicate to the MMC controller via the DMA or program I/O.

To access MSC_RXFIFO/MSC_TXFIFO with the DMA, the software must program the DMA to read or write the FIFO with source port width 32-bit, destination port width 32-bit, transfer data size 32-byte, transfer mode single. For example, to write 64 bytes of data to the MSC_TXFIFO, the software must program the DMA as follows:

```
DMA_DCTRn = 2                 // Write 2 32-bytes (64 bytes)
DMA_DCCRn[SWDH] = 0           // source port width is 32-bit
DMA_DCCRn[DWDH] = 0           // destination port width is 32-bit
DMA_DCCRn[DS] = 4             // transfer data size is 32-byte
DMA_DCCRn[TM] = 4             // transfer mode is single
DMA_DCCRn[RDIL] = 0           // request detection interval length is 0
```

The number of 32-bytes should be calculated from the number of transferred bytes as follows:

```
The number of words = (The number of bytes + 31) / 32
```

If the number of transferred bytes is not the multiple of 4, the controller can correctly process endian.

The DMA trigger level is 8 words, that is to say, the DMA read trigger is when data words in MSC_RXFIFO is >= 8 and the DMA write trigger is when data words in MSC_TXFIFO is < 8. Software can also configure DMA registers based on requirements, but the above 32-byte transfer data size is most efficient.

With program I/O, the software waits for the MSC_IREG [RXFIFO_RD_REQ] or MSC_IREG [TXFIFO_WR_REQ] interrupts before reading or writing the respective FIFO.

NOTES:

1. The MSC_CMDAT [DMA_EN] bit must be set to a 1 to enable communication with the DMA and it must be set to a 0 to enable program I/O.
2. DMA can be enabled only after MSC_CMDAT is written, because MSC_CMDAT [DATA_EN] is used to reset TX/RXFIFO.

7.6.3 Start and Stop clock

The software stops the clock as follows:

1 Write MSC_STRPCL with 0x01 to stop the MMC/SD bus clock.
2 Wait until MSC_STAT[CLK_EN] becomes zero.

To start the clock the software writes MSC_STRPCL with 0x02.

7.6.4 Software Reset

Reset includes the MSC reset and the card reset.

The MSC reset is through MSC_STRPCL [RESET] bit.

The card reset is to make the card into idle state. CMD0 (GO_IDLE_STATE) sets the MMC and SD memory cards into idle state. CMD52 (IO_RW_DIRECT, with argument 0x88000C08) reset the SD I/O card. The MMC/SD card are initialized with a default relative card address (RCA = 0x0001 for MMC and RCA = 0x0000 for SD) and with a default driver stage register setting (lowest speed, highest driving current capability).

The following registers must be set before the clock is started:

Step 1. Stop the clock.
Step 2. Set MSC_STRPCL register to 0x08 to reset MSC.
Step 3. Wait while MSC_STAT [IS_RESETTING] is 1.
Step 4. Set MSC_CMD with CMD0.
Step 5. Update the MSC_CMDAT register as follows:
  a Write 0x0000 to MSC_CMDAT [RESPONSE_FORMAT].
  b Clear the MSC_CMDAT [DATA_EN] bit.
  c Clear the MSC_CMDAT [BUSY] bit.
  d Clear the MSC_CMDAT [INIT] bit.
Step 6. Start the clock.
Step 7. Start the operation. (write MSC_STRPCL with 0x04)
Step 8. Wait for the END_CMD_RES interrupt.
Step 9. Set MSC_CMD with CMD52.
Step 10. Set MSC_ARG with 0x88000C08.
Step 11. Update the MSC_CMDAT register as follows:
  a Write 0x005 to MSC_CMDAT [RESPONSE_FORMAT].
  b Clear the MSC_CMDAT [DATA_EN] bit.
  c Clear the MSC_CMDAT [BUSY] bit.
  d Clear the MSC_CMDAT [INIT] bit.
Step 12. Start the operation.
Step 13. Wait for the END_CMD_RES interrupt.

7.6.5 Voltage Validation and Card Registry

At most 10 MMC and 1 SD (either SDMEM or SDIO) can be inserted MMC/SD bus at the same time, and their voltage validation and card registry steps are different, so the software should be programmed as follows:

Step 1. Check whether SDIO card is inserted.
Step 2. Check whether SDMEM card is inserted.
Step 3. Check whether MMC cards are inserted.

7.6.5.1 Check SDIO

The commands are sent as follows:

Step 1. (Optional) Send CMD52 (IO_RW_DIRECT) with argument 0x88000C08 to reset SDIO card.
Step 2. Send CMD5 (IO_SEND_OP_CMD) to validate voltage.
Step 3. If the response is correct and the number of IO functions > 0, then continue, else go to check SDMEM.
Step 4. If C-bit in the response is ready (the initialization has finished), go to 6.
Step 5. Send CMD5 (IO_SEND_OP_CMD) to validate voltage, then go to 4.
Step 6. If memory-present-bit in the response is true, then it is a combo card (SDIO + Memory), else it is only a SDIO card.
Step 7. If it is a combo card, go to check SDMEM to initialize the memory part.
Step 8. Send CMD3 (SET_RELATIVE_ADDR) to let the card publish a RCA. The RCA is returned from the response.
Step 9. If do not accept the new RCA, go to 8, else record the new RCA.
Step 10. Go to check MMC, because we can assure that there is no SDMEM card.

7.6.5.2 Check SDMEM

If there is no SDIO card or there is a combo card, continue to check SDMEM.

The commands are sent as follows:

Step 1. (Optional) Send CMD0 (GO_IDLE_STATE) to reset MMC and SDMEM card. This command has no response.
Step 2. Send CMD55. Here the default RCA 0x0000 is used for CMD55.
Step 3. If the response is correct (CMD55 has response), then continue, else go to check MMC.
Step 4. Send ACMD41 (SD_SEND_OP_CMD) to validate voltage (the general OCR value is 0x00FF8000).
Step 5. If the initialization has finished, go to 7. (The response is the OCR register and it includes a status information bit (bit [31]). This status bit is set if the card power up procedure has been finished. As long as the card is busy, the corresponding bit[31] is set to LOW.)
Step 6. Send CMD55 and ACMD41 to validate voltage, and then go to 5.
Step 7. Send CMD2 (ALL_SEND_CID) to get the card CID.
Step 8. Send CMD3 (SET_RELATIVE_ADDR) to let card publish a RCA. The RCA is returned

from the response.

Step 9. If do not accept the new RCA, go to 8, else record the new RCA.

Step 10. Go to check MMC.

7.6.5.3 Check MMC

Because there may be several MMC card, so some steps (5 ~ 8) should be repeated several times.

The commands are sent as follows:

Step 1. Send CMD1 (SEND_OP_CMD) to validate voltage (the general OCR value is 0x00FF88000).

Step 2. If the response is correct, then continue, else goto 9.

Step 3. If the initialization has finished, go to 5. (The response is the OCR register and it includes a status information bit (bit [31]). This status bit is set if the card power up procedure has been finished. As long as the card is busy, the corresponding bit[31] is set to LOW.)

Step 4. Send CMD1 (SEND_OP_CMD) to validate voltage, and then go to 3.

Step 5. Send CMD2 (ALL_SEND_CID) to get the card CID.

Step 6. If the response timeout occurs, goto 9.

Step 7. Send CMD3 (SET_RELATIVE_ADDR) to assign the card a RCA.

Step 8. If there are other MMC cards, then go to 5.

Step 9. Finish.

7.6.6 Single Data Block Write

In a single block write command, the following registers must be set before the operation is started:

Step 1. Set MSC_NOB register to 0x0001.

Step 2. Set MSC_BLKLEN to the number of bytes per block.

Step 3. Update the MSC_CMDAT register as follows:

a Write 0x001 to MSC_CMDAT [RESPONSE_FORMAT].

b Write 0x2 to MSC_CMDAT [BUS_WIDTH] if the card is SD, else clear it.

c Set the MSC_CMDAT [DATA_EN] bit.

d Set the MSC_CMDAT [WRITE_READ] bit.

e Clear the MSC_CMDAT [STREAM_BLOCK] bit.

f Clear the MSC_CMDAT [BUSY] bit.

g Clear the MSC_CMDAT [INIT] bit.

Step 4. Start the operation.

Step 5. Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

Step 1. Wait for the MSC_IREG [END_CMD_RES] interrupt.

Step 2. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.

At the same time write data to the MSC_TXFIFO and continue until all of the data have been written to the FIFO.

Step 3. Wait for MSC_IREG [PROG_DONE] interrupt. This interrupt indicates that the card has finished programming. Certainly software may start another command sequence on a different card.

Step 4. Read the MSC_STAT register to verify the status of the transaction (i.e. CRC error status).

To address a different card, the software sends a select command to that card by sending a basic no data command and response transaction. To address the same card, the software must wait for MSC_IREG [PROG_DONE] interrupt. This ensures that the card is not in the busy state.

In addition, CMD26 (PROGRAM_CID), CMD27 (PROGRAM_CSD), CMD42 (LOCK/UNLOCK), CMD56 (GEN_CMD: write) and CMD53 (single_block_write) operations are similar to single block write.

7.6.7 Single Block Read

In a single block read command, the following registers must be set before the operation is started:

Step 1. Set MSC_NOB register to 0x0001.

Step 2. Set MSC_BLKLEN register to the number of bytes per block.

Step 3. Update the following bits in the MSC_CMDAT register:

a Write 0x001 to MSC_CMDAT [RESPONSE_FORMAT].

b Write 0x2 to MSC_CMDAT [BUS_WIDTH] if the card is SD, else clear it.

c Set the MSC_CMDAT [DATA_EN] bit.

d Clear the MSC_CMDAT [WRITE_READ] bit.

e Clear the MSC_CMDAT [STREAM_BLOCK] bit.

f Clear the MSC_CMDAT [BUSY] bit.

g Clear the MSC_CMDAT [INIT] bit.

Step 4. Start the operation.

Step 5. Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

Step 1. Wait for the MSC_IREG [END_CMD_RES] interrupt.

Step 2. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.

At the same time read data from the MSC_RXFIFO as data becomes available in the FIFO, and continue reading until all data is read from the FIFO.

Step 3. Read the MSC_STAT register to verify the status of the transaction (i.e. CRC error status).

In addition, CMD30 (SEND_WRITE_PROT), ACMD13 (SD_STATUS), CMD56 (GEN_CMD-read), ACMD51 (SEND_SCR) and CMD53 (single_block_read) are similar to single block read.

7.6.8 Multiple Block Write

The multiple block write mode is similar to the single block write mode, except that multiple blocks of

data are transferred. Each block is the same length. All the registers are set as they are for the single block write, except that the MSC_NOB register is set to the number of blocks to be written.

The multiple block write mode also requires a stop transmission command, CMD12, after the data is transferred to the card. After the MSC_IREG [DATA_TRAN_DONE] interrupt occurs, the software must program the controller register to send a stop data transmission command.

If multiple block write with pre-defined block count (refer to MMC spec v-3.3) is used, CMD12 should not be sent.

For SDIO card, CMD53 (multiple_block_write) is also similar, but when IO abort (CMD52) is sent, MSC_CMDAT [IO_ABORT] should be 1.

Table 7-7 How to stop multiple block write

Operation	Stop condition	Software processing
Open-ended or SDIO infinite	After write MSC_NOB blocks	1 Wait for DATA_TARN_DONE interrupt. 2 Send CMD12 or CMD52. (IO abort) 3 Wait for END_CMD_RES and PRG_DONE interrupt.
Open-ended or SDIO infinite	Stop writing in advance (not write MSC_NOB blocks)	1 Set MSC_STRPCL [EXIT_MULTIPLE]. 2 Wait for DATA_TRAN_DONE interrupt. 3 Send CMD12 or CMD52. (IO abort) 4 Wait for END_CMD_RES and PRG_DONE interrupt.
Predefined block or SDIO finite	After writing MSC_NOB blocks	1 Wait for DATA_TRAN_DONE interrupt.
Predefined block or SDIO finite	Stop writing in advance (not write MSC_NOB blocks)	1 Set MSC_STRPCL [EXIT_MULTIPLE]. 2 Wait for DATA_TRAN_DONE interrupt. 3 Send CMD12 or CMD52. (IO abort) 4 Wait for END_CMD_RES and PRG_DONE interrupt.

7.6.9 Multiple Block Read

The multiple blocks read mode is similar to the single block read mode, except that multiple blocks of data are transferred. Each block is the same length. All the registers are set as they are for the single block read, except that the MSC_NOB register is set to the number of blocks to be read.

The multiple blocks read mode requires a stop transmission command, CMD12, after the data from the card is received. After the MSC_IREG [DATA_TRAN_DONE] interrupt has occurred, the software must program the controller registers to send a stop data transmission command.

If multiple block read with pre-defined block count (refer to MMC spec v-3.3) is used, CMD12 should not be sent.

For SDIO card, CMD53 (multiple_block_read) is also similar, but when IO abort (CMD52) is sent, MSC_CMDAT [IO_ABORT] should be 1.

Table 7-8 How to stop multiple block read

Operation	Stop condition	Software processing
Open-ended or SDIO infinite	After reading MSC_NOB blocks	1 Wait for DATA_TRAN_DONE interrupt. 2 Send CMD12 or CMD52. (IO abort) 3 Wait for END_CMD_RES interrupt.
Open-ended or SDIO infinite	Stop reading in advance (not write MSC_NOB blocks)	1 Set MSC_STRPCL [EXIT_MULTIPLE]. 2 Wait for DATA_TRAN_DONE interrupt. 3 Send CMD12 or CMD52. (IO abort) 4 Wait for END_CMD_RES interrupt.
Predefined block or SDIO finite	After reading MSC_NOB blocks	1 Wait for DATA_TRAN_DONE interrupt.
Predefined block or SDIO finite	Stop reading in advance (not write MSC_NOB blocks)	1 Set MSC_STRPCL [EXIT_MULTIPLE]. 2 Wait for DATA_TRAN_DONE interrupt. 3 Send CMD12 or CMD52. (IO abort) 4 Wait for END_CMD_RES interrupt.

7.6.10 Stream Write (MMC)

In a stream write command, the following registers must be set before the operation is started:

1 Update MSC_CMDAT register as follows:
   a Write 0x001 to the MSC_CMDAT [RESPONSE_FORMAT].
   b Clear the MSC_CMDAT [BUS_WIDTH] because only MMC support stream write.
   c Set the MSC_CMDAT [DATA_EN] bit.
   d Set the MSC_CMDAT [WRITE_READ] bit.
   e Set the MSC_CMDAT [STREAM_BLOCK] bit.
   f Clear the MSC_CMDAT [BUSY] bit.
   g Clear the MSC_CMDAT [INIT] bit.
2 Start the operation.
3 Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

1 Wait for the MSC_IREG [END_CMD_RES] interrupt.
2 Write data to the MSC_TXFIFO and continue until all of the data is written to the Data FIFO.
3 Stop clock. Wait until MSC_STAT[CLK_EN] becomes 0. The clock must be stopped.
4 Set the command registers for a stop transaction command (CMD12) and other registers.

5 Start the clock and start the operation.
6 Wait for the MSC_IREG [END_CMD_ERS] interrupt.
7 Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
8 Wait for the MSC_IREG [PRG_DONE] interrupt. This interrupt indicates that the card has finished programming. Certainly software may start another command sequence on a different card.
9 Read the MSC_STAT register to verify the status of the transaction.

To address a different card, the software must send a select command to that card by sending a basic no data command and response transaction. To address the same card, the software must wait for MSC_IREG [PRG_DONE] interrupt. This ensures that the card is not in the busy state.

If partial blocks are allowed (if CSD parameter WRITE_BL_PARTIAL is set) the data stream can start and stop at any address within the card address space, otherwise it shall start and stop only at block boundaries. If WRITE_BL_PARTIAL is not set, 16 more stuff bytes need to be written after the useful written data, otherwise only write the useful written data.

7.6.11 Stream Read (MMC)

In a stream read command, the following registers must be set before the operation is turned on:

1 Update the MSC_CMDAT register as follows:
  a Write 0x01 to the MSC_CMDAT [RESPONSE_FORMAT].
  b Clear the MSC_CMDAT [BUS_WIDTH] because only MMC support stream read.
  c Clear the MSC_CMDAT [WRITE_READ] bit.
  d Set the MSC_CMDAT [STREAM_BLOCK] bit.
  e Clear the MSC_CMDAT [BUSY] bit.
  f Clear the MSC_CMDAT [INIT] bit.
2 Start the operation.
3 Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

1 Wait for the MSC_IREG [END_CMD_RES] interrupt.
2 Read data from the MSC_RXFIFO and continue until all of the expected data has been read from the FIFO.
3 Write MSC_STRPCL [EXIT_TRANSER] with 1. If MSC_STAT[DATA_FIFO_FULL] is 1, then read MSC_RXFIFO to make it not full. Because if data FIFO is full, MSC_CLK is stopped. Here, the data FIFO contains useless data.
4 Set the command registers for a stop transaction command (CMD12) and send it. There is no need to stop the clock.
5 Wait for the MSC_IREG [END_CMD_RES].
6 Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
7 Read the MSC_STAT register to verify the status of the transaction.

7.6.12 Erase, Select/Deselect and Stop

For CMD7 (SELECT/DESELECT_CARD), CMD12 (STOP_TRANSMISSION) and CMD38 (ERASE), the following registers must be set before the operation is started:

1 Update the MSC_CMDAT register as follows:
   a Write 0x01 to the MSC_CMDAT [RESPONSE_FORMAT].
   b Clear the MSC_CMDAT [DATA_EN] bit.
   c Clear the MSC_CMDAT [WRITE_READ] bit.
   d Clear the MSC_CMDAT [STREAM_BLOCK] bit.
   e Set the MSC_CMDAT [BUSY] bit.
   f Clear the MSC_CMDAT [INIT] bit.
2 Start the operation.
3 Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

1 Wait for the MSC_IREG [END_CMD_RES] interrupt.
2 Wait for the MSC_IREG [PRG_DONE] interrupt. If CMD12 is sent to terminate data read operation, then there is no need to wait for MSC_IREG [PRG_DONE] interrupt. This interrupt indicates that the card has finished programming. Certainly software may start another command sequence on a different card.

7.6.13 SDIO Suspend/Resume

The actual suspend/resume steps are as follows:

1 During data transfer, send CMD52 to require suspend. BR and RAW flag should be 1.
2 If BS flag in the response is 0, then suspend has been accepted and goto 4.
3 Send CMD52 to query card status. R flag should be 1. Go to 2.
4 Write MSC_STRPCL [EXIT_TRANSFER] with 1.
5 Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
6 Read MSC_NOB, MSC_SNOB and etc, save them into variables.
7 Set registers for high priority transfer and start it.
8 Wait until high priority transfer is finished.
9 Restore registers from variables, but MSC_NOB should be (MSC_NOB – MSC_SNOB).
10 Send CMD52 to require resume. FSx should be resumed function number.

7.6.14 SDIO ReadWait

The actual ReadWait steps are as follows:

1 During multiple block read, read MSC_SNOB. If MSC_SNOB is nearby or equal to MSC_NOB, no need to use ReadWait.
2 Write MSC_STRPCL [START_READWAIT] with 1.
3 Wait until MSC_STAT [IS_READWAIT] becomes 1.
4 Send CMD52 to query card status.
5 Write MSC_STRPCL [STOP_READWAIT] with 1.

7.6.15 Operation and Interrupt

The software can use polling-status method to operate the MMC/SD card, but this is not the proposed method, because its performance is very low. The proposed method is to use interrupt. Generally there are fixed necessary steps to finish each command. The steps are as follows:

1 (Optional) Stop clock. Poll CLK_EN.
2 Fill the registers (MSC_CMD, MSC_CMDAT, MSC_ARG, MSC_CLKRT, and etc).
3 (Optional) Start clock.
4 Start the operation. Wait for the MSC_IREG [END_CMD_RES] interrupt.
5 Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
6 Send STOP_TRANS (CMD12) or I/O abort (CMD52). Wait for the MSC_IREG [END_CMD_ERS] interrupt.
7 Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
8 Wait for the MSC_IREG [PRG_DONE] interrupt.

Table 7-9 The mapping between Commands and Steps

Index	Abbreviation	1	2	3	4	5	6	7	8	Comments
CMD0	GO_IDLE_STATE	Y	Y	Y	Y					
CMD1	SEND_OP_COND	Y	Y	Y	Y					
CMD2	ALL_SEND_CID	Y	Y	Y	Y					
CMD3	SET_RELATIVE_ADDR	Y	Y	Y	Y					
CMD4	SET_DSR	Y	Y	Y	Y					
CMD7	SELECT/DSELECT_CARD	Y	Y	Y	Y				Y	
CMD9	SEND_CID	Y	Y	Y	Y					
CMD10	SEND_CSD	Y	Y	Y	Y					
CMD11	READ_DAT_UNTIL_STOP	Y	Y	Y	Y		Y	Y		
CMD12	STOP_TRANSMISSION	Y	Y	Y	Y				Y	
CMD13	SEND_STATUS	Y	Y	Y	Y					
CMD15	GO_INACTIVE_STATE	Y	Y	Y	Y					
CMD16	SET_BLOCKLEN	Y	Y	Y	Y					
CMD17	READ_SINGLE_BLOCK	Y	Y	Y	Y	Y				
CMD18	READ_MULTIPLE_BLOCK	Y	Y	Y	Y	Y	Y			Open-ended
CMD18	READ_MULTIPLE_BLOCK	Y	Y	Y	Y	Y				Predefine blocks
CMD20	WRITE_DAT_UNTIL_STOP	Y	Y	Y	Y		Y	Y	Y	
CMD23	SET_BLOCK_COUNT	Y	Y	Y	Y					
CMD24	WRITE_SINGLE_BLOCK	Y	Y	Y	Y	Y			Y	
CMD25	WRITE_MULTIPLE_BLOCK	Y	Y	Y	Y	Y	Y		Y	Open-ended
CMD25	WRITE_MULTIPLE_BLOCK	Y	Y	Y	Y	Y			Y	Predefine blocks
CMD26	PROGRAM_CID	Y	Y	Y	Y	Y			Y	
CMD27	PROGRAM_CSD	Y	Y	Y	Y	Y			Y	

CMD28	SET_WRITE_PROT	Y	Y	Y	Y				Y	
CMD29	CLR_WRITE_PROT	Y	Y	Y	Y				Y	
CMD30	SEND_WRITE_PROT	Y	Y	Y	Y	Y				
CMD32	ERASE_WR_BLOCK_START	Y	Y	Y	Y					
CMD33	ERASE_WR_BLOCK_END	Y	Y	Y	Y					
CMD35	ERASE_GROUP_START	Y	Y	Y	Y					
CMD36	ERASE_GROUP_END	Y	Y	Y	Y					
CMD38	ERASE	Y	Y	Y	Y				Y	
CMD39	FAST_IO	Y	Y	Y	Y					
CMD40	GO_IRQ_STATE	Y	Y	Y	Y					
CMD42	LOCK/UNLOCK	Y	Y	Y	Y	Y			Y	
CMD55	APP_CMD	Y	Y	Y	Y					
CMD56	GEN_CMD	Y	Y	Y	Y	Y				Read
CMD56	GEN_CMD	Y	Y	Y	Y	Y			Y	Write
ACMD6	SET_BUS_WIDTH	Y	Y	Y	Y					
ACMD13	SD_STATUS	Y	Y	Y	Y	Y				
ACMD22	SEND_NUM_WR_BLOCKS	Y	Y	Y	Y					
ACMD23	SET_WR_BLOCK_COUNT	Y	Y	Y	Y					
ACMD41	SD_SEND_OP_COND	Y	Y	Y	Y					
ACMD42	SET_CLR_CARD_DETECT	Y	Y	Y	Y					
ACMD51	SEND_SCR	Y	Y	Y	Y	Y				

NOTE: For stream read/write, STOP_CMD is sent after finishing data transfer. For write, STOP_CMD is with the last six bytes. For read, STOP_CMD is sent after receiving data and card sends some data which MSC ignores.

8 UART Interface

8.1 Overview

This chapter describes the universal asynchronous receiver/transmitter (UART) serial ports. There are four UARTs: All UARTs use the same programming model. Each of the serial ports can operate in interrupt based mode or DMA-based mode.

The Universal asynchronous receiver/transmitter (UART) is compatible with the 16550-industry standard and can be used as slow infrared asynchronous interface that conforms to the Infrared Data Association (IrDA) serial infrared specification 1.1.

8.1.1 Features

- Full-duplex operation
- 5-, 6-, 7- or 8-bit characters with optional no parity or even or odd parity and with 1, 1½, or 2 stop bits
- 32x8 bit transmit FIFO and 32x11bit receive FIFO
- Independently controlled transmit, receive (data ready or timeout), line status interrupts
- Internal diagnostic capability Loopback control and break, parity, overrun and framing-error is provided
- Separate DMA requests for transmit and receive data services in FIFO mode
- Supports modem flow control by software or hardware
- Slow infrared asynchronous interface that conforms to IrDA specification

8.1.2 Pin Description

Table 8-1 UART Pins Description

Name	Type	Description
RxD	Input	Receive data input
TxD	Output	Transmit data output
CTS_	Input	Clear to Send — Modem Transmission enabled
RTS_	Output	Request to Send — UART Transmission request

NOTE: UART3, UART2, UART1, UART0 support RxD, TxD, RTS_, CTS_.

8.2 Register Descriptions

All UART register 32-bit access address is physical address. When ULCR.DLAB is 0, URBR, UTHR and UIER can be accessed; When ULCR.DLAB is 1, UDLLR and UDLHR can be accessed.

Table 8-2 UART Registers Description

Name	Description	RW	Reset Value	Address	Access Size
URBR0	UART Receive Buffer Register 0	R	0x??	0x10030000	8
UTHR0	UART Transmit Hold Register 0	W	0x??	0x10030000	8
UDLLR0	UART Divisor Latch Low Register 0	RW	0x00	0x10030000	8
UDLHR0	UART Divisor Latch High Register 0	RW	0x00	0x10030004	8
UIER0	UART Interrupt Enable Register 0	RW	0x00	0x10030004	8
UIIR0	UART Interrupt Identification Register 0	R	0x01	0x10030008	8
UFCR0	UART FIFO Control Register 0	W	0x00	0x10030008	8
ULCR0	UART Line Control Register 0	RW	0x00	0x1003000C	8
UMCR0	UART Modem Control Register 0	RW	0x00	0x10030010	8
ULSR0	UART Line Status Register 0	R	0x00	0x10030014	8
UMSR0	UART Modem Status Register 0	R	0x00	0x10030018	8
USPR0	UART Scratchpad Register 0	RW	0x00	0x1003001C	8
ISR0	Infrared Selection Register 0	RW	0x00	0x10030020	8
UMR0	UART M Register 0	RW	0x00	0x10030024	8
UACR0	UART Add Cycle Register 0	RW	0x00	0x10030028	16
URBR1	UART Receive Buffer Register 1	R	0x??	0x10031000	8
UTHR1	UART Transmit Hold Register 1	W	0x??	0x10031000	8
UDLLR1	UART Divisor Latch Low Register 1	RW	0x00	0x10031000	8
UDLHR1	UART Divisor Latch High Register 1	RW	0x00	0x10031004	8
UIER1	UART Interrupt Enable Register 1	RW	0x00	0x10031004	8
UIIR1	UART Interrupt Identification Register 1	R	0x01	0x10031008	8
UFCR1	UART FIFO Control Register 1	W	0x00	0x10031008	8
ULCR1	UART Line Control Register 1	RW	0x00	0x1003100C	8
UMCR1	UART Modem Control Register 1	RW	0x00	0x10031010	8
ULSR1	UART Line Status Register 1	R	0x00	0x10031014	8
UMSR1	UART Modem Status Register 1	R	0x00	0x10031018	8
USPR1	UART Scratchpad Register 1	RW	0x00	0x1003101C	8
ISR1	Infrared Selection Register 1	RW	0x00	0x10031020	8
UMR1	UART M Register 1	RW	0x00	0x10031024	8
UACR1	UART Add Cycle Register 1	RW	0x00	0x10031028	16
URBR2	UART Receive Buffer Register 2	R	0x??	0x10032000	8
UTHR2	UART Transmit Hold Register 2	W	0x??	0x10032000	8
UDLLR2	UART Divisor Latch Low Register 2	RW	0x00	0x10032000	8
UDLHR2	UART Divisor Latch High Register 2	RW	0x00	0x10032004	8

UIER2	UART Interrupt Enable Register 2	RW	0x00	0x10032004	8
UIIR2	UART Interrupt Identification Register 2	R	0x01	0x10032008	8
UFCR2	UART FIFO Control Register 2	W	0x00	0x10032008	8
ULCR2	UART Line Control Register 2	RW	0x00	0x1003200C	8
UMCR2	UART Modem Control Register 2	RW	0x00	0x10032010	8
ULSR2	UART Line Status Register 2	R	0x00	0x10032014	8
UMSR2	UART Modem Status Register 2	R	0x00	0x10032018	8
USPR2	UART Scratchpad Register 2	RW	0x00	0x1003201C	8
ISR2	Infrared Selection Register 2	RW	0x00	0x10032020	8
UMR2	UART M Register 2	RW	0x00	0x10032024	8
UACR2	UART Add Cycle Register 2	RW	0x00	0x10032028	16
URBR3	UART Receive Buffer Register 3	R	0x??	0x10033000	8
UTHR3	UART Transmit Hold Register 3	W	0x??	0x10033000	8
UDLLR3	UART Divisor Latch Low Register 3	RW	0x00	0x10033000	8
UDLHR3	UART Divisor Latch High Register 3	RW	0x00	0x10033004	8
UIER3	UART Interrupt Enable Register 3	RW	0x00	0x10033004	8
UIIR3	UART Interrupt Identification Register 3	R	0x01	0x10033008	8
UFCR3	UART FIFO Control Register 3	W	0x00	0x10033008	8
ULCR3	UART Line Control Register 3	RW	0x00	0x1003300C	8
UMCR3	UART Modem Control Register 3	RW	0x00	0x10033010	8
ULSR3	UART Line Status Register 3	R	0x00	0x10033014	8
UMSR3	UART Modem Status Register 3	R	0x00	0x10033018	8
USPR3	UART Scratchpad Register 3	RW	0x00	0x1003301C	8
ISR3	Infrared Selection Register 3	RW	0x00	0x10033020	8
UMR3	UART M Register 3	RW	0x00	0x10033024	8
UACR3	UART Add Cycle Register 3	RW	0x00	0x10033028	16

8.2.1 UART Receive Buffer Register (URBR)

The read-only URBR is corresponded to one level 11bit buffer in non-FIFO mode and a 32x11bit FIFO that holds the character(s) received by the UART. Bits in URBR are right justified when being configured to use fewer than eight bits, and the rest of most significant data bits are zeroed and the most significant three bits of each buffer are the status for the character in the buffer. If ULSR.DRY is 0, don't read URBR, otherwise wrong operation may occur.

URBR0, URBR1, URBR2, URBR3 — **0x10030000, 0x10031000, 0x10032000, 0x10033000 (DLAB = 0)**

Bit	7	6	5	4	3	2	1	0
	URBR							
RST	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
7:0	URBR	8-bit UART receive read data.	R

8.2.2 UART Transmit Hold Register (UTHR)

The write-only UTHR is corresponded to one leve 8 bit buffer in non-FIFO mode and a 32x8bit FIFO in FIFO mode that holds the data byte(s) to be transmitted next.

UTHR0, UTHR1, UTHR2, UTHR3 — **0x10030000, 0x10031000, 0x10032000, 0x10033000 (DLAB = 0)**

Bit	7	6	5	4	3	2	1	0
	UTHR							
RST	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
7:0	UTHR	8-bit UART transmit write hold data.	W

8.2.3 UART Divisor Latch Low/High Register (UDLLR / UDLHR)

UART Divisor Latch registers, UDLLR/UDLHR together compose the divisor for the programmable baud rate generator that can take the UART device clock and divide it by 1 to (2^{16} – 1).

The UART device source clock is EXCLK or EXCLK/2 that is determined by CPCCR.ECS. UDLHR/UDLLR stores the high/low 8-bit of the divisor respectively. Load these divisor latches during initialization to ensure that the baud rate generator operates properly. If both Divisor Latch registers are 0, the 16X clock stops.

If you don't set UMR and UACR, UART will work at normal mode with the specified frequency. The relationship between baud rate and the value of Divisor is shown by the formula when UMR and UACR are not set:

Baud Rate = (UART device clock) / (16 * **Divisor**)

UDLLR0, UDLLR1, UDLLR2, UDLLR3 — **0x10030000, 0x10031000, 0x10032000, 0x10033000 (DLAB = 1)**

Bit	7	6	5	4	3	2	1	0
	Divisor Latch Low 8-bit							
RST	0	0	0	0	0	0	0	0

UDLHR0, UDLHR1, UDLHR2, UDLHR3 — **0x10030004, 0x10031004, 0x10032004, 0x10033004 (DLAB = 1)**

Bit	7	6	5	4	3	2	1	0
	Divisor Latch High 8-bit							
RST	0	0	0	0	0	0	0	0

8.2.4 UART Interrupt Enable Register (UIER)

The UART Interrupt Enable Register (UIER) contains the interrupt enable bits for the five types of interrupts (receive data ready, timeout, line status, and transmit data request, and modem status) that set a value in UIIR.

UIER0, UIER1, UIER2, UIER3 — **0x10030004, 0x100301004, 0x10032004, 0x10033004 (DLAB = 0)**

Bit	7	6	5	4	3	2	1	0
	Reserved			RTOIE	MSIE	RLSIE	TDRIE	RDRIE
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:5	Reserved	Writing has no effect, read as zero.	R
4	RTOIE	**Receive Timeout Interrupt Enable.** 0: Disable the receive timeout interrupt 1: Enable the receive timeout interrupt Timeout means the URDR (FIFO mode) is not empty but no character has received for a period of time T: T (bits) = 4 X Word length + 12.	RW
3	MSIE	**Modem Status Interrupt Enable.** 0: Disable the modem status interrupt 1: Enable the modem status interrupt	RW
2	RLSIE	**Receive Line Status Interrupt Enable.** 0: Disable receive line status interrupt 1: Enable receive line status interrupt	RW
1	TDRIE	**Transmit Data Request Interrupt Enable.** 0: Disable the transmit data request interrupt 1: Enable the transmit data request interrupt	RW
0	RDRIE	**Receive Data Ready Interrupt Enable.** 0: Disable the receive data ready interrupt 1: Enable the receive data ready interrupt	RW

8.2.5 UART Interrupt Identification Register (UIIR)

The read-only UART Interrupt Identification Register (UIIR) records the prioritized pending interrupt source information. Its initial value after power-on reset is 0x01.

UIIR0, UIIR1, UIIR2, UIIR3 **0x10030008, 0x10031008, 0x10032008, 0x10033008**

Bit	7	6	5	4	3	2	1	0
	FFMSEL		Reserved		INID			INPEND
RST	0	0	0	0	0	0	0	1

Bits	Name	Description	RW
7:6	FFMSEL	**FIFO Mode Select.** 0b00: Non-FIFO mode 0b01: Reserved 0b10: Reserved 0b11: FIFO mode	R
5:4	Reserved	Writing has no effect, read as zero.	R
3:1	INID	**Interrupt Identifier.** These bits identify the current highest priority pending interrupt. INID: Description 0b000: Modem Status 0b001: Transmit Data Request 0b010: Receive Data Ready 0b011: Receive Line Status 0b100: Reserved 0b101: Reserved 0b110: Receive Time Out 0b111: Reserved See Table 8-3 for details.	R
0	INPEND	**Interrupt Pending.** 0: interrupt is pending 1: No interrupt pending	R

Table 8-3 UART Interrupt Identification Register Description

UIIR.INID	Interrupt Set/Clear Cause			
	Priority	Type	Source	Clear Condition
0b0001	—	None	No pending interrupt	—

0b0110	1st Highest	Receive Line Status	Overrun, Parity, Frame Error, Break Interrupt, and FIFO Error (DMA mode only)	Reading ULSR or empty all the error characters in DMA mode
0b0100	2nd Highest	Receive Data Ready	FIFO mode: Trigger threshold was reached Non-FIFO mode: URBR full	FIFO mode: Reading URBR till below trigger threshold. Non-FIFO mode: Empty URBR
0b1100	2nd Highest	Receive Timeout	FIFO mode only: URBR not empty but no data read in for a period of time	Reset receive buffer by setting UFCR.RFRT to 1 or Reading URBR
0b0010	3rd Highest	Transmit Data Request	FIFO mode: Empty location in UTHR equal to half or more than half Non-FIFO mode: UTHR empty	FIFO mode: Data number in UTHR more than half Non-FIFO mode: Writing UTHR
0b0000	4th Highest	Modem Status	Modem CTS_ pin status change	Reading UMSR

8.2.6 UART FIFO Control Register (UFCR)

The write-only register UFCR contains the control bits for receive and transmit FIFO.

UFCR0, UFCR1, UFCR2, UFCR3 **0x10030008, 0x10031008, 0x10032008, 0x10033008**

Bit	7	6	5	4	3	2	1	0
	RDTR		Reserved	UME	DME	TFRT	RFRT	FME
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:6	RDTR	**Receive Buffer Data Number Trigger.** These bits are used to select the trigger level for the receive data ready interrupt in FIFO mode. 0b00: 1 0b01: 8 0b10: 16 0b11: 24	W
5	Reserved	Writing has no effect, read as zero.	R
4	UME	**UART Module Enable.** 0: Disable UART 1: Enable UART	W

3	DME	**DMA Mode Enable.** 0: Disable DMA mode 1: Enable DMA mode	W
2	TFRT	**Transmit Holding Register Reset.** 0: Not reset 1: Reset transmit FIFO	W
1	RFRT	**Receive Buffer Reset.** 0: Not reset 1: Reset receive FIFO	W
0	FME	**FIFO Mode Enable.** Set this bit before the trigger levels. 0: non-FIFO mode 1: FIFO mode	W

8.2.7 UART Line Control Register (ULCR)

The ULCR defines the format for UART data transmission.

ULCR0, ULCR1, ULCR2, ULCR3 — **0x1003000C, 0x1003100C, 0x1003200C, 0x1003300C**

Bit	7	6	5	4	3	2	1	0
	DLAB	SBK	STPAR	PARM	PARE	SBLS	WLS	
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7	DLAB	**Divisor Latch Access Bit.** 0: Enable to access URBR, UTHR or UIER 1: Enable to access UDLLR or UDLHR	W
6	SBK	**Set Break.** Causes a break condition (at least one 0x00 data) to be transmitted to the receiving UART. Acts only on the TXD pin and has no effect on the transmit logic. 0: No effect on TXD output 1: Forces TXD output to 0	W
5	STPAR	**Sticky Parity.** Setting this bit forces parity location to be opposite of PARM bit when PARE is 1 (it is ignored when PARE is 0). 0: Disable Sticky parity 1: Enable Sticky parity (opposite o f PARM bit)	W
4	PARM	**Parity Odd/Even Mode Select.**	W

		If PARE = 0, PARM is ignored. 0: Odd parity 1: Even parity	
3	PARE	**Parity Enable.** Enables a parity bit to be generated on transmission or checked on reception. 0: No parity 1: Parity	W
2	SBLS	**Stop Bit Length Select.** Specifies the number of stop bits transmitted and received in each character. When receiving, the receiver checks only the first stop bit. 0: 1 stop bit 1: 2 stop bits, except for 5-bit character then 1-1/2 bits	W
1:0	WLS	**Word Length Select.** 0b00: 5-bit character 0b01: 6-bit character 0b10: 7-bit character 0b11: 8-bit character	W

8.2.8 UART Line Status Register (ULSR)

The read-only ULSR indicates status information during the data transfer. Receive error information in ULSR[4:1] remains set until software reads ULSR and it must be read before the error character is read.

ULSR0, ULSR1, ULSR2, ULSR3 **0x10030014, 0x10031014, 0x10032014, 0x10033014**

Bit	7	6	5	4	3	2	1	0
	FIFOE	TEMP	TDRQ	BI	FMER	PARER	OVER	DRY
RST	0	1	1	0	0	0	0	0

Bits	Name	Description	RW
7	FIFOE	**FIFO Error Status. (FIFO mode only)** FIFOE is set when there is at least one kind of receive error (parity, frame, overrun, break) for any of the characters in receive buffer. FIFOE is reset when all error characters are read out of the buffer. During DMA transfer, the error interrupt generates when FIFOE is 1, and no receive DMA request generates even when data in receive buffer reaches the trigger threshold until all the error characters are read out. In non-DMA mode, FIFOE set does not generate error interrupt.	R

		0: No error data in receive buffer or non-FIFO mode 1: One or more error character in receive buffer	
6	TEMP	**Transmit Holding Register Empty.** Set when both UTHR and shift register are empty. It is cleared when either the UTHR or the shift register contains a data character. 0: There is data in the transmit shifter and UTHR 1: All the data in the transmit shifter and UTHR has been shifted out	R
5	TDRQ	**Transmit Data Request.** Set when UTHR has half or more empty location (FIFO mode) or empty (non-FIFO mode). When both UIER.TDRIE and TDRQ are 1, transmit data request interrupt generates or during DMA transfer, DMA request to the DMA controller generates when UIER.TDRIE is 0 and TDRQ is 1. 0: There is one (non-FIFO mode) or more than half data (FIFO mode) in UTHR 1: None data (non-FIFO mode) or half or less than half data (FIFO mode) in UTHR	R
4	BI	**Break Interrupt.** BI is set when the received data input is held low for longer than a full-word transmission time (the total time of start bit + data bits + parity bit + stop bits). BI is cleared when the processor reads the ULSR. In FIFO mode, only one character equal to 0x00 is loaded into the FIFO regardless of the length of the break condition. BI shows the break condition for the character at the front of the FIFO, not the most recently received character. 0: No break signal has been received 1: Break signal received	R
3	FMER	**Framing Error.** Set when the bit following the last data bit or parity bit is detected to be 0. If the ULCR had been set for two or one and half stop bits, the other stop bits are not checked except the first one. In FIFO mode, FMER shows a framing error for the character at the front of the receive buffer, not for the most recently received character. Cleared when the processor reads the ULSR. 0: No framing error 1: Invalid stop bit has been detected	R
2	PARER	**Parity Error.** Indicates that the received data character does not have the correct even or odd parity, as selected by the even parity select bit. PARER is set upon detection if a parity error and is cleared when the processor reads the ULSR. In FIFO mode, PARER shows a parity error for the character	R

		at the front of the FIFO, not the most recently received character. 0: No parity error 1: Parity error has occurred	
1	OVER	**Overrun Error.** Set when both receive buffer and shifter are full and new data is received which will be lost. Cleared when the processor reads the ULSR. 0: No data has been lost 1: Receive data has been lost	R
0	DRY	**Data Ready.** Set when a complete incoming character has been received into the Receive Buffer registers. DRY is cleared when the receive buffer is read (non-FIFO mode) or when the buffer is empty or when the buffer is reset by setting UFCR.RFRT to 1. 0: No data has been received 1: Data is available in URBR	R

8.2.9 UART Modem Control Register (UMCR)

The UMCR uses the modem control pins RTS_ and CTS_ to control the interface with a modem or data set. UMCR also controls the loopback mode. Loopback mode must be enabled before the UART is enabled.

UMCR0, UMCR1, UMCR2, UMCR3 **0x10030010, 0x10031010, 0x10032010, 0x10033010**

Bit	7	6	5	4	3	2	1	0
	MDCE	FCM	Reserved	LOOP	Reserved		RTS	Reserved
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7	MDCE	**Modem Control Enable.** 0: Modem function is disabled 1: Modem function is enabled	W
6	FCM	**Flow Control Mode.** 0: Flow control by software 1: Flow control by hardware	
5	Reserved	Writing has no effect, read as zero.	R
4	LOOP	**Loop Back.** This bit is used for diagnostic testing of the UART. When LOOP is 1, TXD output pin is set to a logic 1 state, RXD is disconnected from the pin, and the output of the transmitter shifter register is looped back into the	W

		receiver shift register input internally, similar to CTS_ and RTS_ pins and the RTS bit of the UMCR is connected to CTS bit of UMSR respectively. Loopback mode must be selected before the UART is enabled. 0: Normal operation mode 1: Loopback-mode UART operation	
3:2	Reserved	Writing has no effect, read as zero.	R
1	RTS	**Request To Send.** This bit can control the RTS_ output state. 0: RTS_ force to high 1: RTS_ force to low	W
0	Reserved	Writing has no effect, read as zero.	R

8.2.10 UART Modem Status Register (UMSR)

The read-only UMSR provides the current state of the control lines from the modem to the processor. They are cleared when the processor reads UMSR.

UMSR0, UMSR1, UMSR2, UMSR3 **0x10030018, 0x10031018, 0x10032018, 0x10033018**

Bit	7	6	5	4	3	2	1	0
	Reserved			CTS	Reserved			CCTS
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:5	Reserved	Writing has no effect, read as zero.	R
4	CTS	**Status of Clear To Send.** When MDCE bit is 1, this bit is the complement of CTS_ input. If Loop bit of UMCR is 1, this bit is equivalent to RTS bit of UMCR. 0: CTS_ pin is 1 1: CTS_ pin is 0	R
3:1	Reserved	Writing has no effect, read as zero.	R
0	CCTS	**Change status of CTS_.** When MDCE bit is 1, this bit indicates the state change on CTS_ pin. 0: No state change on CTS_ pin since last read of UMSR 1: A change occurs on the state of CTS_ pin	R

8.2.11 UART Scratchpad Register

This Scratchpad register is used as a scratch register for the programmer and has no effect on the UART.

USPR0, USPR1, USPR2, USPR3 — **0x1003001C, 0x1003101C, 0x1003201C, 0x1003301C**

Bit	7	6	5	4	3	2	1	0
	Scratch Data							
RST	?	?	?	?	?	?	?	?

8.2.12 Infrared Selection Register (ISR)

The ISR is used to configure the slow-infrared (SIR) interface that is provided in each UART to support two-way wireless communication using infrared transmission that conforms to the IrDA serial infrared specification 1.1. The maximum frequency is up to 115.2kbps.

ISR0, ISR1, ISR2, ISR3 — **0x10030020, 0x10031020, 0x10032020, 0x10033020**

Bit	7	6	5	4	3	2	1	0
	Reserved			RDPL	TDPL	XMODE	RCVEIR	XMITIR
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:5	Reserved	Writing has no effect, read as zero.	R
4	RDPL	**Receive Data Polarity.** 0: Slow-infrared (SIR) interface decoder takes positive pulses as zeros 1: SIR decoder takes negative pulses as zeros	W
3	TDPL	**Transmit Data Polarity.** 0: SIR encoder generates a positive pulse for a data bit of zero 1: SIR encoder generates a negative pulse for a data bit of zero	W
2	XMODE	**Transmit Pulse Width Mode.** Set when the transmit encoder needs to generate 1.6us pulses (that are 3/16 of a bit-time at 115.2 kbps). Cleared when the transmit encoder needs to generate 3/16 of a bit-time wide according to current baud rate. 0: Transmit pulse width is 3/16 of a bit-time wide 1: Transmit pulse width is 1.6 us	W
1	RCVEIR	**Receiver SIR Enable.** This bit is used to select the signal from the RXD pin is processed by the	W

		IrDA decoder before it is fed to the UART (RCVEIR = 1) or bypass IrDA decoder and is fed directly to the UART (RCVEIR = 0). 0: Receiver is in UART mode 1: Receiver is in SIR mode	
0	XMITIR	**Transmitter SIR Enable.** This bit is used to select TXD output pin is processed by the IrDA encoder before it is fed to the device pin (XMITIR = 1) or bypass IrDA encoder and is fed directly to the device pin (XMITIR = 0). **NOTE:** disable infrared LED before XMITIR is set, otherwise a false start bit may occur. 0: Transmitter is in UART mode 1: Transmitter is in SIR mode	W

8.2.13 UART M Register (UMR)

UMR0, UMR1, UMR2, UMR3 **0x10030024, 0x10031024, 0x10032024, 0x10033024**

Bit	7	6	5	4	3	2	1	0
	Reserved			M				
RST	0	0	0	0	0	0	0	0

M is the value of UMR register.
It will take UART at least M cycles to transmit or receive one bit.
It will take UART at most M+1 cycles to transmit or receive one bit.

8.2.14 UART Add Cycle Register (UACR)

UACR0, UACR1, UACR2, UACR3 **0x10030028, 0x10031028, 0x10032028, 0x10033028**

Bit	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				AC							
RST	0	0	0	0	0	0	0	0	0	0	0	0

If nth bit of the register is 1, it will take UART M+1 cycles to transmit or receive the bit of date.
If the register is 12'h0, UART will receive or transmit a bit by M cycle(s).
If the register is 12'hfff, UART will receive or transmit a bit by M+1 cycle(s).
For the detail to see For any frequency clock to use the Uart.

8.3 Operation

The following sections describe the UART operations that include flow of configuration, data transmission, data reception, and Infrared mode.

8.3.1 UART Configuration

Before UART starts to transfer data or changing transfer format, configuration must be done to define the transfer format. The sample flow is as the following:

In FIFO mode, set FME bit of UFCR to 1, reset receive and transmit FIFO, then initialize the UART as described below:

1 Clear UFCR.UME to 0.
2 Set value in UDLL/UDHR to generate the baud rate clock.
3 Set data format in ULCR.
4 If it is in FIFO MODE, set FME bit and other FIFO control in UFCR, reset receive and transmit FIFO, otherwise skip item 4.
5 Set each interrupt enable bit in UIER in interrupt-based transfer or set UFCR.DME in DMA-based transfer (DMA transfer is FIFO mode only), then set UFCR.UME.

8.3.2 Data Transmission

After configuration, UART is ready for data transfer. For data transmission, refer to the following procedure:

1 Read ULSR.TDRQ (interrupt disable) or wait for transmit data request interrupt (interrupt enable), if TDRQ = 1 or transmit data request interrupt generates, that means there is enough empty location in UTHR for new data.
2 If ULSR.TDRQ is 1 or get the transmit data request interrupt, write transmit data to UTHR to start transmission.
3 Do item 1 and item 2 if there are more data waiting for transmit.
4 After all necessary data are written to UTHR, wait ULSR.TEMP = 1, that means all data completely transmitted.
5 If it is necessary to send break, set ULCR.SBK and at least wait for 1-bit interval time to send a valid break, then clear ULCR.SBK.
6 Clear UME bit to finish UART transmission.

8.3.3 Data Reception

After configuration, UART is ready for data transfer. For data reception, refer to the following sample procedure:

1 Read ULSR.DRY (interrupt disable) or wait for receive data request interrupt (interrupt enable), if ULSR.DRY =1 or receive data request interrupt generates, that means URBR has one data (non-FIFO mode) or data in URBR reaches the trigger value. (FIFO mode)
2 If ULSR.DRY = 1 or receive data request interrupt generates, then read ULSR.FIFOE or see if

there is error interrupt, if FIFOE = 1, it means received data has receive error, then go to error handler, other wise go to item 3.

3 Read one received data in URBR (non-FIFO mode) or data equal to trigger value in URBR. (FIFO mode)

4 Check whether all data received: check whether ULSR.DRY = 0, in FIFO mode and interrupt is enabled, timeout interrupt may generate, when timeout interrupt generates, read URBR till ULSR.DRY = 0.

5 Clear UFCR.UME to end data reception when all data are received and ULSR.DRY = 0.

8.3.4 Receive Error Handling

A sample error handling flow is as the following:

1 If ULSR.FIFOE = 1, it means there is receive error in received data, then check what error it is.

2 If ULSR.OVER = 1, go to OVER error handling.

3 If ULSR.BI = 1, go to Break handling.

4 If ULSR.FMER = 1, go to Frame error handling.

5 If PARER = 1, go to PARER error handling.

8.3.5 Modem Transfer

When UMCR.MDCE = 1, modem control is enabled. Transfer flow can be stopped and restarted by software through RTS_ and CTS_ pin. When UART transmitter detects low level on CTS_ pin, it stops transmission and TxD pin goes to mark state after finishing transmitting the current character until it detects CTS_ pin goes back to high level. RTS_ pin is output to receiving UART and its state can be controlled by setting UMCR.RTS bit, that is, setting UMCR.RTS to 1, RTS_ pin is low level output that means UART is ready to receive data, on the contrary, it means UART currently can't receive more data.

8.3.6 DMA Transfer

UART can operate in DMA-based (UFCR.DME = 1, FIFO mode only), that is, dma request initiated by UART takes the place of interrupt request and transmission/reception is carried out using DMA instead of CPU. Be sure that software guarantee to disable transmit and receive interrupt except timeout and error interrupts.

During DMA transfer, if an interrupt occurs, software must first read the ULSR to see if an error interrupt exists, then check the UIIR for the source of the interrupt and if DMA channel is already halt because of the error indicator from UART, then disable DMA channel and read out all the error data from receive FIFO. Software re-set and re-enable DMA and data transfer by DMA will re-start.

8.3.7 Slow IrDA Asynchronous Interface

Each UART supports slow infra-red (SIR) transmission and reception by setting ISR.XMITIR and ISR.RCVEIR to 1 (make sure the two bits are not set to 1 at the same time because SIR can't operate full-duplex). According to the IrDA 1.1, data rate is limited at a maximum value of 115.2Kbps.

In SIR transmit mode, the transmit pulse comes out at a rate of 3/16 (when the transmit data bit is zero); in SIR receive mode, the receiver must detect the 3/16 pulsed period to recognize a zero value (an active high or low pulse is demodulation to 0, and no pulse is demodulation to 1).

Compared to normal UART, there are some limitations to SIR, that is, each character is fixed to 8-bit data width, no parity and 1 stop bit and modem function is ignored. The IrDA 1.1 specifies a minimum 10ms latency after an optical node ceases transmitting before its receiver recovers its receiving function and software must guarantee this delay.

In the IrDA 1.1 specification, communication must start up at the rate of 9600bps, but then allows the link to negotiate higher (or lower) data rates if supported by both ends. However, the communication rate will not automatically change. Change, if necessary, is performed by software.

8.3.8 For any frequency clock to use the UART

NOTE: if you don't set M register and UACR the UART work at normal mode with the specified frequencies. To use other frequency you should to set M register and UACR to right value.

1 The Improving

Following changes are made:

a One bit is composed by M CLK_{BR} cycles, which can be 4~1024.

b Some extra CLK_{BR} cycles can be inserted in some bits in one frame, so that like M has fraction.

For instance:

$CLK_{BR} = CLK_{DEV} / N$ N = 1, 2, …

$CLK_{BR} = CLK_{DEV}$ = 4MHz

Band rate = 460800

In accurate

$M_a = 8.681$

We take

M = 8, with 8 extra cycles in every frame

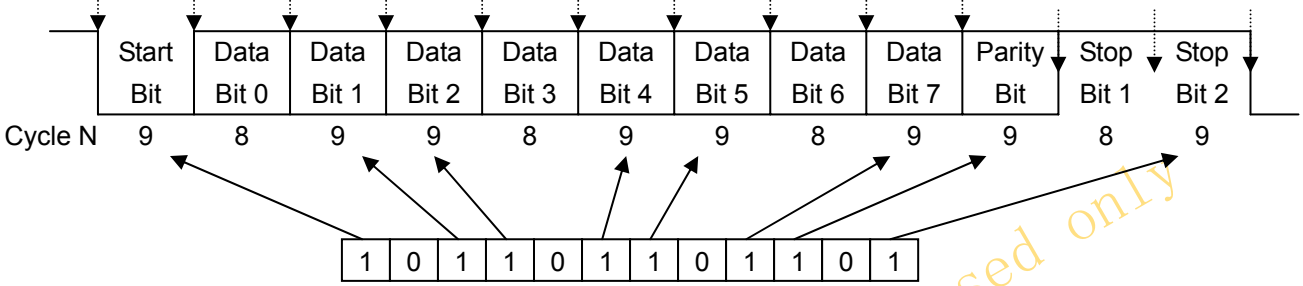
A 12-bit register is used to indicate where to insert the extra cycles.

The first line is the time expected

The second line is the time actual

The third line is the time error

Time expected	0	2.17	4.34	4.34	6.51	8.68	10.85	13.02	15.19	17.36	19.53	21.70	23.87
Time actual	0	2.25	4.25	4.25	6.5	8.75	10.75	13	15.25	17.25	19.5	21.75	23.75
Time error	0	0.08	0.09	0.09	0.01	0.07	0.10	0.02	0.06	0.11	0.03	0.05	0.12



For transmission, in theory, the biggest error is half of CLK_{BR} cycle, which is 0.125us here.

2 To set UMR register

$CLK_{BR} = CLK_{DEV} / N$

$M_a = CLK_{BR}$/band rate

M is modem of M_a.

Write M to Mregister.

Considering the power and the robust quality, for M form 6 to 32 is you better select by set the UDLR.

The max error

$$\frac{0.5/CLK_{BR}}{M_a/CLK_{BR}} = 0.5/M_a < 0.5/M$$

M	4	8	16	32	64
error/W_{bit}	12.5%	6.25%	3.125%	1.56%	0.78%

3 To set UACR value

For each bit of it means:

0: means not to add additional cycle to the bit that UART is prepare to transmit or receive, in another word, you will to use M cycles to transmit or receive the bit

1: means to add additional cycle to the bit that UART is prepare to transmit or receive, in another word, you will to use M+1 cycles to transmit or receive the bit

To set UACR value you must ensure that the max error of each bit should be less than $0.5P_{BR}$.

For example：M_a-M =0.15; M+1-M_a=0.85;

Write 8 to UMR,

Write 0x408 to UACR

cycle/bit	：	M，	M，	M，	M+1，	M，	M，	M，	M，	M，	M，	M+1，	M
UACR	：	0	0	0	1	0	0	0	0	0	0	1	0

9 Smart Card Controller

9.1 Overview

Smart Card Controller (SCC) interface is a primary device and communications interface for kinds of IC cards. The SCC interface supports communication with smart cards as specified in standard ISO7816-3. There are two SCC interfaces integrated in this SOC. And they perform the same functions. The SCC interface supports T=0 and T=1 protocol defined in ISO7816-3.

Software controls the session between the SCC interface and the card by SCC registers. Choosing protocol type and parameters, receiving and sending a byte to/from the card, activating/deactivating the card, and similar operations are accomplished with read/write operations to SCC registers. Transforming byte convention (inverse to direct and vice-versa, according to the session convention) is performed within SCC. Hence, software does not have to perform format inversion before character receipt. The SCC interface provides functionality to support the above standards, but it is the responsibility of software to ensure the standards are met.

Features:

- Supports normal card and UIM card
- 8-bit, 16-level receive-/transmit- FIFO
- Supports asynchronous character (T=0) communication modes
- Supports asynchronous block (T=1) communication modes
- Supports setting of clock-rate conversion factor F (372, 512, 558, etc.), and bit-rate adjustment factor D (1, 2, 4, 8, 16, 32, 12, 20, etc.)
- Supports extra guard time waiting
- Auto-error detection in T=0 receive mode
- Auto-character repeat in T=0 transmit mode
- Transforms inverted format to regular format and vice versa
- Support stop clock function in some power consuming sensitive applications

9.2 Pin Description

Table 9-1 Smart Card Controller Pins Description

Name	I/O	Description
SCC_CLK	Output	Serial clock connects SCC and the card
SCC_DAT	Input/Output	Data communication pin

9.3 Register Description

Table 9-2 Smart Card Controller Registers Description

Name	RW	Reset Value	Address	Access Size
SCCDR	RW	0x??	0x10041000	8
SCCFDR	R	0x00	0x10041004	8
SCCCR	RW	0x00000000	0x10041008	32
SCCSR	RW	0x8000	0x1004100C	16
SCCTFR	RW	0x0173	0x10041010	16
SCCEGTR	RW	0x00	0x10041014	8
SCCECR	RW	0x00000000	0x10041018	32
SCCRTOR	RW	0x00	0x1004101C	8

9.3.1 Transmit/Receive FIFO Data Register (SCCDR)

SCCDR 0x10041000

Bit	7	6	5	4	3	2	1	0
	DR							
RST	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
7:0	DR	Data port of HW FIFO.	RW

9.3.2 FIFO Data Count Register (SCCFDR)

SCCFDR 0x10041004

Bit	7	6	5	4	3	2	1	0
	Reserved			R				
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:5	Reserved	Writing has no effect, read as zero.	R
4:0	R	Characters resisted in FIFO.	R

9.3.3 Control Register (SCCCR)

SCCCR 0x10041008

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SCCE	TRS	T2R	Reserved			FDIV		FLUSH	Reserved					TRIG		TP	CONV	TXIE	RXIE	TENDIE	RTOIE	ECIE	EPIE	RETIE	EOIE	Reserved		TSEND	PX		CLKSTP
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	SCCE	Enables or disables SCC. When disable it, the SCC_CLK will be stopped.	RW
30	TRS	Transmit or Receive Select. 0: Reception mode; 1: Transmission mode.	RW
29	T2R	Auto-T2R support. T2R means Transmit turn to Reception. 0: controlled by SW; 1: controlled by HW.	RW
28:26	Reserved	Writing has no effect, read as zero.	R
25:24	FDIV	Frequency Divider Select. 00: Same as device clk 01: Half of device clk 10: ¼ of device clk 11: Reserved	RW
23	FLUSH	Flush FIFO. 0: Does not empty the Rx/Tx FIFO; 1: Empty the Rx/Tx FIFO.	RW
22:18	Reserved	Writing has no effect, read as zero.	R
17:16	TRIG	Receive/Transmit FIFO trigger. 00: 1 01: 4 10: 8 11: 14	RW
15	TP	Communicate protocol. 0: (T=0); 1: (T=1).	RW
14	CONV	Card data transfer convention. 0: LSB first; 1: MSB first and inverted.	RW
13	TXIE	Tx FIFO counter meets the trigger value interrupt enable bit.	RW
12	RXIE	Rx FIFO counter meets the trigger value interrupt enable bit.	RW
11	TENDIE	Transmission finished interrupt enable bit. (Both FIFO and transmitter are empty)	RW
10	RTOIE	Reception timeout interrupt enable bit.	RW
9	ECIE	ETU counter overflow interrupt enable bit.	RW
8	EPIE	Parity error interrupt enable bit.	RW
7	RETIE	Re-transmitting 3 times interrupt enable bit.	RW
6	EOIE	Receive overrun error interrupt enable bit.	RW
5:4	Reserved	Writing has no effect, read as zero.	R

3	TSEND	Receive TS character stage finished bit. 0: TS character has not been received; 1: TS character has been received.	RW
2:1	PX	Parameter X. SCC stop clock mode selection. 00 – Does not support SCC clock stop 01 – SCC_CLK stops at state low 10 – SCC_CLK stops at state high 11 – Reserved	RW
0	CLKSTP	SCC clock stop. 0:SCC has left or is leaving clock stop mode; 1: SCC has entered or is entering clock stop mode.	RW

9.3.4 Status Register (SCCSR)

SCCSR 0x1004100C

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TRANS	Reserved		ORER	RTO	PER	TFTG	RFTG	TEND	Reserved		RETR_3	Reserved			ECNTO
RST	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15	TRANS	Transfer status. Specifies whether the transfer is stopped or not.	R
14:13	Reserved	Writing has no effect, read as zero.	R
12	ORER	Receive overrun error.	RW
11	RTO	Reception timeout.	R
10	PER	Parity error.	RW
9	TFTG	Hit Tx FIFO trigger.	R
8	RFTG	Hit Rx FIFO trigger.	R
7	TEND	Transmission end. Both Tx FIFO and transmitter are empty.	RW
6:5	Reserved	Writing has no effect, read as zero.	R
4	RETR_3	Re-transmit exceed 3 times.	RW
3:1	Reserved	Writing has no effect, read as zero.	R
0	ECNTO	ETU counter overflow.	RW

9.3.5 Transmission Factor Register (SCCTFR)

SCCTFR 0x10041010

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved					FD										
RST	0	0	0	0	0	0	0	1	0	1	1	1	0	0	1	1

Bits	Name	Description	RW
15:11	Reserved	Writing has no effect, read as zero.	R
10:0	FD	Value of F/D. The initial value is 0x173(371).	RW

9.3.6 Extra Guard Timer Register (SCCEGTR)

SCCEGTR 0x10041014

Bit	7	6	5	4	3	2	1	0
	EGTR							
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:0	EGTR	The register is corresponding with N value of ISO7816-3. Which indicates the extra-guard time of a transmission?	RW

9.3.7 ETU Counter Value Register (SCCECR)

SCCECR 0x10041018

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved												EC																			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:20	Reserved	Writing has no effect, read as zero.	R
19:0	EC	ETU counter. Write operation will clear the internal counter automatically.	RW

9.3.8 Reception Timeout Register (SCCRTOR)

SCCRTOS								0x1004101C
Bit	7	6	5	4	3	2	1	0
	RTO							
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:0	RTO	Retry times when parity error detected.	RW

10 TS Slave Interface (TSSI)

10.1 Overview

The TS Slave Interface (TSSI) in JZ4770 is used to connect DTV Demodulator. It supports MPEG-2 Transport Stream (TS) as its input.

Features:

- Support both parallel mode and serial mode for TS data transfer
- TSDI0 or TSDI7 can be used to transfer data in serial mode
- The order of data in one byte supports LSB at first or MSB at first
- The order of data in one word supports LSB at first or MSB at first
- Input control signals and data can be either active high or active low
- Support using either positive or negative edge of TSCLK
- Support PID filtering function
- Up to 33 PID filters can be used when PID filtering function is enabled
- Support adding data 0 before or after transport stream
- Support master DMA transmission

10.2 Pin Description

Table 10-1 TSSI Pin Description

Name	I/O	Description
TSDI0~7	I	TS data bus
TSFAIL	I	TS packet uncorrectable
TSCLK	I	TS clock
TSFRM	I	TS data valid
TSSTR	I	TS packet start

10.3 Register Description

In this section, we will describe the registers in TSSI. Following table lists all the register definitions. All registers' 32bit addresses are physical addresses. And detailed function of each register will be described below.

Table 10-2 TSSI Register Description

Name	Description	RW	Reset Value	Address	Access Size
TSENA	TSSI Enable Register	RW	0x08	0x134E0000	8
TSCFG	TSSI Configure Register	RW	0x04FF	0x134E0004	16
TSCTRL	TSSI Control Register	RW	0x03	0x134E0008	8
TSSTAT	TSSI State Register	RW	0x00	0x134E000C	8
TSFIFO	TSSI FIFO Register	R	0x????????	0x134E0010	32
TSPEN	TSSI PID Enable Register	RW	0x00000000	0x134E0014	32
TSNUM	TSSI Data Number Register	RW	0x00	0x134E0018	8
TSDTR	TSSI Data Trigger Register	RW	0x7F	0x134E001C	8
TSPID0	TSSI PID Filter Register 0	RW	0x00000000	0x134E0020	32
TSPID1	TSSI PID Filter Register 1	RW	0x00000000	0x134E0024	32
TSPID2	TSSI PID Filter Register 2	RW	0x00000000	0x134E0028	32
TSPID3	TSSI PID Filter Register 3	RW	0x00000000	0x134E002C	32
TSPID4	TSSI PID Filter Register 4	RW	0x00000000	0x134E0030	32
TSPID5	TSSI PID Filter Register 5	RW	0x00000000	0x134E0034	32
TSPID6	TSSI PID Filter Register 6	RW	0x00000000	0x134E0038	32
TSPID7	TSSI PID Filter Register 7	RW	0x00000000	0x134E003C	32
TSPID8	TSSI PID Filter Register 8	RW	0x00000000	0x134E0040	32
TSPID9	TSSI PID Filter Register 9	RW	0x00000000	0x134E0044	32
TSPID10	TSSI PID Filter Register 10	RW	0x00000000	0x134E0048	32
TSPID11	TSSI PID Filter Register 11	RW	0x00000000	0x134E004C	32
TSPID12	TSSI PID Filter Register 12	RW	0x00000000	0x134E0050	32
TSPID13	TSSI PID Filter Register 13	RW	0x00000000	0x134E0054	32
TSPID14	TSSI PID Filter Register 14	RW	0x00000000	0x134E0058	32
TSPID15	TSSI PID Filter Register 15	RW	0x00000000	0x134E005C	32
TSDDA	TSSI DMA Descriptor Address	RW	0x00000000	0x134E0060	32
TSDTA	TSSI DMA Target Address	R	0x00000000	0x134E0064	32
TSDID	TSSI DMA Identifier	R	0x00000000	0x134E0068	32
TSDCMD	TSSI DMA Command	R	0x00000000	0x134E006C	32
TSDST	TSSI DMA Status	RW	0x00000000	0x134E0070	32
TSTC	TSSI Transfer Control Register	RW	0x00000000	0x134E0074	32

10.3.1 TSSI Enable Register (TSENA)

The register TSENA is used to trigger TSSI to work.

TSENA **0x134E0000**

Bit	7	6	5	4	3	2	1	0
	SFT_RST	Reserved		FAIL	PEN_0	PID_EN	DMA_EN	ENA
RST	0	0	0	0	1	0	0	0

Bits	Name	Description	RW
7	SFT_RST	TSSI FIFO software-reset. Set it to 1 and later it will be cleared by hardware auto. 0: Stop reset 1: Start reset	RW
6:5	Reserved	Writing has no effect, read as zero.	R
4	FAIL	0:FAIL signal is decided by TSSI Demodulator 1:FAIL signal is equal to 1	
3	PEN_0	Choose PID filter enable for PID=0. 0: not enable 1: enable	RW
2	PID_EN	Enable / disenable the PID filtering function. 0: PID filtering function is not enabled 1: PID filtering function is enabled	RW
1	DMA_EN	Enable / disenable the DMA mode. 0: DMA mode is not enabled (CPU only) 1: DMA mode is enabled	RW
0	ENA	Enable / disenable the TSSI module. 0: TSSI is not enabled 1: TSSI is enabled	RW

10.3.2 TSSI Configure Register (TSCFG)

The register TSCFG is used to configure the TSSI.

TSCFG **0x134E0004**

Bit	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	F_TRIGV			Reserved		TRANS_MD		EDN_WD	EDN_BT	TSDI_H	USE_0	TSCLK_CH	PARAL	TSCLK_P	TSFRM_H	TSSTR_H	TSFAIL_H
RST	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1	0

Bits	Name	Description	RW
16:14	F_TRIGV	Specify the trigger value of FIFO. F_TRIGV / Description 0 – Trigger Value is 4 1 – Trigger Value is 8 2 – Trigger Value is 16 3 – Trigger Value is 32 4 – Trigger Value is 48 5 – Trigger Value is 64 6 – Trigger Value is 80 7 – Trigger Value is 96 Trigger value should be greater than Expected Transfer Size.	RW
13:12	Reserved	Writing has no effect, read as zero.	R
11:10	TRANS_MD	Choose the mode of adding data 0. TRANS_MD / Description 0 – Add data 0 before transport stream 1 – Add data 0 after transport stream 2 – Do not add data 0 3 – Reserved	RW
9	EDN_WD[*1]	The order of data in word.	RW
8	EDN_BT[*1]	The order of data in byte.	RW
7	TSDI_H	Choose the polarity of TSDI0~7. 0: TSDI0~7 is active low 1: TSDI0~7 is active high	RW
6	USE_0	USE_0 is only used in SERIAL mode (TSCFG.PARAL=0). 0: Use TSDI7 to transfer data 1: Use TSDI0 to transfer data	RW
5	TSCLK_CH	Choose how to use TSCLK. 0: When $f_{pclk} > 3f_{TSCLK}$ 1: When $f_{pclk} > 2f_{TSCLK}$	RW
4	PARAL	Choose the working mode of TSSI.	RW

		0: Serial Mode 1: Parallel Mode	
3	TSCLK_P	This bit is used to determine which edge of TSCLK is used when TSSI is sampling data. 0: Use the negative edge of TSCLK 1: Use the positive edge of TSCLK	RW
2	TSFRM_H	Choose the polarity of TSFRM. 0: TSFRM is active low 1: TSFRM is active high	RW
1	TSSTR_H	Choose the polarity of TSSTR. 0: TSSTR is active low 1: TSSTR is active high	RW
0	TSFAIL_H	Choose the polarity of TSFAIL. 0: TSFAIL is active low 1: TSFAIL is active high	RW

NOTE:

*1: END_BT and END_WD in register TSCFG.

Byte0 Bit0-7	Byte1 Bit0-7	Byte2 Bit0-7	Byte3 Bit0-7	Byte4 Bit0-7	Byte5 Bit0-7	Byte6 Bit0-7	Byte7 Bit0-7

END_WD=0

Byte0	Byte1	Byte2	Byte3
Byte4	Byte5	Byte6	Byte7

END_WD=1

Byte3	Byte2	Byte1	Byte0
Byte7	Byte6	Byte5	Byte4

END_BT=0	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7
END_BT=1	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

10.3.3 TSSI Control Register (TSCTRL)

The register TSCTRL is used to control TSSI to work.

TSCTRL 0x134E0008

Bit	7	6	5	4	3	2	1	0
	Reserved					F_DTRM	F_OVRNM	F_TRIGM
RST	0	0	0	0	0	0	1	1

Bits	Name	Description	RW
7:3	Reserved	Writing has no effect, read as zero.	R
2	F_DTRM	FIFO data trigger interrupt mask. 0: enabled 1: masked	RW
1	F_OVRNM	FIFO overrun interrupt mask. 0: enabled 1: masked	RW
0	F_TRIGM	FIFO trigger interrupt mask. 0: enabled 1: masked	RW

10.3.4 TSSI State Register (TSSTAT)

The register TSSTAT is used to keep the state of TSSI.

TSSTAT 0x134E000C

Bit	7	6	5	4	3	2	1	0
	Reserved					F_DTR	F_OVRN	F_TRIG
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:3	Reserved	Writing has no effect, read as zero.	R
2	F_DTR	FIFO data trigger interrupt flag. 1: active 0: not active	RW
1	F_OVRN	FIFO overrun interrupt flag. 1: active 0: not active	RW

0	F_TRIG	FIFO trigger interrupt flag. 1: active 0: not active	RW

10.3.5 TSSI FIFO Register (TSFIFO)

The register TSFIFO is corresponded to TSSI FIFO.

TSFIFO **0x134E0010**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA																															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

10.3.6 TSSI PID Enable Register (TSPEN)

The register TSPEN is used to control the PID filtering.

TSPEN **0x134E0014**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EN_151	EN_141	EN_131	EN_121	EN_111	EN_101	EN_91	EN_81	EN_71	EN_61	EN_51	EN_41	EN_31	EN_21	EN_11	EN_01	EN_150	EN_140	EN_130	EN_120	EN_110	EN_100	EN_90	EN_80	EN_70	EN_60	EN_50	EN_40	EN_30	EN_20	EN_10	EN_00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:16	EN_x1 (x=15~0)	PID filter enable for TSPIDx.PID1. 0: not enable 1: enable	RW
15:0	EN_x0 (x=15~0)	PID filter enable for TSPIDx.PID0. 0: not enable 1: enable	RW

10.3.7 TSSI Data Number Register (TSNUM)

The register TSNUM is used to show the current number of the data in FIFO.

TSNUM **0x134E0018**

Bit	7	6	5	4	3	2	1	0
	Reserved	DNUM						
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7	Reserved	Writing has no effect, read as zero.	R
6:0	DNUM	The current number of data in FIFO. The range of the data number is from 0 to 127.	RW

10.3.8 TSSI Data Trigger Register (TSDTR)

The register TSDTR is used to trigger the FIFO level interrupt when the current data number in the FIFO is more than the value in TSDTR.

TSDTR **0x134E001C**

Bit	7	6	5	4	3	2	1	0
	Reserved	DTR						
RST	0	1	1	1	1	1	1	1

Bits	Name	Description	RW
7	Reserved	Writing has no effect, read as zero.	R
6:0	DTRG	The trigger number of FIFO. The range of the data number is from 0 to 127.	RW

10.3.9 TSSI PID Filter Registers (TSPID0~15)

The registers TSPID0~15 are used to store PID values that need to be filtered from MPEG-2 TS.

TSPID0~15 **0x134E0020~0x134E005C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved			PID1													Reserved			PID0												
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:29	Reserved	Writing has no effect, read as zero.	R
28:16	PID1	Set the PID value that needs to be filtered.	RW
15:13	Reserved	Writing has no effect, read as zero.	R
12:0	PID0	Set the PID value that needs to be filtered.	RW

10.3.10 TSSI DMA Descriptor Address (TSDDA)

TDDA 0x134E0060

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DA																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
[31:0]	DA	Descriptor Address. It must be aligned to 4-word. It can be written by software and coped from the descriptor. Software should write the descriptor address before enable the DMA, and only the first DA needed if LINK enabled.	RW

10.3.11 TSSI DMA Target Address (TSDTA)

TDTA 0x134E0064

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TA																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R
31:0	TA	Target Address. It should be aligned to 32-word. It will be coped from the descriptor.	R

10.3.12 TSSI DMA Identifier (TSDID)

TDID 0x134E0068

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																DID															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R
31:16	Reserved	Writing has no effect, read as zero.	R
15:0	DID	Descriptor Identifier. It will be coped from the descriptor. It will be copied to TDST.DID.	R

10.3.13 TSSI DMA Command (TSDCMD)

TDCMD 0x134E006C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TLEN																								Reserved			TEFE	TSZ		TEIE	LINK
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R
31:8	TLEN	Transfer Length. Unit is word. When DMA transfer end, the TLEN will be counted down to 0. This field will be coped from the descriptor.	R
7:5	Reserved	Writing has no effect, read as zero.	R
4	TEFE	Transfer End Flag Enable. 0: disable TSDST.TEND 1: enable TSDST.TEND This field will be coped from the descriptor.	R
3:2	TSZ	Expected Transfer Size. 0: 4 word 1: 8 word 2: 16 word 3: 32 word This field will be coped from the descriptor.	R
1	TEIE	Transfer End Interrupt Enable.	R

		0, disable interrupt 1, enable interrupt This field will be coped from the descriptor.	
0	LINK	Descriptor Link Enable. 0, disable 1, enable This field will be coped from the descriptor.	R

10.3.14 TSST DMA Status (TSDST)

TDST 0x134E0070

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DID																Reserved															TEND
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:16	DID	Copied from Descriptor Identifier when DMA interrupt occurred. Write 0 will clear this field.	RW
15:1	Reserved	Write has no effect, read as zero.	R
0	TEND	Transfer End Flag. 0: Transfer is not finished 1: Transfer is finished Write 0 will clear this field.	RW

10.3.15 TSSI Transfer Control Register (TSTC)

TSTC 0x134E0074

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																										OP		Reserved	OPE	EMM	APM
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:2	Reserved	Writing has no effect, read as zero.	R

<table>
<tr><td>5:4</td><td>OP</td><td>Optional Priority Configuration. Only used when OPE is set to 1.

<table><tr><th>OP</th><th>TSSI AHB Priority</th><th>Number of Data in FIFO</th></tr><tr><td>2’b00</td><td>0
1
2
3</td><td>n <= 16
16 < n <= 32
32 < n <= 64
64 < n</td></tr><tr><td>2’b01</td><td>0
1
2
3</td><td>n <= 16
16 < n <= 48
48 < n <= 80
80 < n</td></tr><tr><td>2’b10</td><td>0
1
2
3</td><td>n <= 32
32 < n <= 64
64 < n <= 96
96 < n</td></tr><tr><td>2’b11</td><td>0
1
2
3</td><td>n <= 48
48 < n <= 80
80 < n <= 112
112 < n</td></tr></table>
It is suggested to use 2’b10.</td><td></td></tr>
<tr><td>3</td><td>Reserved</td><td>Writing has no effect, read as zero.</td><td>R</td></tr>
<tr><td>2</td><td>OPE</td><td>Optional Priority Mode Enable Control. Only used when APM is 1.
0: TSSI calculates the priority according to the fifo status
1: TSSI calculates the priority according to OP which is configured by software</td><td>RW</td></tr>
<tr><td>1</td><td>EME</td><td>Emergency Mode Enable Control.
0: Emergency Mode Disable
1: Emergency Mode Enable</td><td>RW</td></tr>
<tr><td>0</td><td>APM</td><td>Auto Priority Mode Enable Control.
0: Auto priority mode disables. TSSI uses the priority set by arbiter
1: Auto priority mode enables. TSSI can use the priority according the FIFO status</td><td>RW</td></tr>
</table>

10.4 TSSI Timing

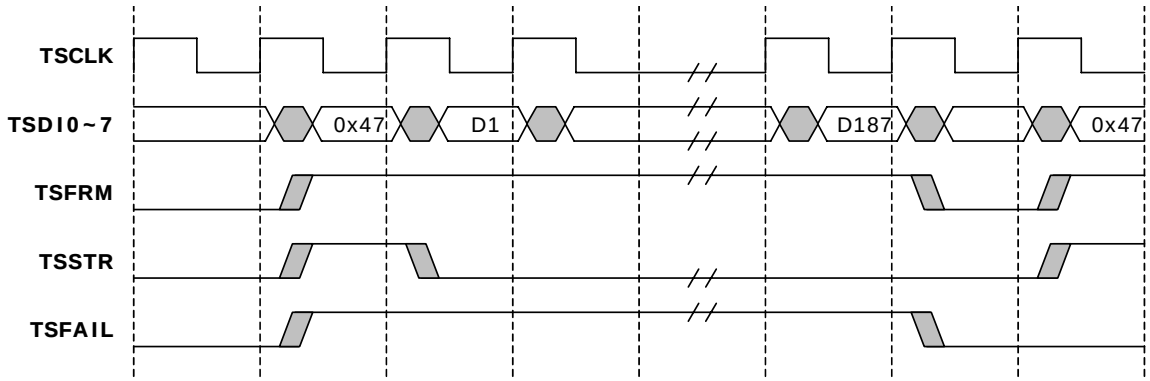


Figure 10-1 Timing waveform in parallel mode

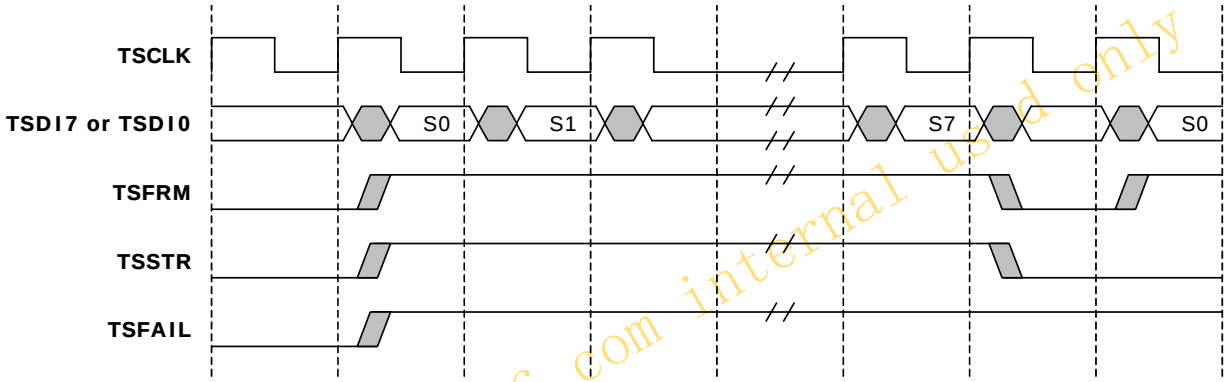


Figure 10-2 Timing waveform in serial mode

10.5 TSSI Guide

10.5.1 TSSI Operation without PID Filtering Function

1 Set TSCTRL to 0x03 to mask all interrupts.
2 Set TSCFG to choose the working mode of TSSI and the trigger value of FIFO.
3 Set TSENA.PID_EN to 0 to turn off the PID filtering function.
4 set the DMA descriptor first if using the master DMA.
5 Set TSENA.DMA_EN to 1 or 0 to decide whether to use the DMA mode or not.
6 Write 0x00 to TSSTAT clear all interrupt flag.
7 Set TSCTRL to 0x00 to enable all interrupts.
8 Set TSENA.ENA to 1 to turn on TSSI module.

10.5.2 TSSI Operation with PID Filtering Function

1 Set TSCTRL to 0x03 to mask all interrupts.
2 Set TSCFG to choose the working mode of TSSI and the trigger value of FIFO.
3 Set TSENA.PID_EN to 1 to turn on the PID filtering function.
4 set the DMA descriptor first if using the master DMA.
5 Set TSENA.DMA_EN to 1 or 0 to decide whether to use the DMA mode or not.
6 Write 0x00 to TSSTAT clear all interrupt flag.
7 Set TSCTRL to 0x00 to enable all interrupts.
8 Set TSENA.ENA to 1 to turn on TSSI module.
9 Change TSPID registers and then set TSPID to enable the PID filter.
10 When PID in TS package is equal to the valve in TSPID register, the TS package will be getting.

11 Ethernet MAC Controller

11.1 Overview

The processor contains one Ethernet media access controller (MAC), each capable of supporting 10/100Mbps Ethernet. The MAC provides the interface between the host application and the PHY layer through the media independent interface (MII) or the Reduced-MII (RMII). The PHY layer device is external to the processor.

The MAC supports the protocol requirements to meet the Ethernet/IEEE 802.3 specification. The MAC operates in both half and full duplex modes. In half-duplex mode, the controller supports the IEEE802.3 Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol. In full-duplex mode, it supports the IEEE802.3 MAC Control Layer, including the pause operation for flow control.

A dedicated DMA engine is implemented to support the MAC so that the general purpose DMA is not required.

The MAC features are:

- IEEE 802.3, 802.3u specification compliance
- 10/100 Mbps data transfer rates
- IEEE 802.3 compliant MII interface to talk to an external PHY
- The Reduced-MII interface to an external PHY
- Full and half duplex
- CSMA/CD in half duplex
- Flow control support for full duplex
- Collision detection and auto retransmit on collisions in half duplex
- Automatic 32-bit CRC generation and checking
- Optional to insert PAD/CRC32 on transmit packets
- Optional automatic Pad stripping on the receive packets
- External and internal loopback support on the MII
- Filtering modes supported on the Ethernet side
  - One 48 bit Perfect address
  - 64 hash-filtered multicast addresses
  - Pass all multicast addresses
  - Promiscuous Mode
  - Pass all incoming packets with a status report
  - Toss bad packets
- Dedicated DMA engine using burst mode
- DMA linked list Descriptor Chaining support
- Descriptor architecture allows large blocks of data transfer with minimum CPU intervention

11.2 VLAN support Ethernet Signals

The following table shows the signals associated with the Ethernet MAC MII interfaces.

Table 11-1 Ethernet MII Signals

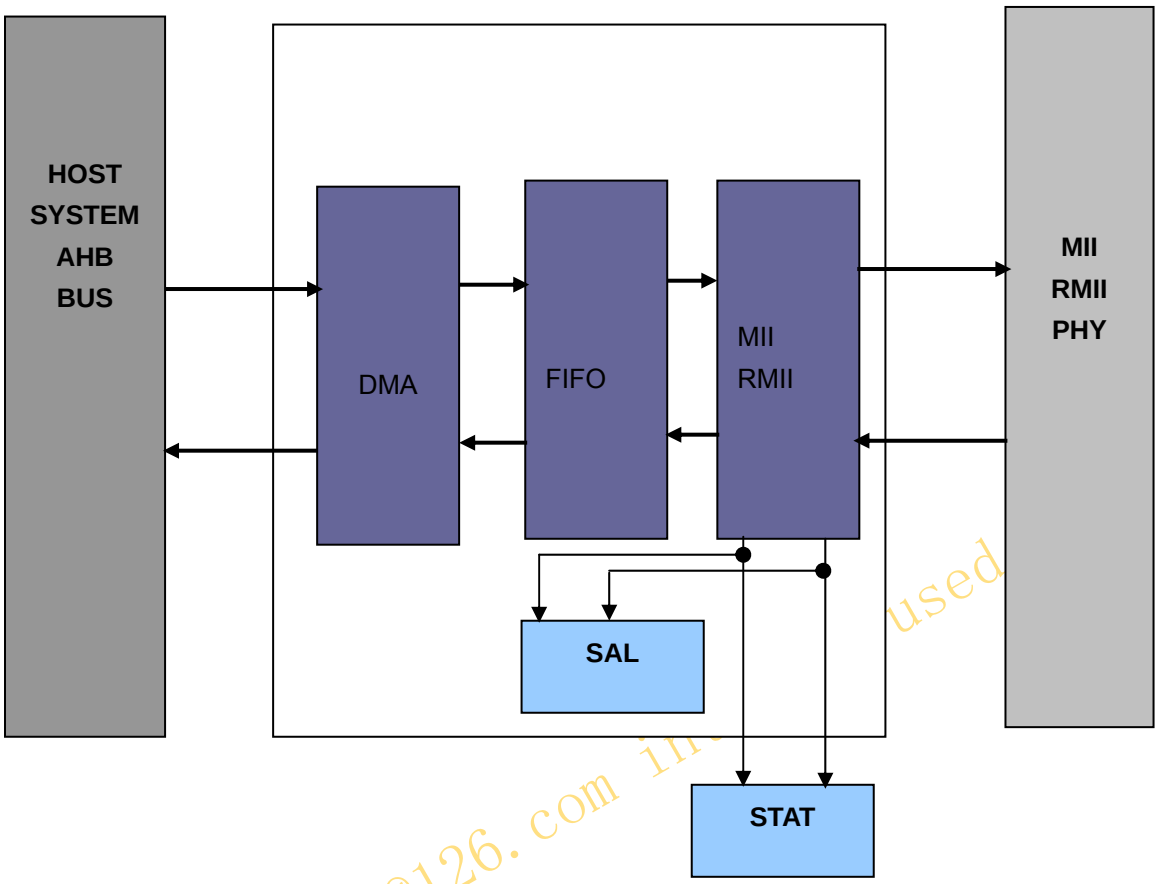
Signal	I/O	Description
RX_CLK	Input	Continuous clock that provides the timing reference for the datatransfer from the PHY to the MAC. RX_CLK is sourced by the PHY.RX_CLK must have a frequency equal to 25% of the data rate of the received signal data stream (typically 25 MHz at 100-Mbps and 2.5MHz at 10-Mbps).
RX_DV	Input	RX_DV is driven by the external Ethernet PHY and indicates that a receive frame is in process and that the data on RXD[3:0] is valid.
RX_ER	Input	RX_ER is driven by the Ethernet PHY. RX_ER shall be asserted for one or more RX_CLK periods to indicate to the MAC that an error was detected some where in the frame presently being transferred from the PHY to the MAC.
RXD [3:0]	Input	RXD[3:0] is a 4-bit wide data bus driven by the PHY to the MAC synchronous with RX_CLK. For each RX_CLK period in which RX_DV is asserted, RXD[3:0] transfers four bits of recovered data from the PHY to the MAC. While RX_DV is negated, RXD[3:0] has no effect on the MAC.
TX_CLK(REF_CLK)	Input	Continuous clock input for synchronization of transmit data. 25 MHz when operating at 100-Mbps and 2.5 MHz when operating at 10-Mbps. When in RMII mode, REF_CLK 50MZ whenever at 100-Mbps or at 10-Mbps.
TX_EN	Output	Indicates that the data on TXD[3:0] is valid.
TXD [3:0]	Output	4-bit wide data bus synchronous to TX_CLK. For each TX_CLK period in which TX_EN is asserted, TXD[3:0] presents valid data to the PHY. While TX_EN is negated the data presented on TXD[3:0] is not valid.
CRS	Input	The PHY asserts CRS when either transmit or receive medium is non idle. The PHY negates CRS when both the transmit and receive medium are idle. CRS is an asynchronous input.
COL	Input	The PHY asserts COL upon detection of a collision on the medium and continues to assert COL while the collision condition persists. COL is an asynchronous input. The COL signal is ignored by the MAC when operating in full duplex mode.
MDC	Output	MDC is sourced by the MAC to the PHY as the timing reference for transfer of information on the MDIO signal. MDC is an aperiodic signal that has no maximum high or low times. The MDC frequency is

		fixed at system bus clock divided by 160.
MDIO	I/O	MDIO is the bidirectional data signal between the MAC and the PHY that is clocked by MDC.

Table 11-2 Pin Map of MII and RMII Mode

Normal MII Mode	Reduced MII Mode
TXD[1:0]	TXD[1:0]
TXD[3:2]	NC
TXEN	TXEN
TXCLK	REF_CLK
RXD[1:0]	RXD[1:0]
RXD[3:2]	NC
RXDV	CRS DV
RXCLK	NC
COL	NC
CRS	NC
MDC	MDC
MDIO	MDIO

11.3 Block Diagram



11.4 DMA Module

11.4.1 Overview

The DMA module provides a DMA bridge between a host system that uses an AMBA AHB™ bus and the Ethernet MAC.

The DMA module interfaces to the host system through 32-bit AHB Master and Slave ports. On the MAC side of the module, it has a high-performance synchronous interface for DMA data transfer to/from the FIFO and a semi-synchronous interface for configuring the MAC and the FIFO through their HST interfaces.

For ease of handling by software, transfers are handled using linked lists of transfer descriptors which together define one buffer in host memory for Tx operations and another for Rx operations. These buffers will typically be configured as ring buffers but this is up to the user to implement.

Registers within the DMA provide Control and Status information concerning these transfers. These registers are accessed through the AHB Slave port, alongside accesses to the HST interfaces on the MAC and the FIFO.

11.4.2 Features

- AMBA AHB 2.0 compliant Master port for payload data transfer
- AMBA AHB 2.0 compliant Slave port for DMA bridge, FIFO and MAC configuration
- RAM-free synchronous single-clock domain design
- 32-bit architecture
- Linked-List Transfer Descriptors for minimal software overhead

11.4.3 Register Map

Name	RW	Reset Value	Address	Access Size
DMATxCtrl	RW	0x00000000	0x134B0180	32
DMATxDesc	RW	0x?0000000	0x134B0184	32
DMATxStatus	RW	0x????????	0x134B0188	32
DMARxCtrl	RW	0x00000000	0x134B018C	32
DMARxDesc	RW	0x?0000000	0x134B0190	32
DMARxStatus	RW	0x????????	0x134B0194	32
DMAIntrMask	RW	0x?0000000	0x134B0198	32
DMAInterrupt	RW	0x?0000000	0x134B019C	32

11.4.4 Register Description

11.4.4.1 DMATxCtrl

DMATxCtrl **0x134B0180**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																															TXEN
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:1	Reserved	Writing has no effect, read as zero.	R
0	TXEN	Setting this bit enables DMA transmit packet transfers. The bit is cleared by the built-in DMA controller whenever it encounters a Tx Underrun or Bus Error state. Default is '0'.	RW

11.4.4.2 DMATxDesc

DMATxDesc **0x134B0184**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DMATxDesc																															Reserved
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:1	DMATxDesc	When TxEnable is set by the host, the built-in DMA controller reads this register to discover the location in host memory of the first transmit packet descriptor.	R
0	Reserved	Writing has no effect, read as zero.	R

11.4.4.3 DMATxStatus

DMATxStatus **0x134B0188**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								TxPktCount								Reserved														TxUdrun	TxPktSent
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:24	Reserved	Writing has no effect, read as zero.	R
23:16	TxPktCount	8-bit transmit packet counter that is incremented whenever the built-in DMA controller successfully transfers a packet, and decremented whenever the host writes a '1' to bit 0 of this register.	RW
15:2	Reserved	Writing has no effect, read as zero.	R
1	TxUdrun	Set whenever the DMA controller reads a set ('1') Empty Flag in the descriptor it is processing.	RW
0	TxPktSent	When set, indicates that one or more packets have been successfully transferred. Writing a '1' to this bit reduces the TxPktCount value by one. The bit is cleared whenever TxPktCount is zero.	RW

11.4.4.4 DMARxCtrl

DMARxCtrl 0x134B018C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																															RXEN
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:1	Reserved	Writing has no effect, read as zero.	R
0	RXEN	Setting this bit enables DMA receive packet transfers. When set, the built-in DMA controller will start to receive a new packet whenever the FIFO indicates that a new packet is available (FRSOF asserted). The bit is cleared by the built-in DMA controller whenever it encounters an Rx Overflow or Bus Error state.	RW

11.4.4.5 DMARxDescriptor

DMARxDescriptor 0x134B0190

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DESCP_ADDR																															0
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:2	DESCP_	When RxEnable is set by the host, the built-in DMA controller reads this	RW

	ADDR	register to discover the location in host memory of the first receive packet Descriptor.	
1:0	0	Ignored by the DMA controller, since it is a requirement of the system that all descriptors are 32-bit aligned in host memory.	RW

11.4.4.6 DMARxStatus

DMARxStatus **0x134B0194**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								RxPKT_CNT								Reserved												BUS_ERR	Rx_OF	0	RxPKT
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:24	Reserved	Writing has no effect, read as zero.	R
23:16	RxPKT_CNT	8-bit receive packet counter that is incremented whenever the built-in DMA controller successfully transfers a packet, and is decremented whenever the host writes a '1' to bit zero of this register.	RWC
15:4	Reserved	Writing has no effect, read as zero.	R
3	BUS_ERR	When set, indicates that a host slave split, retry or error response was received by the DMA controller.	RWC
2	Rx_OF	Set whenever the DMA controller reads a zero Empty Flag in the descriptor it is processing.	RWC
1	0	0.	R
0	RxPKT	When set, indicates that one or more packets have been successfully transferred. Writing a '1' to this bit reduces the RxPktCount value by one. The bit is cleared whenever RxPktCount is zero.	RW

11.4.4.7 DMAIntrMask

DMAIntrMask **0x134B0198**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	STEN	CLRCNT	AUTOZ	BURST	Reserved																				RX_BUS_ERR	Rx_OF	0	RxPKT	TX_BUS_ERR	0	Tx_UR	TxPKT
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	STEN	Enable counting STAT.	RW
30	CLRCNT	Zero all counters and carries immediately.	RW
27	AUTOZ	Auto zero addressed.	RW
28:27	BURST	00: BURST4 01: BURST4 10: BURST8 11: BURST16	RW
26:8	Reserved	Writing has no effect, read as zero.	R
7	RX_BUS_ERR	Setting this bit to '1' enables the BusError bit in the DMARxStatus register as an interrupt source.	RW
6	Rx_OF	Setting this bit to '1' enables the RxOverflow bit in the DMARxStatus register as an interrupt source.	RW
5	0	0.	R
4	RxPKT	Setting this bit to '1' enables the RxPktReceived bit in the DMARxStatus register as an interrupt source.	RW
3	TX_BUS_ERR	Setting this bit to '1' enables the BusError bit in the DMATxStatus register as an interrupt source.	RW
2	0	0.	R
1	Tx_UR	Setting this bit to '1' enables the TxUnderrun bit in the DMATxStatus register as an interrupt source.	RW
0	TxPKT	Setting this bit to '1' enables the TxPktSent bit in the DMATxStatus register as an interrupt source.	RW

11.4.4.8 DMAInterrupt

DMAIntrrupt 0x134B019c

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																								RX_BUS_ERR	Rx_OF	0	RxPKT	TX_BUS_ERR	0	Tx_UR	TxPKT
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:8	Reserved	Writing has no effect, read as zero.	R
7	RX_BUS_ERR	Set to '1' to record a Receive Bus Error interrupt when the BusError bit in the DMARxStatus register and bit 7 of the DMAIntrMask register are both set.	R
6	Rx_OF	Set to '1' to record an Rx Overflow interrupt when the RxOverflow bit	R

		in the DMARxStatus register and bit 6 of the DMAIntrMask register are both set.	
5	0	0.	R
4	RxPKT	Set to '1' to record a RxPkt Received interrupt when the RxPktReceived bit in the DMARxStatus register and bit 4 of the DMAIntrMask register are both set.	R
3	TX_BUS_ERR	Set to '1' to record a Transmit Bus Error interrupt when the BusError bit in the DMATxStatus register and bit 3 of the DMAIntrMask register are both set.	R
2	0	0.	R
1	Tx_UR	Set to '1' to record a Tx Underrun interrupt when the TxUnderrun bit in the DMATxStatus register and bit 1 of the DMAIntrMask register are both set.	R
0	TxPKT	Set to '1' to record a Tx Pkt Sent interrupt when the TxPktSent bit in the DMATxStatus register and bit 0 of the DMAIntrMask register are both set.	R

11.5 FIFO Module

11.5.1 Overview

The FIFO module consists of the following seven major RTL modules:

- 10/100 Mb/s FIFO Fabric Transmit Interface Module
- 10/100 Mb/s FIFO System Transmit Interface Module
- 10/100 Mb/s FIFO Fabric Receive Interface Module
- 10/100 Mb/s FIFO System Receive Interface Module
- 10/100 Mb/s FIFO System Watermark Module
- 10/100 Mb/s FIFO Host Interface Module
- 10/100 Mb/s FIFO Clock and Reset Module

11.5.2 Features

- Data queuing for increased system-level throughput
- Synthesis-definable buffer sizes from 64K bytes to 256 bytes
- Clock frequency-independent I/O ports
- Single- or multiple-word I/O data transfers
- Programmable high and low receive storage-level indicators
- CPU frame insertion and inspection capabilities
- Automatic pause frame handshaking circuitry
- Programmable pause frame handshaking reassertion interval
- Programmable high transmit storage-level indicator
- Graceful receive memory full frame drop
- Graceful enable and disable
- Programmable frame mode or word cut-through threshold
- Transmit storage underrun indication
- Transmit storage frame rewind capabilities
- Full memory utilization
- Optional per transmit frame MAC configuration data
- Synchronous dual-port memory utilization
- Low gate count
- Easy synthesis

11.5.3 Register Map

Name	RW	Reset Value	Address	Access Size
CFG_R0	RW	0x0000001F	0x134B003C	32
CFG_R1	rw	0x0FFFFFFF	0x134b004C	32
CFG_R2	RW	0x1FFF1FFF	0x134B0050	32
CFG_R3	RW	0x0FFF0FFF	0x134B0054	32
CFG_R4	RW	0x00000000	0x134B0058	32

CFG_R5	RW	0x0000FFFF	0x134B005C	32
RAM_ACC_R0	RW	0x00000000	0x134B0060	32
RAM_ACC_R1	RW	0x00000000	0x134B0064	32
RAM_ACC_R2	RW	0x00000000	0x134B0068	32
RAM_ACC_R3	R	0x00000000	0x134B006C	32
RAM_ACC_R4	RW	0x00000000	0x134B0070	32
RAM_ACC_R5	RW	0x00000000	0x134B0074	32
RAM_ACC_R6	RW	0x00000000	0x134B0078	32
RAM_ACC_R7	R	0x00000000	0x134B007C	32

11.5.4 Register Description

11.5.4.1 CFG_R0

CFG_R 0x134B003C

Bit	31:21	20	19	18	17	16	15:13	12	11	10	9	8	7:6	5	4	3	2	1	0
	Reserved	FTFENRPLY	STFENRPLY	FRFENRPLY	SRFENRPLY	WTMENRPLY	Reserved	FTFENREQ	STFENREQ	FRFENREQ	SRFENREQ	WTMENREQ	Reserved		HSTRSTFT	HSTRSTST	HSTRSTFR	HSTRSTSR	HSTRSTWT
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Bits	Name	Description	RW
31:21	Reserved	Writing has no effect, read as zero.	R
20	FTFENRPLY	When asserted, the mmiitfif_fab module is enabled. When negated, the mmiitfif_fab module is disabled. The bit should be polled until it reaches the expected value.	R
19	STFENRPLY	When asserted, the mmiitfif_sys module is enabled. When negated, the mmiitfif_sys module is disabled. The bit should be polled until it reaches the expected value.	R
18	FRFENRPLY	When asserted, the mmiirfif_fab module is enabled. When negated, the mmiirfif_fab module is disabled. The bit should be polled until it reaches the expected value.	R
17	SRFENRPLY	When asserted, the mmiirfif_sys module is enabled. When negated, the mmiirfif_sys module is disabled. The bit should be polled until it reaches the expected value.	R
16	WTMENRPLY	When asserted, the mmiitfif_wtm module is enabled. When negated, the mmiitfif_wtm module is disabled. The bit should be polled until it reaches the expected value.	R
15:13	Reserved	Writing has no effect, read as zero.	R
12	FTFENREQ	When asserted, requests enabling of the mmiitfif_fab module. When	RW

		negated, requests disabling of the mmiitfif_fab module.	
11	STFENREQ	When asserted, requests enabling of the mmiitfif_sys module. When negated, requests disabling of the mmiitfif_sys module.	RW
10	FRFENREQ	When asserted, requests enabling of the mmiirfif_fab module. When negated, requests disabling of the mmiirfif_fab module.	RW
9	SRFENREQ	When asserted, requests enabling of the mmiirfif_sys module. When negated, requests disabling of the mmiirfif_sys module.	RW
8	WTMENREQ	When asserted, requests enabling of the mmiitfif_wtm module. When negated, requests disabling of the mmiitfif_wtm module.	RW
7:5	Reserved	Writing has no effect, read as zero.	R
4	HSTRSTFT	When asserted, this bit will place the mmiitfif_fab module in reset.	RW
3	HSTRSTST	When asserted, this bit will place the mmiitfif_sys module in reset.	RW
2	HSTRSTFR	When asserted, this bit will place the mmiirfif_fab module in reset.	RW
1	HSTRSTSR	When asserted, this bit will place the mmiirfif_sys module in reset.	RW
0	HSTRSTWT	When asserted, this bit will place the mmiitfif_wtm module in reset.	RW

11.5.4.2 CFG_R1

CFG_R1 0x134B004C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				CFGFRTH												CFGXOFFRTX																
RST	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	RW
31:28	Reserved	Writing has no effect, read as zero.	R
27:16	CFGFRTH	This hex value represents the minimum number of 4-byte locations that will be simultaneously stored in the receive RAM, relative to the beginning of the frame being input, before **frrdy** may be asserted. Note that **frrdy** will be latent a certain amount of time due to fabric transmit clock to system transmit clock time domain crossing, and conditional on **fracpt** assertion. When set to maximum value, **frrdy** may be asserted only after the completion of the input frame. The value of this register must be greater than 18d when **hstdrplt64** is asserted; the additional two bits allow for overhead. The register length shown is for a receive RAM with 12 address bits (16K Bytes). The register length will vary with the addressable RAM size used.	RW
15:0	CFGXOFFR	This hex value represents the number of pause quanta (64 bit times)	RW

	TX	that occur after an XOFF pause frame has been acknowledged (**psack**), while the M-MIIFIF receive storage level has remained higher than the low watermark, until the M-MIIFIF will reassert **tpcf**.	

11.5.4.3 CFG_R2

CFG_R2 **0x134B0050**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved			CFGHWM													Reserved			CFGLWM													
RST	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	RW
31:29	Reserved	Writing has no effect, read as zero.	R
28:16	CFGHWM	This hex value represents the maximum number of 4-byte words that will be simultaneously stored in the receive RAM before **tpcf** and **psval** will facilitate an XOFF pause control frame. The register length shown is for a receive RAM with 12 address bits (16K Bytes). The register length will vary with the addressable RAM size used.	RW
15:13	Reserved	Writing has no effect, read as zero.	R
12:0	CFGLWM	This hex value represents the minimum number of 4-byte words that will be simultaneously stored in the receive RAM before **tpcf** and **psval** will facilitate an XON pause control frame in response to a previously transmitted XOFF pause control frame. The register length shown is for a receive RAM with 12 address bits (16K Bytes). The register length will vary with the addressable RAM size used.	RW

11.5.4.4 CFG_R3

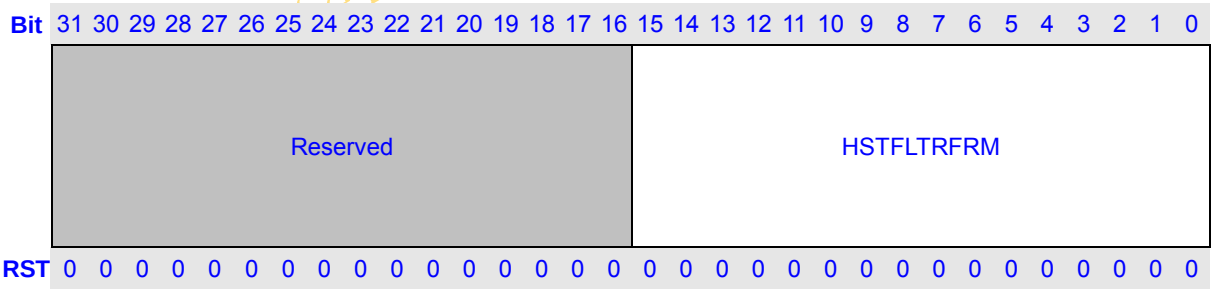
CFG_R3 **0x134B0054**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				CFGHWMFT												Reserved				CFGFTTH											
RST	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	RW
31:28	Reserved	Writing has no effect, read as zero.	R
27:16	CFGHWMFT	This hex value represents the maximum number of 4-byte locations that will be simultaneously stored in the transmit RAM before **fthwm** will be asserted. Note that **fthwm** has two **ftclk** clock periods of latency before assertion or negation. This should be considered when calculating any headroom required for maximum size packets. The register length shown is for a transmit RAM with 11 address bits (8K Bytes). The register length will vary with the addressable RAM size used.	RW
15:12	Reserved	Writing has no effect, read as zero.	R
11:0	CFGFTTH	This hex value represents the minimum number of 4-byte locations that will be simultaneously stored in the transmit RAM, relative to the beginning of the frame being input, before **tpsf** will be asserted. Note that **tpsf** will be latent a certain amount of time due to fabric transmit clock to system transmit clock time domain crossing. When set to maximum value, **tpsf** will be asserted only after the completion of the input frame. The register length shown is for a transmit RAM with 11 address bits (8K Bytes). The register length will vary with the addressable RAM size used.	RW

11.5.4.5 CFG_R4

CFG_R4 0x134B0058



Bits	Name	Description	RW
31:16	Reserved	Writing has no effect, read as zero.	R
15:0	HSTFLTRFRM	These configuration bits are used to signal the drop frame conditions internal to the M-MIIFIF. The bits correspond to the Receive Statistics Vector on a one-to-one basis. For example bit 0 corresponds to RSV[16], and bit 1 corresponds to RSV[17]. When these bits compare and the don't care is not asserted, the frame will be dropped. This is true for all **hstfltrm** bits with the exception of **hstfltrfrm[15]**, which is used to compare with the unicast address match input port of the M-MIIFIF.	RW

		The setting of these bits, along with their don't care values in the **hstfltrfrmdc** configuration registers, create the filter that will assert the **srdrpfrm** output if the receive frame does not pass the acceptable conditions and should be dropped by the system. For example, if you want to drop a frame that contains a FCS Error, bit 4 would be set, and all receive frames that have RSV[20] asserted would be dropped.	

11.5.4.6 CFG_R5

CFG_R5 **0x134B005C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved									CFGHDPLX	SRFULL	HSTSRFULLCLR	CLKENMODE	HSTDRPLT64	Reserved		HSTFLTRFRMDC															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	RW
31:23	Reserved	Writing has no effect, read as zero.	R
22	CFGHDPLX	Assertion of this bit configures the M-MIIFIF to enable half-duplex backpressure as a flow control mechanism. Deassertion of this bit configures the M-MIIFIF to enable pause frames as a flow control mechanism.	RW
21	SRFULL	Assertion of this read-only bit indicates that the maximum capacity of the receive FIFO storage has been met or exceeded. If the **cfgfrth** bits are not all set, exceeding the FIFO storage capacity may result in data frame truncation as opposed to full frame deletion. Frame truncation is not recommended as an operation of the M-MIIFIF. The bit should be polled until it reaches the expected value.	RW
20	HSTSRFULLCLR	This bit should be written asserted when it is desired to clear the **srfull** indicator bit. After **hstfullclr** assertion, **srfull** should be read until it becomes deasserted. **hstfullclr** should then be written deasserted for the indicator to become operational again.	RW
19	CLKENMODE	This bit should be asserted when TXCENI is being toggled. This is done when the PE-MACMII is integrated with the PE-RMII.	RW
18	HSTDRPLT64	Setting this bit will cause the frame to be dropped if a receive frame is less than 64 bytes in length. This bit should not be asserted if **cfgfrth** is less than 12h.	RW

17:16	Reserved	Writing has no effect, read as zero.	R
15:0	HSTFLTRFRMDC	These configuration bits indicate which Receive Statistics Vectors are don't cares for M-MIIFIF frame drop circuitry. The bits correspond to the Receive Statistics Vector on a one per one basis. For example, bit 0 corresponds to RSV[16], and bit 1 corresponds to RSV[17]. Setting of a **hstfiltrfrmdc** bit will indicate a don't care for that RSV bit. Clearing the bit will look for a matching level on the corresponding **hstfiltrfrm** bit. If a match is made, then the frame is dropped. All bits should be set when **cfgfrth** bits are not all set.	RW

11.5.4.7 RAM_ACC_R0

RAM_ACC_R0 **0x134B0060**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	HSTTRAMWREQ	HSTTRAMWACK	Reserved						HSTTRAMWDAT								Reserved			HSTTRAMWADX												
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	HSTTRAMWREQ	Host transmit RAM write request. Requests the handshake of **hsttramwdat** and **hsttramwadx** values to the transmit FIFO RAM. Should only be asserted while **hsttramwack** is negated and while the transmit data path is disabled from receiving data and in a steady state. It should only be negated after receiving an asserted **hsttramwack**.	RW
30	HSTTRAMWACK	Host transmit RAM write acknowledge. Signifies the acceptance of **hsttramwdat** and **hsttramwadx** values to the transmit FIFO RAM or mmiitfif_fab module. Will only be asserted or negated following assertion or negation of **hsttramwreq.**	R
29:24	Reserved	Writing has no effect, read as zero.	R
23:16	HSTTRAMWDAT	Host transmit RAM write data. This is the upper byte of transmit FIFO RAM data that will be written at the address of **hsttramwadx[10:0]** if **hsttramwadx[12]** is negated and **hsttramwreq** is asserted. This part of the transmit FIFO RAM contains control information for the frame as follows: hsttramwdat[39] = ftcfrm hsttramwdat[38:37] = ftpppadmode[1:0]	RW

		hsttramwdat[36] = ftppen hsttramwdat[35] = ftppgenfcs hsttramwdat[34] = fteof hsttramwdat[33:32] = ftdatnvld[1:0]	
15:13	Reserved	Writing has no effect, read as zero.	R
12:0	HSTTRAMWADX	Host transmit RAM write address. This field has different functionality based on the value of **hsttramwadx[12]** and whether the M-MIIFIF FIFO RAM access register 0 is being written to or read from. When read from, **hsttramwadx[11:0]** field contains the actual write pointer value of the mmiitfif_fab module. When written to, the **hsttramwadx** register will be loaded. If **hsttramwadx[12]** is low, **hsttramwadx[10:0]** will be the transmit RAM address to which **hsttramwdat** is written. If **hsttramwadx[12]** is high, **hsttramwadx[11:0]** contains the pointer value that will be written to the mmiitfif_fab module. The register definition for **hsttramwadx[10:0]** is for a transmit RAM with 11 address bits (8K Bytes). The register length will vary with the addressable RAM size used. The definition of **hsttramwadx[12:11]** will shift to remain left-justified in the register.	RW

11.5.4.8 RAM_ACC_R1

RAM_ACC_R1 **0x134B0064**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	HSTTRAMWDAT																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:0	HSTTRAMWDAT	Host transmit RAM write data. This is the lower 4 bytes of transmit FIFO RAM data that will be written at the address of **hsttramwadx[10:0]** if **hsttramwadx[12]** is negated and **hsttramwreq** is asserted.	RW

11.5.4.9 RAM_ACC_R2

RAM_ACC_R2 0x134B0068

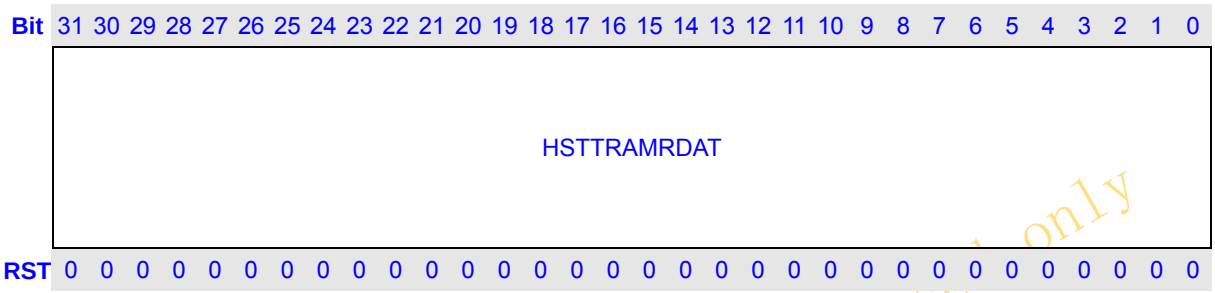
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	HSTTRAMRREQ	HSTTRAMRACK	Reserved						HSTTRAMRDAT								Reserved			HSTTRAMRADX												
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	HSTTRAMRREQ	Host transmit RAM read request. Requests the handshake of **hsttramradx** values to the transmit FIFO RAM and **hsttramrdat** values from the transmit FIFO RAM. Should only be asserted while **hsttramrack** is negated and while the transmit data path is disabled from receiving data and in a steady state. It should only be negated after receiving an asserted **hsttramrack**.	RW
30	HSTTRAMRACK	Host transmit RAM read acknowledge. Signifies the acceptance of **hsttramradx** values to the transmit FIFO RAM and reception of **hsttramrdat** from the transmit FIFO RAM location addressed. Will only be asserted or negated following assertion or negation of **hsttramrreq**.	R
29:24	Reserved	Writing has no effect, read as zero.	R
23:16	HSTTRAMRDAT	Host transmit RAM read data. This is the upper byte of transmit FIFO RAM data that was read at the address of **hsttramradx[10:0]** if **hsttramradx[12]** is negated and **hsttramrreq** is asserted. This part of the transmit FIFO RAM contains control information for the frame as follows: hsttramrdat[39] = ftcfrm hsttramrdat[38:37] = ftpppadmode[1:0] hsttramrdat[36] = ftppen hsttramrdat[35] = ftppgenfcs hsttramrdat[34] = fteof hsttramrdat[33:32] = ftdatnvld[1:0]	R
15:13	Reserved	Writing has no effect, read as zero.	R
12:0	HSTTRAMRADX	Host transmit RAM read address. If **hsttramradx[12]** is written low, **hsttramradx[10:0]** is the transmit FIFO RAM address that **hsttramrdat** is read from. If **hsttramradx[12]** is written high, **hsttramradx[11:0]** contains the pointer value read from the mmiitfif_sys module.	RW

		The register definition for **hsttramradx[10:0]** is for a transmit RAM with 11 address bits (8K Bytes). The register length will vary with the addressable RAM size used. The definition of **hsttramradx[12:11]** will shift to remain left-justified in the register.	

11.5.4.10 RAM_ACC_R3

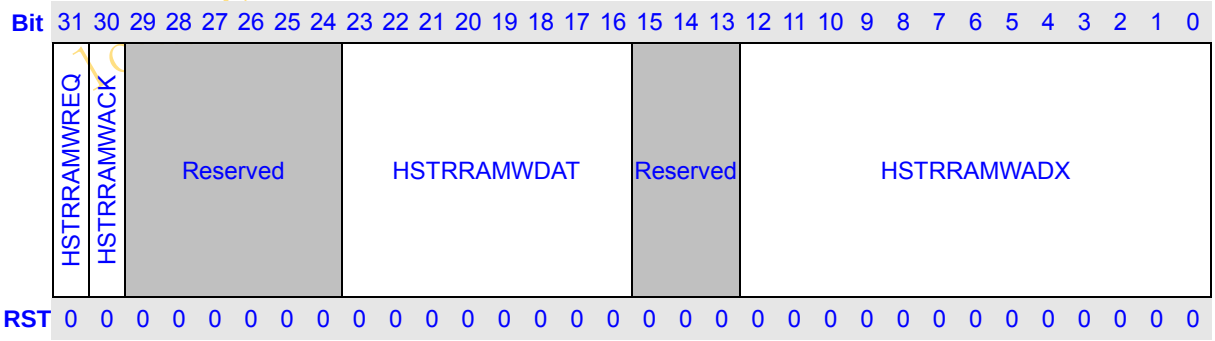
RAM_ACC_R3 **0x134B006C**



Bits	Name	Description	RW
31:0	HSTTRAMRDAT	Host transmit RAM read data. This is the lower 4 bytes of transmit FIFO RAM data that is read at the address of **hsttramradx[10:0]** if **hsttramradx[12]** is negated and **hsttramrreq** is asserted.	R

11.5.4.11 RAM_ACC_R4

RAM_ACC_R4 **0x134B0070**



Bits	Name	Description	RW
31	HSTRRAMWREQ	Host receive RAM write request. Requests the handshake of **hstrramwdat** and **hstrramwadx** values to the receive FIFO RAM. Should only be asserted while **hstrramwack** is negated and while the receive data path is disabled from receiving data and is in a steady state. It should only be negated after receiving an asserted **hstrramwack**.	RW

30	HSTRRAMWACK	Host receive RAM write acknowledge. Signifies the acceptance of **hstrramwdat** and **hstrramwadx** values to the receive FIFO RAM or mmiirfif_sys module. Will only be asserted or negated following assertion or negation of **hstrramwreq**.	R
29:24	Reserved	Writing has no effect, read as zero.	R
23:16	HSTRRAMWDAT	Host receive RAM write data. This is the upper byte of receive FIFO RAM data that will be written at the address of **hstrramwadx[11:0]** if **hstrramwadx[13]** is negated and **hstrramwreq** is asserted. This part of the receive FIFO RAM contains control information for the frame as follows: hstrramwdat[35] = frsof hstrramwdat[34] = freof hstrramwdat[33:32] = frdatnvld[1:0]	RW
15:13	Reserved	Writing has no effect, read as zero.	R
12:0	HSTRRAMWADX	Host receive RAM write address. This field has different functionality based on the value of **hstrramwadx[13]** and whether the M-MIIFIF FIFO RAM access register 4 is being written to or read from. When read from, **hstrramwadx[12:0]** field contains the actual write pointer value of the mmiirfif_sys module. When written to, the **hstrramwadx** register will be loaded. If **hstrramwadx[13]** is low, **hstrramwadx[11:0]** will be the receive FIFO RAM address to which **hstrramwdat** is written. If **hstrramwadx[13]** is high, **hstrramwadx[12:0]** contains the pointer value that will be written to the mmiirfif_sys module. The register definition for **hstrramwadx[12:0]** is for a receive RAM with 12 address bits (16K bytes). The register length will vary with the addressable RAM size used. The definition of **hstrramwadx[13]** will shift to remain left-justified in the register.	RW

11.5.4.12 RAM_ACC_R5

RAM_ACC_R5 **0x134B0074**

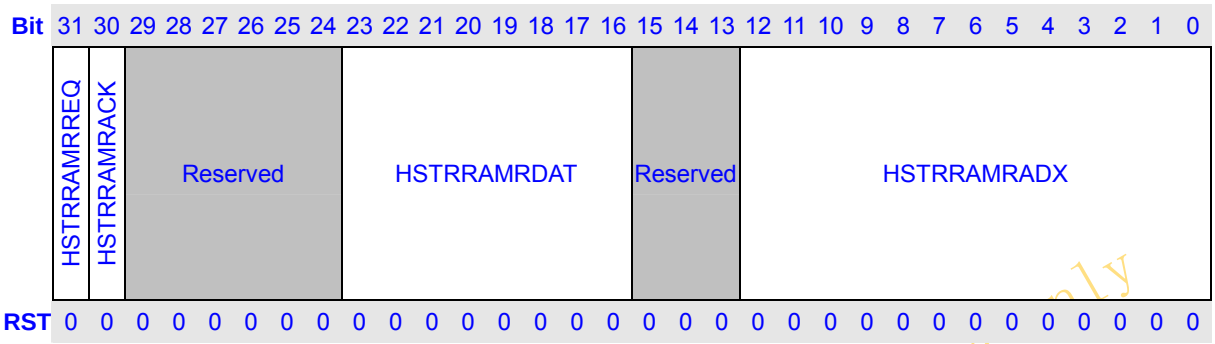
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	HSTRRAMWDAT																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:0	HSTRRAMW	Host receive RAM write data. This is the lower 4 bytes of receive	RW

	DAT	FIFO RAM data that will be written at the address of **hstrramwadx[11:0]** if **hsttramwadx[13]** is negated and **hstrramwreq** is asserted.	

11.5.4.13 RAM_ACC_R6

RAM_ACC_R6 **0x134B0078**



Bits	Name	Description	RW
31	HSTRRAMRREQ	Host receive RAM read request. Requests the handshake of **hstrramradx** values to the receive FIFO RAM and **hstrramrdat** values from the receive FIFO RAM. Should only be asserted while **hstrramrack** is negated and while the receive data path is disabled from receiving data and in a steady state. It should only be negated after receiving an asserted **hstrramrack**.	RW
30	HSTRRAMRACK	Host receive RAM read acknowledge. Signifies the acceptance of **hstrramradx** values to the receive FIFO RAM and reception of **hstrramrdat** from the receive FIFO RAM location addressed. Will only be asserted or negated following assertion or negation of **hstrramrreq**.	R
29:24	Reserved	Writing has no effect, read as zero.	R
23:16	HSTRRAMRDAT	Host receive RAM read data. This is the upper byte of receive FIFO RAM data that was read at the address of **hstrramradx[10:0]** if **hstrramradx[13]** is negated and **hstrramrreq** is asserted. This part of the receive FIFO RAM contains control information for the frame as follows: hstrramrdat[35] = frsof hstrramrdat[34] = freof hstrramrdat[33:32] = frdatnvld[1:0]	R
15:13	Reserved	Writing has no effect, read as zero.	R
12:0	HSTRRAMRADX	Host receive RAM read address. If **hstrramradx[13]** is written low, **hstrramradx[11:0]** is the receive FIFO RAM address that **hstrramrdat** is read from. If **hstrramradx[13]** is written high,	RW

		hstrramradx[12:0] contains the pointer value read from the mmiirfif_fab module. The register definition for **hstrramradx[12:0]** is for a receive RAM with 12 address bits (16K bytes). The register length will vary with the addressable RAM size used. The definition of **hstrramradx[13]** will shift to remain left-justified in the register.	

11.5.4.14 RAM_ACC_R7

RAM_ACC_R7 **0x134B007C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	HSTRRAMRDAT																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:0	HSTRRAMRDAT	Host receive RAM read data. This is the lower 4 bytes of receive FIFO RAM data that is read at the address of **hstrramradx[11:0]** if **hstrramradx[13]** is negated and **hstrramrreq** is asserted.	R

11.6 MII Module

11.6.1 Overview

The MII module is a full function 10 / 100 Mbps Media Access Controller modules with Media Independent Interface and optional interface modules.

11.6.2 Register Map

Name	RW	Reset Value	Address	Access Size
MAC1	RW	0x00008000	0x134B0000	32
MAC2	RW	0x00000000	0x134B0004	32
IPGT	RW	0x00000000	0x134B0008	32
IPGR	RW	0x00000000	0x134B000C	32
CLRT	RW	0x0000370f	0x134B0010	32
MAXF	RW	0x00000600	0x134B0014	32
SUPP	RW	0x00001000	0x134B0018	32
TEST	RW	0x00000000	0x134B001C	32
MCFG	RW	0x00000000	0x134B0020	32
MCMD	RW	0x00000000	0x134B0024	32
MADR	RW	0x00000000	0x134B0028	32
MWTD	RW	0x00000000	0x134B002C	32
MRDD	RW	0x00000000	0x134B0030	32
MIND	RW	0x00000000	0x134B0034	32
SA0	RW	0x00000000	0x134B0040	32
SA1	RW	0x00000000	0x134B0044	32
SA2	RW	0x00000000	0x134B0048	32

11.6.3 Register Description

11.6.3.1 MAC1

MAC1 **0x134B0000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																SFT_RST	SIM_RST	RMII_MODE	Reserved	RST_RMCS	RST_RFUN	RST_TMCS	RST_TFUN	Reserved			LOOPBACK	TX_PAUSE	RX_PAUSE	PASS_ALL	RX_EN
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15	SFT_RST	Setting this bit will put all modules within the PE-MACMII in reset except the Host Interface. The Host Interface is reset via HRST.	RW
14	SIM_RST	Setting this bit will cause a reset to the random number generator within the Transmit Function.	RW
13	RMII_MODE	RMII_MODE.	RW
12:11	Reserved	Writing has no effect, read as zero.	R
11	RST_RMCS	Setting this bit will put the MAC Control Sublayer / Receive domain logic in reset.	RW
10	RST_RFUN	Setting this bit will put the Receive Function logic in reset.	RW
9	RST_TMCS	Setting this bit will put the MAC Control Sublayer / Tx domain logic in reset.	RW
8	RST_TFUN	Setting this bit will put the Transmit Function logic in reset.	RW
7:5	Reserved	Writing has no effect, read as zero.	R
4	LOOPBACK	Setting this bit will cause the MAC Transmit interface to be loop backed to the MAC Receive interface. Clearing this bit results in normal operation.	RW
3	TX_PAUSE	When enabled, PAUSE Flow Control frames are allowed to be transmitted. When disabled, Flow Control frames are blocked.	RW
2	RX_PAUSE	When enabled, the MAC acts upon received PAUSE Flow Control frames. When disabled, received PAUSE Flow Control frames are ignored.	RW
1	PASS_ALL	When enabled, the MAC will indicate PASS CURRENT RECEIVE FRAME for all frames regardless of type (normal vs. Control). When disabled, the MAC de-asserts PASS CURRENT RECEIVE FRAME for valid Control frames.	RW
0	RX_EN	Set this to allow receive frames to be received. Internally the MAC synchronizes this control bit to the incoming receive stream and outputs SYNCHRONIZED RECEIVE ENABLE, to be used by host system to qualify receive frames.	RW

11.6.3.2 MAC2

MAC2 0x134B0004

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																	EXC_DEFER	BP_NB	NO_BACKOFF	Reserved		LONG_PRE	PURE_PRE	AUTO_PAD	VLAN_PAD	PAD_EN	CRC_EN	DLY_CRC	HUGE_FRM	LEN_CK	FULL_DPX
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
14	EXC_DEFER	When enabled the MAC will defer to carrier indefinitely as per the Standard. When disabled, the MAC will abort when the excessive deferral limit is reached and provide feedback to the host system.	RW
13	BP_NB	When enabled, the MAC after incidentally causing a collision during backpressure will immediately retransmit without backoff reducing the chance of further collisions and ensuring transmit packets get sent.	RW
12	NO_BACKOFF	When enabled, the MAC will immediately retransmit following a collision rather than using the Binary Exponential Backoff algorithm as specified in the Standard.	RW
11:10	Reserved	Writing has no effect, read as zero.	R
9	LONG_PRE	When enabled, the MAC only allows receive packets, which contain preamble fields less than 12 bytes in length. When disabled, the MAC allows any length preamble as per the Standard.	RW
8	PURE_PRE	When enable, the MAC will verify the content of the preamble to ensure it contains 0x55 and is error-free. A packet with errored preamble is discarded. When disabled, no preamble checking is performed.	RW
7	AUTO_PAD	Set this bit to cause the MAC to automatically detect the type of frame, either tagged or un-tagged, by comparing the two octets following the source address with 0x8100 (VLAN Protocol ID) and pad accordingly.	RW
6	VLAN_PAD	Set this bit to cause the MAC to pad all short frames to 64 bytes and append a valid CRC.	RW
5	PAD_EN	Set this bit to have the MAC pad all short frames. Clear this bit if frames presented to the MAC have a valid length. This bit is used in conjunction with *AUTO PAD ENABLE* and *VLAN PAD ENABLE*.	RW
4	CRC_EN	Set this bit to append a CRC to every frame whether padding was required or not. Must be set if PAD/CRC ENABLE is set. Clear this bit if frames presented to the MAC contain a CRC.	RW
3	DLY_CRC	This bit determines the number of bytes, if any, of proprietary header information that exists on the front of IEEE 802.3 frames.	RW
2	HUGE_FRM	When enabled frames of any length are transmitted and received.	RW
1	LEN_CK	When enabled, both transmit and receive frame lengths are compared to the Length/Type field. If the Length/Type field represents a length then the check is performed. Mismatches are reported on the Transmit/Receive Statistics Vector.	RW
0	FULL_DPX	When enabled, MAC operates in Full-Duplex mode. Disable for Half-Duplex operation.	RW

11.6.3.3 IPGT

IPGT 0x134B0008

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																									B2B_IPG						
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:7	Reserved	Writing has no effect, read as zero.	R
6:0	B2B_IPG	This is a programmable field representing the nibble time offset of the minimum possible period between the end of any transmitted packet, to the beginning of the next. In Full-Duplex mode, the register value should be the desired period in nibble times minus 3. In Half-Duplex mode, the register value should be the desired period in nibble times minus 6. In Full-Duplex the recommended setting is 0x15 (21d), which represents the minimum IPG of 0.96 µs (in 100 Mb/s) or 9.6 µs (in 10 Mb/s). In Half-Duplex the recommended setting is 0x12 (18d), which also represents the minimum IPG of 0.96 µs (in 100 Mb/s) or 9.6 µs (in 10 Mb/s).	RW

11.6.3.4 IPGR

IPGR 0x134B000C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																	NB2B_IPG_P1							Reserved	NB2B_IPG_P2						
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:15	Reserved	Writing has no effect, read as zero.	R
14:8	NB2B_IPG_P1	This is a programmable field representing the optional carrierSense window referenced in IEEE 802.3/4.2.3.2.1 'Carrier Deference'. If carrier is detected during the timing of IPGR1, the MAC defers to carrier. If, however, carrier becomes active after IPGR1, the MAC continues timing IPGR2 and transmits, knowingly causing a collision,	RW

		thus ensuring fair access to medium. Its range of values is 0x0 to IPGR2. The recommended value is 0xC (12d).	
7	Reserved	Writing has no effect, read as zero.	R
6:0	NB2B_IPG_P2	This is a programmable field representing the Non-Back-to-Back Inter-Packet-Gap. The recommended value is 0x12 (18d), which represents the minimum IPG of 0.96 µs (in 100 Mb/s) or 9.6 µs (in 10 Mb/s).	RW

11.6.3.5 CLRT

CLRT **0x134B0010**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																		CW						Reserved				RT_MAX			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	1	1	1	1

Bits	Name	Description	RW
31:14	Reserved	Writing has no effect, read as zero.	R
13:8	CW	This is a programmable field representing the slot time or collision window during which collisions occur in properly configured networks. Since the collision window starts at the beginning of transmission, the preamble and SFD is included. Its default of 0x37 (55d) corresponds to the count of frame bytes at the end of the window.	RW
7:4	Reserved	Writing has no effect, read as zero.	R
3:0	RT_MAX	This is a programmable field specifying the number of retransmission attempts following a collision before aborting the packet due to excessive collisions. The Standard specifies the attemptLimit to be 0xF (15d). Its default is '0xF'.	RW

11.6.3.6 MAXF

MAXF **0x134B0014**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																MAXF															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

11.6.3.7 RAM_ACC_R7

Bits	Name	Description	RW
31:16	Reserved	Writing has no effect, read as zero.	R
15:0	MAXF	This field resets to 0x0600, which represents a maximum receive frame of 1536 octets. An untagged maximum size Ethernet frame is 1518 octets. A tagged frame adds four octets for a total of 1522 octets. If a shorter maximum length restriction is desired, program this 16-bit field.	RW

11.6.3.8 SUPP

SUPP 0x134B0018

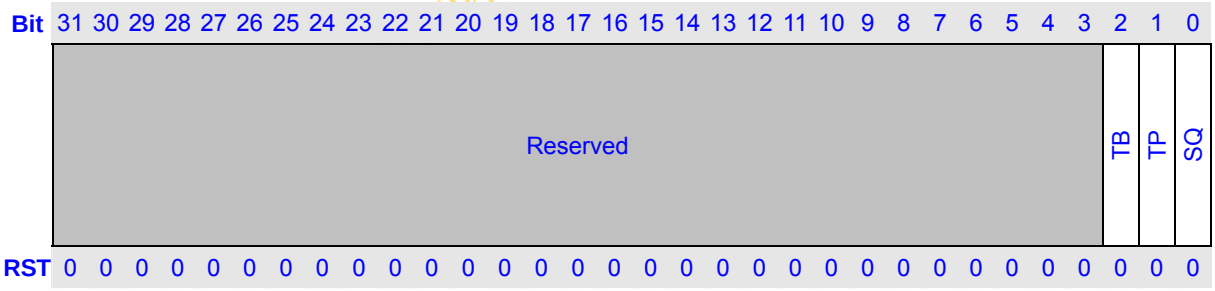
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																RST_INT	Reserved		PHY_MODE	RST_RMII	Reserved		SPEED	RST_100X	FQ	NC	LF	RST_10T	Reserved	EN_JAB	BM
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:16	Reserved	Writing has no effect, read as zero.	R
15	RST_INT	Setting this bit resets the attached Interface module. Clearing this bit allows for normal operation. This bit can be used in place of bits 11, 7, and 3 when connecting only 1 interface module.	RW
14:13	Reserved	Writing has no effect, read as zero.	R
12	PHY_MODE	This bit configures the Serial MII logic with the connecting SMII device type. Set this bit when connecting to a SMII PHY. Clear this bit when connecting to a SMII MAC. When MAC is selected, the SMII will operate at 100 Mb/s, Full Duplex.	RW
11	RST_RMII	This bit resets the Reduced MII logic.	RW
10:9	Reserved	Writing has no effect, read as zero.	R
8	SPEED	This bit configures the Reduced MII logic for the current operating speed. When set, 100 Mb/s mode is selected. When cleared, 10 Mb/s mode is selected.	RW
7	RST_100X	This bit resets the PE100X module, which contains the 4B/5B symbol encipher/decipher logic. Effects PE100X module only.	RW
6	FQ	When enabled, transmit data is quieted which allows the contents of the cipher to be output. When cleared, normal operation is enabled. Effects PE100X module only.	RW

5	NC	When enabled, the raw transmit 5B symbols are transmitted without ciphering. When disabled, normal ciphering occurs. Effects PE100X module only.	RW
4	LF	When enabled, the 330ms Link Fail timer is disabled allowing for shorter simulations. Removes the 330 µS link-up time before reception of streams is allowed. When cleared, normal operation occurs. Effects PE100X module only.	RW
3	RST_10T	This bit resets the PE10T module which converts MII nibble streams to the serial bit stream of 10T transceivers. Effects PE10T module only.	RW
2	Reserved	Writing has no effect, read as zero.	R
1	EN_JAB	This bit enables the Jabber Protection logic within the PE10T in ENDEC mode. Jabber is the condition where a transmitter is stuck on for longer than 50ms preventing other stations from transmitting. Effects PE10T module only.	RW
0	BM	When '1' - MAC is in 10BASE-T ENDEC mode which changes decodes (such as Excess Defer) to be based on the bit clock rather than the nibble clock.	RW

11.6.3.9 TEST

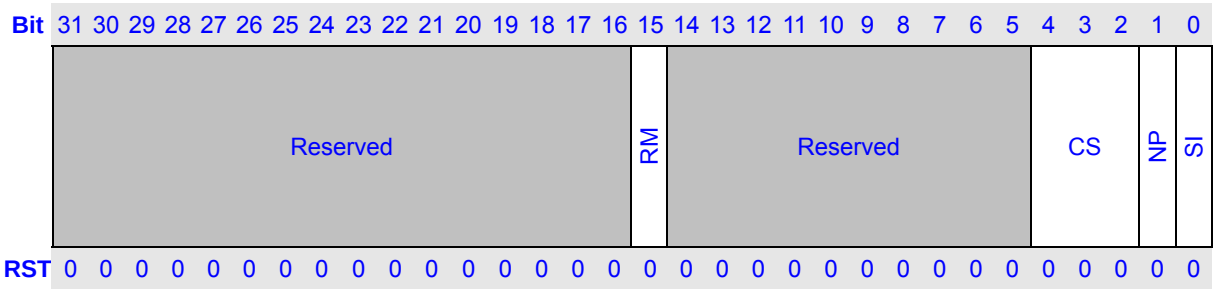
TEST 0x134B001C



Bits	Name	Description	RW
31:3	Reserved	Writing has no effect, read as zero.	R
2	TB	Setting this bit will cause the MAC to assert backpressure on the link. Backpressure causes preamble to be transmitted, raising carrier sense. A transmit packet from the system will be sent during backpressure.	RW
1	TP	This bit causes the MAC Control sublayer to inhibit transmissions, just as if a PAUSE Receive Control frame with a non-zero pause time parameter was received.	RW
0	SQ	This bit reduces the effective PAUSE Quanta from 64 byte-times to 1 byte-time.	RW

11.6.3.10 MCFG

MCFG 0x134B0020



Bits	Name	Description	RW
31:16	Reserved	Writing has no effect, read as zero.	R
15	RM	This bit resets the MII Management module.	RW
14:5	Reserved	Writing has no effect, read as zero.	R
4:2	CS	This field is used by the clock divide logic in creating the MII Management Clock (MDC) which IEEE 802.3u defines to be no faster than 2.5 MHz. Some PHYs support clock rates up to 12.5 MHz, however.	RW
1	NP	Set this bit to cause the MII Management module to perform read/write cycles without the 32-bit preamble field. Clear this bit to cause normal cycles to be performed. Some PHYs support suppressed preamble.	RW
0	SI	Set this bit to cause the MII Management module to perform read cycles across a range of PHYs. When set, the MII Management module will perform read cycles from address 1 through the value set in PHY ADDRESS[4:0]. Clear this bit to allow continuous reads of the same PHY.	RW

Clock Select	4	3	2
Host Clock divided by 4	0	0	x
Host Clock divided by 6	0	1	0
Host Clock divided by 8	0	1	1
Host Clock divided by 10	1	0	0
Host Clock divided by 14	1	0	1
Host Clock divided by 20	1	1	0
Host Clock divided by 28	1	1	1

11.6.3.11 MCMD

MCMD 0x134B0024

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																														SCAN	READ
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:2	Reserved	Writing has no effect, read as zero.	R
1	SCAN	This bit causes the MII Management module to perform Read cycles continuously. This is useful for monitoring Link Fail for example.	RW
0	READ	This bit causes the MII Management module to perform a single Read cycle. The Read data is returned in Register 0xC (MII Mgmt Read Data).	RW

11.6.3.12 MADR

MADR 0x134B0028

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																			PHY_ADR					Reserved			REG_ADR				
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:13	Reserved	Writing has no effect, read as zero.	R
12:8	PHY_ADR	This field represents the 5-bit PHY Address field of Mgmt cycles. Up to 31 PHYs can be addressed (0 is reserved).	RW
7:5	Reserved	Writing has no effect, read as zero.	R
4:0	REG_ADR	This field represents the 5-bit Register Address field of Mgmt cycles. Up to 32 registers can be accessed.	RW

11.6.3.13 MWTD

MWTD 0x134B002C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																WDATA															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:16	Reserved	Writing has no effect, read as zero.	R
15:0	WDATA	When written, a MII Mgmt write cycle is performed using the 16-bit data and the pre-configured PHY and Register addresses from Register (0x0A).	W

11.6.3.14 MRDD

MRDD 0x134B0030

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																RDATA															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:16	Reserved	Writing has no effect, read as zero.	R
15:0	RDATA	Following a MII Mgmt Read Cycle, the 16-bit data can be read from this location.	R

11.6.3.15 MIND

MIND 0x134B0034

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																												LF	NVLD	SCAN	BUSY
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:4	Reserved	Writing has no effect, read as zero.	R
3	LF	When '1' is returned - indicates MII Mgmt link fail has occurred.	R
2	NVLD	When '1' is returned - indicates MII Mgmt Read cycle has not completed and the Read Data is not yet valid.	R
1	SCAN	When '1' is returned - indicates a scan operation (continuous MII Mgmt Read cycles) is in progress.	R
0	BUSY	When '1' is returned - indicates MII Mgmt module is currently performing an MII Mgmt Read or Write cycle.	R

11.6.3.16 SA0

SA0 0x134B0040

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																SADD1								SADD2								
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:16	Reserved	Writing has no effect, read as zero.	R
15:8	SADD1	This field holds the first octet of the station address.	RW
7:0	SADD2	This field holds the second octet of the station address.	RW

11.6.3.17 SA1

SA1 0x134B0044

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																SADD3								SADD4								
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:16	Reserved	Writing has no effect, read as zero.	R
15:8	SADD3	This field holds the third octet of the station address.	RW
7:0	SADD4	This field holds the fourth octet of the station address.	RW

11.6.3.18 SA2

SA2 0x134B0048

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																SADD5								SADD6							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:16	Reserved	Writing has no effect, read as zero.	R
15:8	SADD5	This field holds the fifth octet of the station address.	RW
7:0	SADD6	This field holds the sixth octet of the station address.	RW

11.7 RMII Module

11.7.1 Overview

The Reduced-MII was designed to convey the complete 16-bit MII information between a 10/100 Mb/s MAC and an RMII PHY with 7 pins per port.
The RMII module converts the Media Independent Interface to the new and improved Reduced Media Independent Interface for use with new lower pin count PHYs. The RMII adheres to the Reduced-MII Specification version 1.2, dated March 20, 1998.

11.7.2 Feature

- Provides reduced pin-count interface for Ethernet physical layer devices
- Supports Preliminary Draft Version 1.2 of the RMII Specification
- Supports Full and Half duplex operation
- Supports 10 and 100 Mb/s operation

11.8 SAL Module

11.8.1 Overview

The SAL module, Station Address Logic module, accepts the destination address field of incoming packets and all the related control signals and performs an address comparison.

Two types of valid destination addresses exist; a single physical address and a multicast address. When a single physical destination address is received, the PE-SAL module performs a direct comparison between the destination address and the address stored in the 48-bit Station Address Register. The 48-bit Station Address Register is loaded by the host when the module is first configured.

When a multicast destination address is received, a hash algorithm is applied to the address after it goes through a 32-bit CRC checker. The upper 6 bits of the 32-bit result from the CRC checker is decoded and used to select a bit in a 64-bit hash table which is pre-configured by the host. If the address compares correctly, the signal ACCEPT is set high to reflect a successful address comparison and the packet will be flagged as accepted. Otherwise, REJECT will be set high to reject the packet. The PE-SAL module can also be programmed to accept all incoming packets (promiscuous mode), all packets with multicast address or all packets with broadcast address.

In addition to the above functionality, the PE-SAL module facilitates insertion of its internal 48-bit Station Address Register into the system transmit data stream through byte multiplexed enabled output.

11.8.2 Register Map

Name	RW	Reset Value	Address	Access Size
AFR	RW	0x00000000	0x134B01A0	32
HT1	RW	0x00000000	0x134B01A4	32
HT2	RW	0x00000000	0x134B01A8	32

11.8.3 Register Description

11.8.3.1 AFR

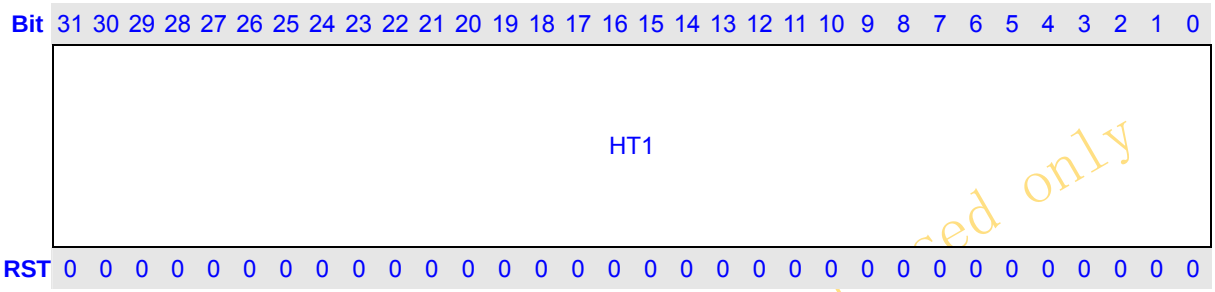
AFR 0x134B01A0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																												PRO	PRM	AMC	ABC
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:4	Reserved	Writing has no effect, read as zero.	R
3	PRO	Promiscuous Mode.	RW
2	PRM	Accept Multicast.	RW
1	AMC	Accept Multicast (qualified).	RW
0	ABC	Accept Broadcast.	RW

11.8.3.2 HT1

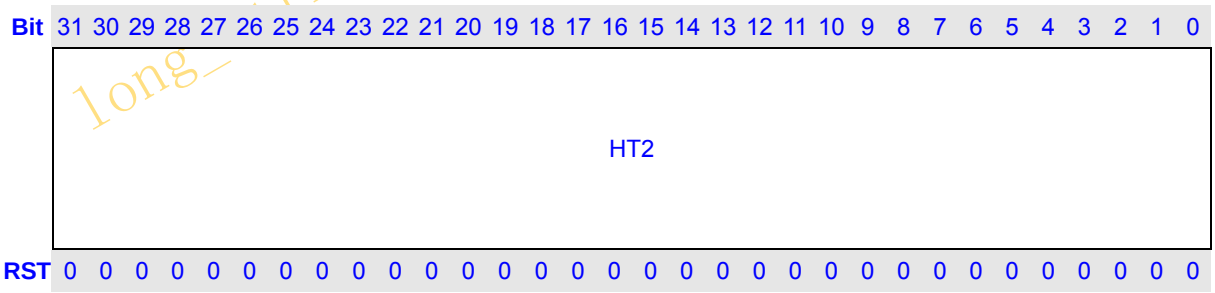
HT1 0x134B01A4



Bits	Name	Description	RW
31:0	HT1	Hash Table [64:32].	RW

11.8.3.3 HT2

HT2 0x134B01A8



Bits	Name	Description	RW
31:4	HT2	Hash Table [31:0].	RW

11.9 STAT Module

11.9.1 Overview

The STAT module is a low gate count, register based, statistics gathering module. It has 37 separate counters, which simply count or accumulate conditions that occur as packets are transmitted and received. These counters will support RMON MIB group 1, RMON MIB group 2 if table counters, RMON MIB group 3, RMON MIB group 9, RMON MIB 2, and the dot 3 Ethernet MIB.

On the transmit side, the STAT module looks at bits from the Transmit Statistics Vector (TSV[51:0]) whenever Transmit Statistics Vector Pulse (TSVP) is asserted. A particular MACs Transmit Function sub-module generates the Transmit Statistics Vector.

During receive packets, the Receive Statistics Vector (RSV[30:0]) is examined whenever the Receive Statistics Vector Pulse (RSVP) is asserted. A particular MACs Receive Function sub-module generates the Receive Statistics Vector.

One or more non-zero elements in either of these statistics vectors trigger the STAT module to update its statistics counters. These counters are stored in internal data registers. The host can access the internal data registers at any time. The STAT is capable of maintaining worst-case throughput conditions of one RSV and/or one TSV per four STAT internal clocks.

The ECU may be interrupted upon any one counter's rollover condition via a carry interrupt output from the STAT. Each counters rollover condition can be discreetly masked from causing an interrupt by internal masking registers. In addition, each individual counter value may be reset on read access, or all counters may be simultaneously reset by assertion of an external module input pin.

11.9.2 Register Map

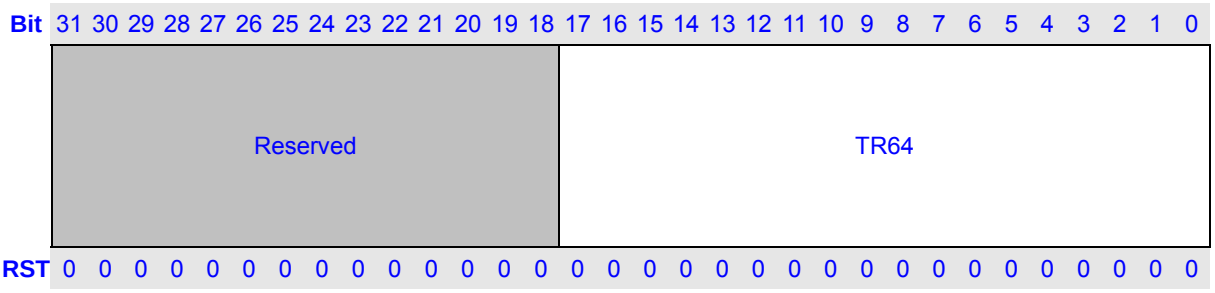
Name	RW	Reset Value	Address	Access Size
TR64	RW		0x134B0080	32
TR127	RW		0x134B0084	32
TR255	RW		0x134B0088	32
TR511	RW		0x134B008C	32
TR1K	RW		0x134B0090	32
TRMAX	RW		0x134B0094	32
TRMGV	RW		0x134B0098	32
RBYT	RW		0x134B009C	32
RPKT	RW		0x134B00A0	32
RFCS	RW		0x134B00A4	32
RMCA	RW		0x134B00A8	32
RBCA	RW		0x134B00AC	32

RXCF	RW		0x134B00B0	32
RXPF	RW		0x134B00B4	32
RXUO	RW		0x134B00B8	32
RALN	RW		0x134B00BC	32
RFLR	RW		0x134B00C0	32
RCDE	RW		0x134B00C4	32
RCSE	RW		0x134B00C8	32
RUND	RW		0x134B00CC	32
ROVR	RW		0x134B00D0	32
RFRG	RW		0x134B00D4	32
RJBR	RW		0x134B00D8	32
RDRP	RW		0x134B00DC	32
TBYT	RW		0x134B00E0	32
TPKT	RW		0x134B00E4	32
TMCA	RW		0x134B00E8	32
TBCA	RW		0x134B00EC	32
TXPF	RW		0x134B00F0	32
TDFR	RW		0x134B00F4	32
TEDF	RW		0x134B00F8	32
TSCL	RW		0x134B00FC	32
TMCL	RW		0x134B0100	32
TLCL	RW		0x134B0104	32
TXCL	RW		0x134B0108	32
TNCL	RW		0x134B010C	32
TPFH	RW		0x134B0110	32
TDRP	RW		0x134B0114	32
TJBR	RW		0x134B0118	32
TFCS	RW		0x134B011C	32
TXCF	RW		0x134B0120	32
TOVR	RW		0x134B0124	32
TUND	RW		0x134B0128	32
TFRG	RW		0x134B012C	32
CAR1	R		0x134B0130	32
CAR2	R		0x134B0134	32
CAM1	RW		0x134B0138	32
CAM2	RW		0x134B013C	32

11.9.3 Register Description

11.9.3.1 TR64

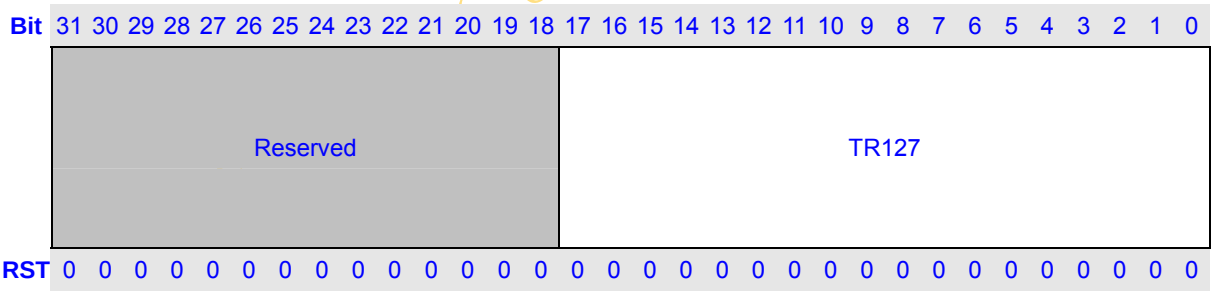
TR64 0x134B0080



Bits	Name	Description	RW
31:18	Reserved	Writing has no effect, read as zero.	R
17:0	TR64	Transmit and Receive 64 Byte Frame Counter: Incremented for each good or bad frame transmitted and received which is 64 bytes in length inclusive (excluding framing bits but including FCS bytes).	RW

11.9.3.2 TR127

TR127 0x134B0084



Bits	Name	Description	RW
31:18	Reserved	Writing has no effect, read as zero.	R
17:0	TR127	Transmit and Receive 65 to 127 Byte Frame Counter: Incremented for each good or bad frame transmitted and received which is 65 to 127 bytes in length inclusive (excluding framing bits but including FCS bytes).	RW

11.9.3.3 TR255

TR255 0x134B0088

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved														TR255																	
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:18	Reserved	Writing has no effect, read as zero.	R
17:0	TR255	Transmit and Receive 128 to 255 Byte Frame Counter: Incremented for each good or bad frame transmitted and received which is 128 to 255 bytes in length inclusive (excluding framing bits but including FCS bytes).	RW

11.9.3.4 TR511

TR511 0x134B008C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved														TR511																	
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:18	Reserved	Writing has no effect, read as zero.	R
17:0	TR511	Transmit and Receive 256 to 511 Byte Frame Counter: Incremented for each good or bad frame transmitted and received which is 256 to 511 bytes in length inclusive (excluding framing bits but including FCS bytes).	RW

11.9.3.5 TR1K

TR1K 0x134B0090

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved														TR1K																	
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:18	Reserved	Writing has no effect, read as zero.	R
17:0	TR1K	Transmit and Receive 512 to 1023 Byte Frame Counter: Incremented for each good or bad frame transmitted and received which is 512 to 1023 bytes in length inclusive (excluding framing bits but including FCS bytes).	RW

11.9.3.6 TRMAX

TRMAX 0x134B0094

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved														TRMAX																	
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:18	Reserved	Writing has no effect, read as zero.	R
17:0	TRMAX	Transmit and Receive 1024 to 1518 Byte Frame Counter: Incremented for each good or bad frame transmitted and received which is 1024 to 1518 bytes in length inclusive (excluding framing bits but including FCS bytes).	RW

11.9.3.7 TRMGV

TRMGV 0x134B0098

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																TRMGV															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:18	Reserved	Writing has no effect, read as zero.	R
17:0	TRMGV	Transmit and Receive 1519 to 1522 Byte VLAN Frame Counter: Incremented for each good VLAN frame transmitted and received which is 1519 to 1522 bytes in length inclusive (excluding framing bits but including FCS bytes).	RW

11.9.3.8 RBYT

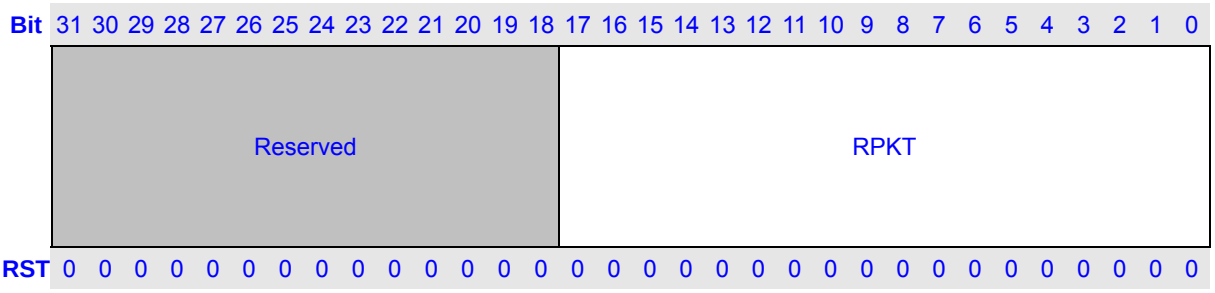
RBYT 0x134B009C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved									RBYT																						
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:23	Reserved	Writing has no effect, read as zero.	R
22:0	RBYT	Receive Byte Counter: The Statistic Counter register is incremented by the byte count of all frames received, including those in bad packets, excluding framing bits but including FCS bytes.	RW

11.9.3.9 RPKT

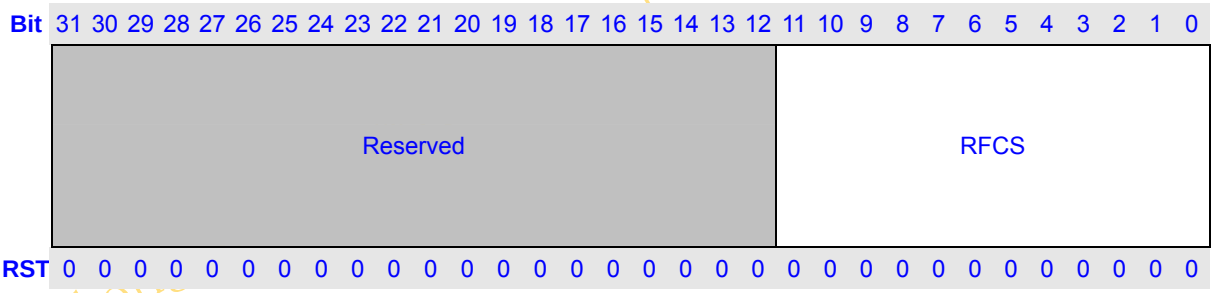
RPKT 0x134B00A0



Bits	Name	Description	RW
31:18	Reserved	Writing has no effect, read as zero.	R
17:0	RPKT	Receive Packet Counter: Incremented for each frame received packet (including bad packets, all Unicast, Broadcast, and Multicast packets).	RW

11.9.3.10 RFCS

RFCS 0x134B00A4



Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	RFCS	Receive FCS Error Counter: Incremented for each frame received that has a integral 64 to 1518 length and contains a Frame Check Sequence error.	RW

11.9.3.11 RMCA

RMCA 0x134B00A8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved														RMCA																	
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:18	Reserved	Writing has no effect, read as zero.	R
17:0	RMCA	Receive Multicast Packet Counter: Incremented for each Multicast good frame of lengths smaller than 1518 (non VLAN) or 1522 (VLAN) excluding Broadcast frames. This does not look at range/length errors.	RW

11.9.3.12 RBCA

RBCA 0x134B00AC

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved														RBCA																	
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:22	Reserved	Writing has no effect, read as zero.	R
21:0	RBCA	Receive Broadcast Packet Counter: Incremented for each Broadcast good frame of lengths smaller than 1518 (non VLAN) or 1522 (VLAN) excluding Multicast frames. This does not look at range/length errors.	RW

11.9.3.13 RXCF

RXCF 0x134B00B0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved														RXCF																	
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:18	Reserved	Writing has no effect, read as zero.	R
17:0	RXCF	Receive Control Frame Packet Counter: Incremented for each MAC Control frame received (PAUSE & Unsupported).	RW

11.9.3.14 RXPF

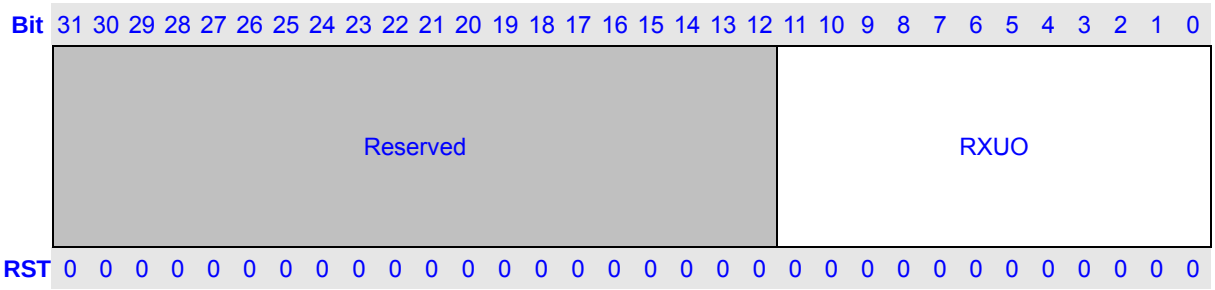
RXPF 0x134B00B4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				RXPF											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	RXPF	Receive PAUSE Frame Packet Counter: Incremented each time a valid PAUSE MAC Control frame is received.	RW

11.9.3.15 RXUO

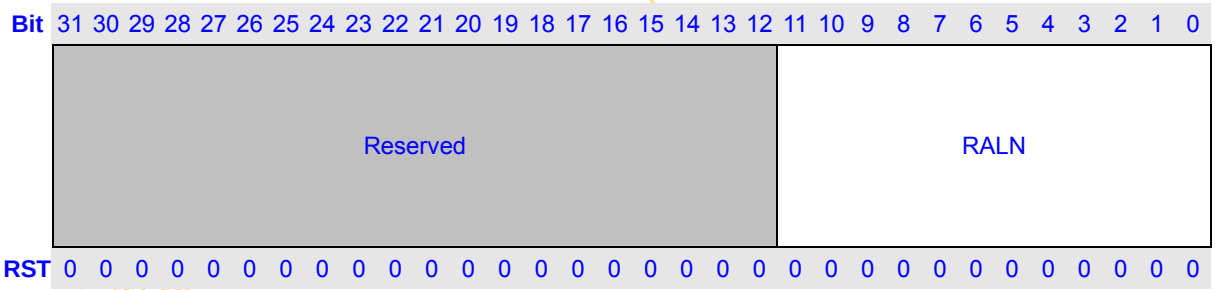
RXUO 0x134B00B8



Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	RXUO	Receive Unknown OPcode Counter: Incremented each time a MAC Control Frame is received which contains an opcode other than a PAUSE.	RW

11.9.3.16 RALN

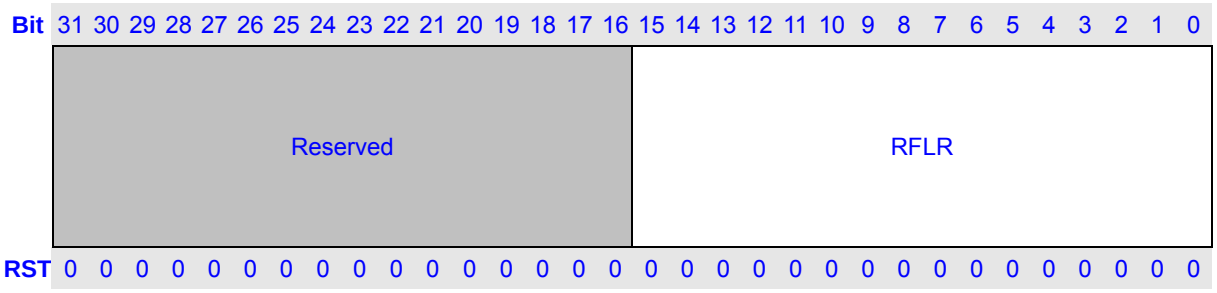
RALN 0x134B00BC



Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	RALN	Receive Alignment Error Counter: Incremented for each received frame from 64 to 1518 which contains an invalid FCS and is not an integral number of bytes.	RW

11.9.3.17 RFLR

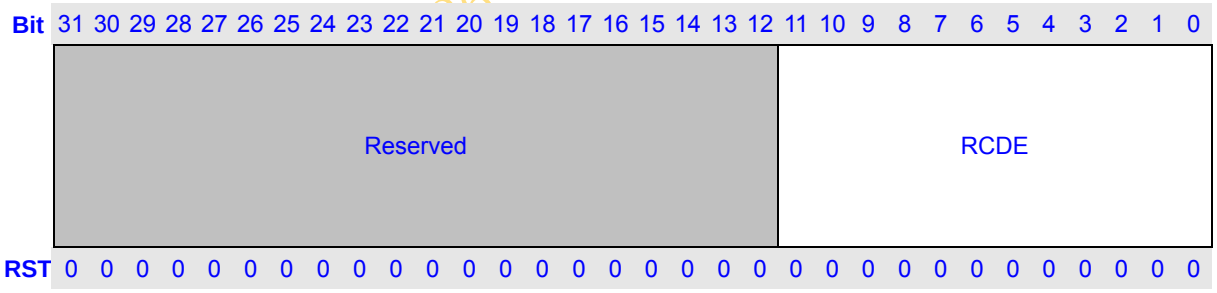
RFLR 0x134B00C0



Bits	Name	Description	RW
31:16	Reserved	Writing has no effect, read as zero.	R
15:0	TR127	Receive Frame Length Error Counter: Incremented for each frame received in which the 802.3 length field did not match the number of data bytes actually received (46 - 1500 bytes). The counter is not incremented if the length field is not a valid 802.3 length, such as an EtherType value.	RW

11.9.3.18 RCDE

RCDE 0x134B00C4



Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	RCDE	Receive Code Error Counter: Incremented each time a valid carrier was present and at least one invalid data symbol was detected.	RW

11.9.3.19 RCSE

RCSE 0x134B00C8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				RCSE											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	RCSE	Receive False Carrier Counter: Incremented each time a false carrier is detected during idle, as defined by a 1 on RX_ER and an '0xE' on RXD. The event is reported along with the statistics generated on the next received frame. Only one false carrier condition can be detected and logged between frames.	RW

11.9.3.20 RUND

RUND 0x134B00CC

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				RUND											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	RUND	Receive Undersize Packet Counter: Incremented each time a frame is received which is less than 64 bytes in length and contains a valid FCS and were otherwise well formed. This does not look at Range Length errors.	RW

11.9.3.21 ROVR

ROVR 0x134B00D0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				ROVR											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	ROVR	Receive Oversize Packet Counter: Incremented each time a frame is received which exceeded 1518 (non VLAN) or 1522 (VLAN) and contains a valid FCS and were otherwise well formed. This does not look at Range Length errors.	RW

11.9.3.22 RFRG

RFCS 0x134B00D4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				RFRG											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	RFRG	Receive Fragments Counter: Incremented for each frame received which is less than 64 bytes in length and contains an invalid FCS, includes integral and non-integral lengths.	RW

11.9.3.23 RJBR

RJBR 0x134B00D8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				RJBR											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	RJBR	Receive Jabber Counter: Incremented for frames received which exceed 1518 (non VLAN) or 1522 (VLAN) bytes and contains an invalid FCS, includes alignment errors.	RW

11.9.3.24 RDRP

RDRP 0x134B00DC

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				RDRP											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	RDRP	Receive Dropped packets Counter: Incremented for frames received which are streamed to system but are later dropped due to lack of system resources.	RW

11.9.3.25 TBYT

TBYT 0x134B00E0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								TBYT																							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:24	Reserved	Writing has no effect, read as zero.	R
23:0	TBYT	Transmit Byte Counter: Incremented by the number of bytes that were put on the wire including fragments of frames that were involved with collisions. This count does not include preamble/SFD or jam bytes.	RW

11.9.3.26 TPKT

TPKT 0x134B00E4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved														TPKT																	
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:18	Reserved	Writing has no effect, read as zero.	R
17:0	TPKT	Transmit Packet Counter: Incremented for each transmitted packet (including bad packets, excessive deferred packets, excessive collision packets, late collision packets, all Unicast, Broadcast, and Multicast packets).	RW

11.9.3.27 TMCA

TMCA 0x134B00E8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																TMCA															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:18	Reserved	Writing has no effect, read as zero.	R
17:0	TMCA	Transmit Multicast Packet Counter: Incremented for each Multicast valid frame transmitted (excluding Broadcast frames).	RW

11.9.3.28 TBCA

TBCA 0x134B00EC

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																TBCA															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:18	Reserved	Writing has no effect, read as zero.	R
17:0	TBCA	Transmit Broadcast Packet Counter: Incremented for each Broadcast frame transmitted (excluding Multicast frames).	RW

11.9.3.29 TXPF

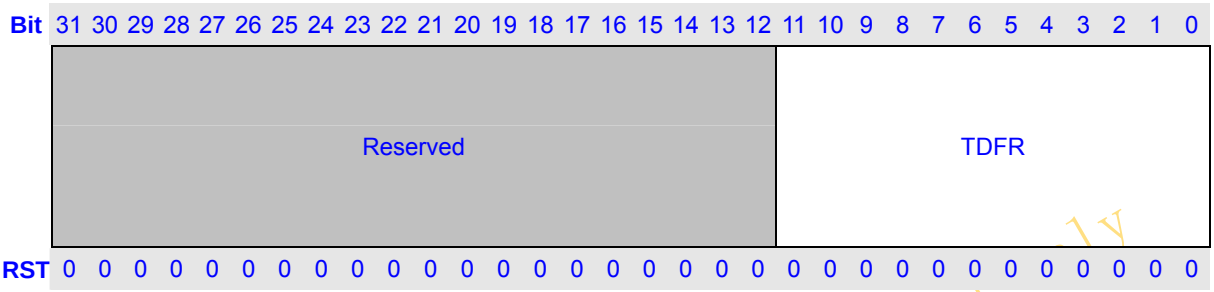
TXPF 0x134B00F0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				TXPF											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	TXPF	Transmit PAUSE Frame Packet Counter: Incremented each time a valid PAUSE MAC Control frame is transmitted.	RW

11.9.3.30 TDFR

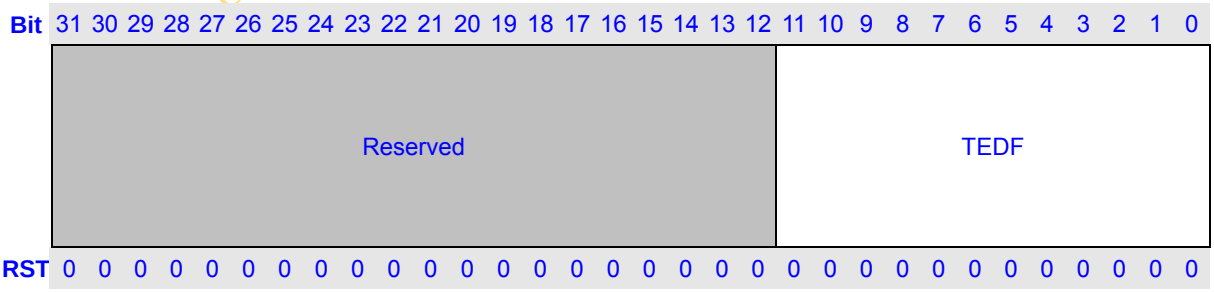
TDFR 0x134B00F4



Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	TDFR	Transmit Deferral Packet Counter: Incremented for each frame, which was deferred on its first transmission attempt. Does not include frames involved in collisions.	RW

11.9.3.31 TEDF

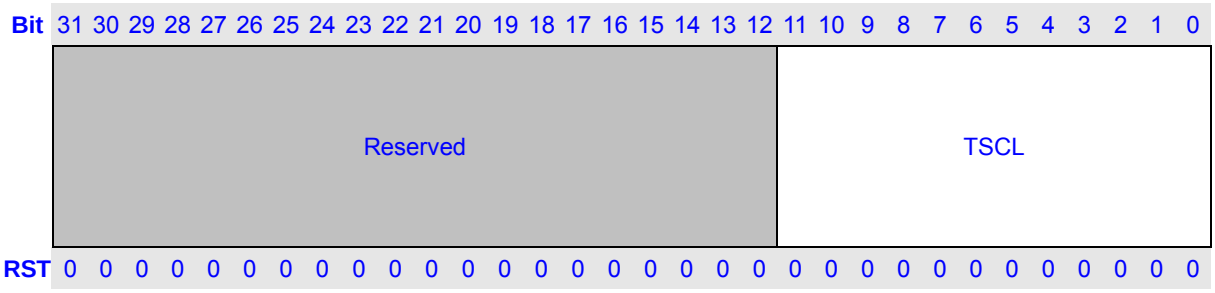
TEDF 0x134B00F8



Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	TEDF	Transmit Excessive Deferral Packet Counter: Incremented for frames aborted which were deferred for an excessive period of time (3036 byte times).	RW

11.9.3.32 TSCL

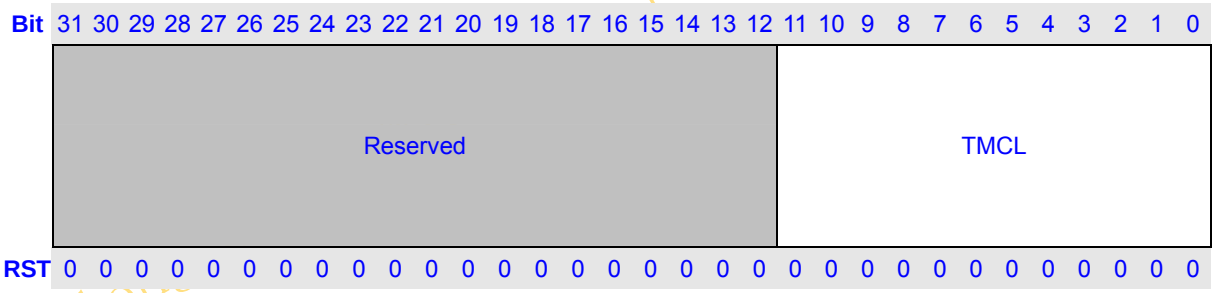
TSCL 0x134B00FC



Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	TSCL	Transmit Single Collision Packet Counter: Incremented for each frame transmitted which experienced exactly one collision during transmission.	RW

11.9.3.33 TMCL

TMCL 0x134B0100



Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	TMCL	Transmit Multiple Collision Packet Counter: Incremented for each frame transmitted which experienced 2-15 collisions (including any late collisions) during transmission as defined using the RETRY[3:0] field of the TX function control register.	RW

11.9.3.34 TLCL

TLCL 0x134B0104

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				TLCL											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	TLCL	Transmit Late Collision Packet Counter: Incremented for each frame transmitted which experienced a late collision during a transmission attempt. Late collisions are defined using the LCOL[5:0] field of the TX Function control register.	RW

11.9.3.35 TXCL

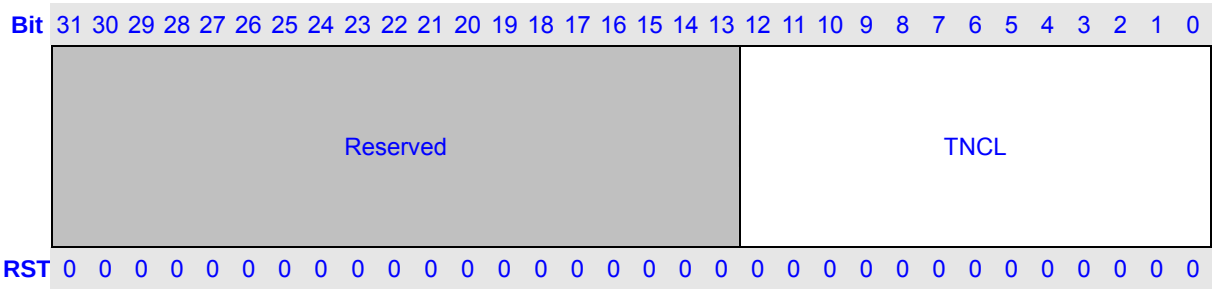
TXCL 0x134B0108

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				TXCL											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	TXCL	Transmit Excessive Collision Packet Counter: Incremented for each frame that experienced 16 collisions during transmission and was aborted.	RW

11.9.3.36 TNCL

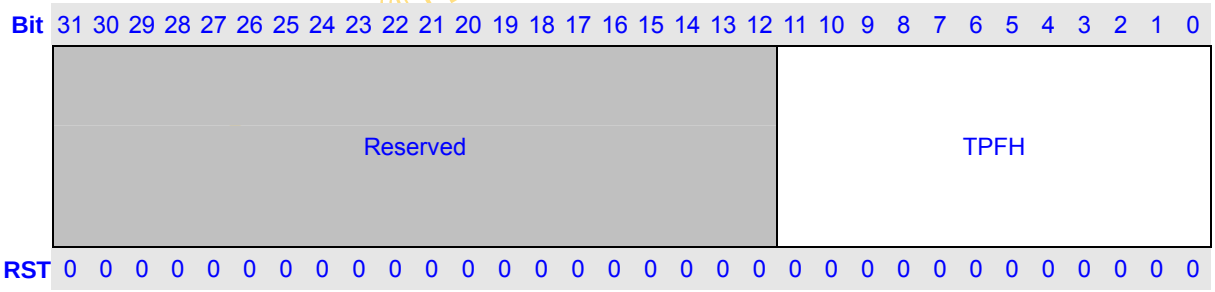
TNCL 0x134B010C



Bits	Name	Description	RW
31:13	Reserved	Writing has no effect, read as zero.	R
12:0	TNCL	Transmit Total Collision Counter: Incremented by the number of collisions experienced during the transmission of a frame as defined as the simultaneous presence of signals on the DO and RD circuits (i.e. transmitting and receiving at the same time). Note, this register does not include collisions that result in an excessive collision condition.	RW

11.9.3.37 TPFH

TPFH 0x134B0110



Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	TPFH	Transmit PAUSE Frames Honored Counter: Incremented each time a valid PAUSE MAC Control frame is transmitted and honored.	RW

11.9.3.38 TDRP

TDRP 0x134B0114

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				TDRP											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	TDRP	Transmit Drop Frame Counter: Incremented each time input PFH is asserted.	RW

11.9.3.39 TJBR

TJBR 0x134B0118

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				TJBR											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	TJBR	Transmit Jabber Frame Counter: Incremented for each oversized transmitted frame with an incorrect FCS value.	RW

11.9.3.40 TFCS

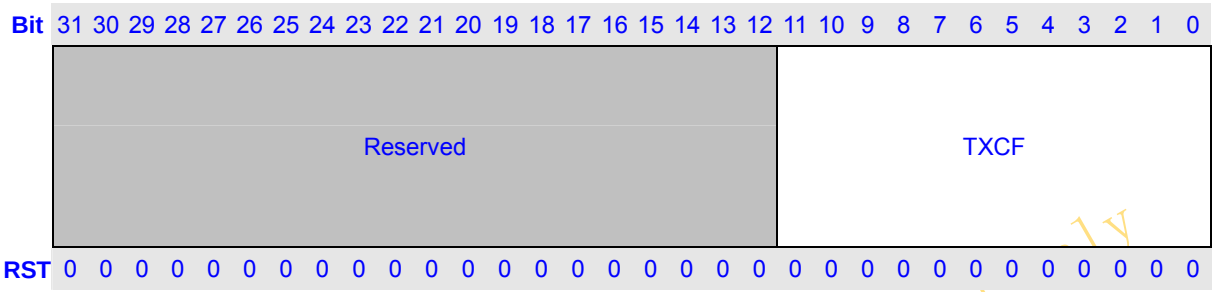
TFCS 0x134B011C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				TFCS											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	TFCS	Transmit FCS Error Counter: Incremented for every valid sized packet with an incorrect FCS value.	RW

11.9.3.41 TXCF

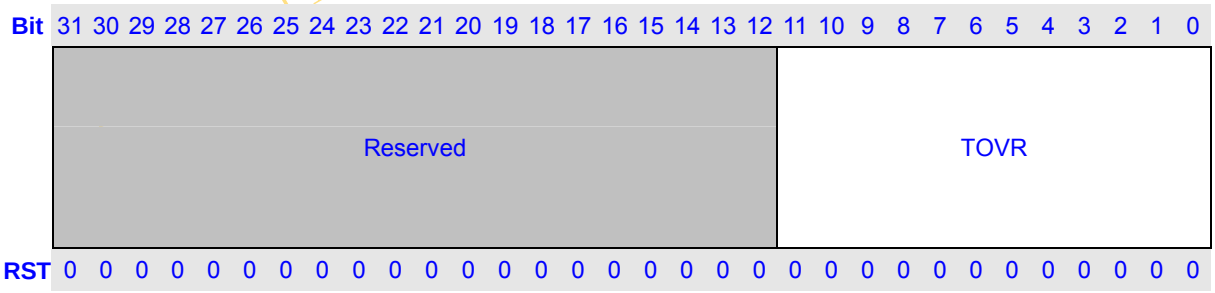
TXCF 0x134B0120



Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	TXCF	Transmit Control Frame Counter: Incremented for every valid size frame with a Type Field signifying a Control frame.	RW

11.9.3.42 TOVR

TOVR 0x134B0124



Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	TOVR	Transmit Oversize Frame Counter: Incremented for each oversized transmitted frame with an correct FCS value.	RW

11.9.3.43 TUND

TUND 0x134B0128

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				TUND											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	TUND	Transmit Undersize Frame Counter: Incremented for every frame less then 64 bytes, with a correct FCS value.	RW

11.9.3.44 TFRG

TFRG 0x134B012C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				TFRG											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	TFRG	Transmit Fragment Counter: Incremented for every frame less then 64 bytes, with an incorrect FCS value.	RW

11.9.3.45 CAR1

CAR1 0x134B0130

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	C164	C1127	C1255	C1511	C11K	C1MAX	C1MGV	Reserved								C1RBY	C1RPK	C1RFC	C1RMC	C1RBC	C1RXC	C1RXP	C1RXU	C1RAL	C1RFL	C1RCD	C1RCS	C1RUN	C1ROV	C1RFR	C1RJB	C1RDR
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	C164	Carry register 1 TR64 Counter Carry bit.	R
30	C1127	Carry register 1 TR127 Counter Carry bit.	R
29	C1255	Carry register 1 TR255 Counter Carry bit.	R
28	C1511	Carry register 1 TR511 Counter Carry bit.	R
27	C11K	Carry register 1 TR1K Counter Carry bit.	R
26	C1MAX	Carry register 1 TRMAX Counter Carry bit.	R
25	C1MGV	Carry register 1 TRMGV Counter Carry bit.	R
24:17	Reserved	Writing has no effect, read as zero.	R
16	C1RBY	Carry register 1 RBYT Counter Carry bit.	R
15	C1RPK	Carry register 1 RPKT Counter Carry bit.	R
14	C1RFC	Carry register 1 RFCS Counter Carry bit.	R
13	C1RMC	Carry register 1 RMCA Counter Carry bit.	R
12	C1RBC	Carry register 1 RBCA Counter Carry bit.	R
11	C1RXC	Carry register 1 RXCF Counter Carry bit.	R
10	C1RXP	Carry register 1 RXPF Counter Carry bit.	R
9	C1RXU	Carry register 1 RXUO Counter Carry bit.	R
8	C1RAL	Carry register 1 RALN Counter Carry bit.	R
7	C1RFL	Carry register 1 RFLR Counter Carry bit.	R
6	C1RCD	Carry register 1 RCDE Counter Carry bit.	R
5	C1RCS	Carry register 1 RCSE Counter Carry bit.	R
4	C1RUN	Carry register 1 RUND Counter Carry bit.	R
3	C1ROV	Carry register 1 ROVR Counter Carry bit.	R
2	C1RFR	Carry register 1 RFRG Counter Carry bit.	R
1	C1RJB	Carry register 1 RJBR Counter Carry bit.	R
0	C1RDR	Carry register 1 RDRP Counter Carry bit.	R

11.9.3.46 CAR2

CAR2 0x134B0134

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved												C2TJB	C2TFC	C2TCF	C2TOV	C2TUN	C2TFG	C2TBY	C2TPK	C2TMC	C2TBC	C2TPF	C2TDF	C2TED	C2TSC	C2TMA	C2TLC	C2TXC	C2TNC	C2TPH	C2TDP
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:20	Reserved	Writing has no effect, read as zero.	R
19	C2TJB	Carry register 2 TJBR Counter Carry bit.	R

18	C2TFC	Carry register 2 TFCS Counter Carry bit.	R
17	C2TCF	Carry register 2 TXCF Counter Carry bit.	R
16	C2TOV	Carry register 2 TOVR Counter Carry bit.	R
15	C2TUN	Carry register 2 TUND Counter Carry bit.	R
14	C2TFG	Carry register 2 TFRG Counter Carry bit.	R
13	C2TBY	Carry register 2 TBYT Counter Carry bit.	R
12	C2TPK	Carry register 2 TPKT Counter Carry bit.	R
11	C2TMC	Carry register 2 TMCA Counter Carry bit.	R
10	C2TBC	Carry register 2 TBCA Counter Carry bit.	R
9	C2TPF	Carry register 2 TXPF Counter Carry bit.	R
8	C2TDF	Carry register 2 TDFR Counter Carry bit.	R
7	C2TED	Carry register 2 TEDF Counter Carry bit.	R
6	C2TSC	Carry register 2 TSCL Counter Carry bit.	R
5	C2TMA	Carry register 2 TMCL Counter Carry bit.	R
4	C2TLC	Carry register 2 TLCL Counter Carry bit.	R
3	C2TXC	Carry register 2 TXCL Counter Carry bit.	R
2	C2TNC	Carry register 2 TNCL Counter Carry bit.	R
1	C2TPH	Carry register 2 TPFH Counter Carry bit.	R
0	C2TDP	Carry register 2 TDRP Counter Carry bit.	R

11.9.3.47 CAM1

CAM1 **0x134B0138**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	M164	M1127	M1255	M1511	M11K	M1MAX	M1MGV	Reserved								M1RBY	M1RPK	M1RFC	M1RMC	M1RBC	M1RXC	M1RXP	M1RXU	M1RAL	M1RFL	M1RCD	M1RCS	M1RUN	M1ROV	M1RFR	M1RJB	M1RDR
RST	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	RW
31	M164	Mask register 1 TR64 Counter Carry bit Mask.	RW
30	M1127	Mask register 1 TR127 Counter Carry bit Mask.	RW
29	M1255	Mask register 1 TR255 Counter Carry bit Mask.	RW
28	M1511	Mask register 1 TR511 Counter Carry bit Mask.	RW
27	M11k	Mask register 1 TR1K Counter Carry bit Mask.	RW
26	M1MAX	Mask register 1 TRMAX Counter Carry bit Mask.	RW
25	M1MGV	Mask register 1 TRMGV Counter Carry bit Mask.	RW
24:17	Reserved	Writing has no effect, read as zero.	R
16	M1RBY	Mask register 1 RBYT Counter Carry bit Mask.	RW

15	M1RPK	Mask register 1 RPKT Counter Carry bit Mask.	RW
14	M1RFC	Mask register 1 RFCS Counter Carry bit Mask.	RW
13	M1RMC	Mask register 1 RMCA Counter Carry bit Mask.	RW
12	M1RBC	Mask register 1 RBCA Counter Carry bit Mask.	RW
11	M1RXC	Mask register 1 RXCF Counter Carry bit Mask.	RW
10	M1RXP	Mask register 1 RXPF Counter Carry bit Mask.	RW
9	M1RXU	Mask register 1 RXUO Counter Carry bit Mask.	RW
8	M1RAL	Mask register 1 RALN Counter Carry bit Mask.	RW
7	M1RFL	Mask register 1 RFLR Counter Carry bit Mask.	RW
6	M1RCD	Mask register 1 RCDE Counter Carry bit Mask.	RW
5	M1RCS	Mask register 1 RCSE Counter Carry bit Mask.	RW
4	M1RUN	Mask register 1 RUND Counter Carry bit Mask.	RW
3	M1ROV	Mask register 1 ROVR Counter Carry bit Mask.	RW
2	M1RFR	Mask register 1 RFRG Counter Carry bit Mask.	RW
1	M1RJB	Mask register 1 RJBR Counter Carry bit Mask.	RW
0	M1RDR	Mask register 1 RDRP Counter Carry bit Mask.	RW

11.9.3.48 CAM2

CAM2 0x134B013C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved												M2TJB	M2TFC	M2TCF	M2TOV	M2TUN	M2TFG	M2TBY	M2TPK	M2TMC	M2TBC	M2TPF	M2TDF	M2TED	M2TSC	M2TMA	M2TLC	M2TXC	M2TNC	M2TPH	M2TDP
RST	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	RW
31:20	Reserved	Writing has no effect, read as zero.	R
19	M2TJB	Mask register 2 TJBR Counter Carry bit Mask.	RW
18	M2TFC	Mask register 2 TFCS Counter Carry bit Mask.	RW
17	M2TCF	Mask register 2 TXCF Counter Carry bit Mask.	RW
16	M2TOV	Mask register 2 TOVR Counter Carry bit Mask.	RW
15	M2TUN	Mask register 2 TUND Counter Carry bit Mask.	RW
14	M2TFG	Mask register 2 TFRG Counter Carry bit Mask.	RW
13	M2TBY	Mask register 2 TBYT Counter Carry bit Mask.	RW
12	M2TPK	Mask register 2 TPKT Counter Carry bit Mask.	RW
11	M2TMC	Mask register 2 TMCA Counter Carry bit Mask.	RW
10	M2TBC	Mask register 2 TBCA Counter Carry bit Mask.	RW
9	M2TPF	Mask register 2 TXPF Counter Carry bit Mask.	RW

8	M2TDF	Mask register 2 TDFR Counter Carry bit Mask.	RW
7	M2TED	Mask register 2 TEDF Counter Carry bit Mask.	RW
6	M2TSC	Mask register 2 TSCL Counter Carry bit Mask.	RW
5	M2TMA	Mask register 2 TMCL Counter Carry bit Mask.	RW
4	M2TLC	Mask register 2 TLCL Counter Carry bit Mask.	RW
3	M2TXC	Mask register 2 TXCL Counter Carry bit Mask.	RW
2	M2TNC	Mask register 2 TNCL Counter Carry bit Mask.	RW
1	M2TPH	Mask register 2 TPFH Counter Carry bit Mask.	RW
0	M2TDP	Mask register 2 TDRP Counter Carry bit Mask.	RW

12 EFUSE Slave Interface (EFUSE)

12.1 Overview

Total 256 bits of EFUSE are provided, separated into lower 128bits segment and higher 128bits segment.

Each segment can be programmed separately or together.
Each segment has a protect bit, which has higher priority than program segment selection.

Programming frequency should be with in 133Mhz and 166Mhz.
Programming time is around 3ms for either program in 128bits or 256bits.

Initial value of EFUSE is 0, when programmed to 1, it won't able to program back to 0 anymore.
Do not attempt to program any bit that already programmed to 1, such action will result unpredictable status to whole effuse block.

Programming voltage supply pin VDDQ:

- In program mode, supply VDDQ with 2.5V.
  Important: VDDQ pin should be kept 0v except during programming. Maximum accumulative time for VDDQ pin exposed under 2.5V+/-10% should be less than 1 sec.
- In read mode, leave VDDQ to 0V.

12.2 Register Description

Following table lists all the register definitions. All registers' 32bit addresses are physical addresses.

Table 12-1 EFUSE Register Description

Name	Description	RW	Reset Value	Address	Access Size
EFSCTL	EFUSE Control Register	RW	0x00	0x134100DC	32
EFUSE0	EFUSE Data 0 Register	RW	0x????????	0x134100E0	32
EFUSE1	EFUSE Data 1 Register	RW	0x????????	0x134100E4	32
EFUSE2	EFUSE Data 2 Register	RW	0x????????	0x134100E8	32
EFUSE3	EFUSE Data 3 Register	RW	0x????????	0x134100EC	32
EFUSE4	EFUSE Data 4 Register	RW	0x????????	0x134100F0	32
EFUSE5	EFUSE Data 5 Register	RW	0x????????	0x134100F4	32
EFUSE6	EFUSE Data 6 Register	RW	0x????????	0x134100F8	32
EFUSE7	EFUSE Data 7 Register	RW	0x????????	0x134100FC	32

12.2.1 EFUSE Control Register (EFSCTL)

EFSCTL 0x134100DC

Bit	7	6	5	4	3	2	1	0
	Reserved						HI_WEN	LO_WEN
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:2	Reserved	Writing has no effect, read as zero.	R
1	HI_WEN	High 128 bit write enable bit. *1 0: Disable 1: Enabled	RW
0	LO_WEN	Low 128 bit write enable bit. *1 0: Disable 1: Enabled	RW

NOTE:

*1: DATA[31]/PRTCT bits take higher priorities than HI_WEN & LOW_WEN.

12.2.2 EFUSE Data Register (EFUSEn)

EFUSEn (n=0,1,2,4,5,6) **0x134100E0~0x134100FC**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:0	DATA	EUFSE DATA register.	RW

EFUSE7 [31]	EFUSE7 [30]	~	EFUSE7 [1]	EFUSE7 [0]
EFUSE6 [31]	EFUSE6 [30]	~	EFUSE6 [1]	EFUSE6 [0]
EFUSE5 [31]	EFUSE5 [30]	~	EFUSE5 [1]	EFUSE5 [0]
EFUSE4 [31]	EFUSE4 [30]	~	EFUSE4 [1]	EFUSE4 [0]
EFUSE3 [31]	EFUSE3 [30]	~	EFUSE3 [1]	EFUSE3 [0]
EFUSE2 [31]	EFUSE2 [30]	~	EFUSE2 [1]	EFUSE2 [0]
EFUSE1 [31]	EFUSE1 [30]	~	EFUSE1 [1]	EFUSE1 [0]
EFUSE0 [31]	EFUSE0 [30]	~	EFUSE0 [1]	EFUSE0 [0]

EFUSEn (n=3,7) **0x134100E0~0x134100FC**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA[31]/PRTCT	DATA [0-30]																														
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	DATA[31]/PRTCT	EUFSE DATA bit 31 / Protect bit. *1 EFSUE3: When this bit programmed to 1, EFUSE0~3 lower 128 bits become un-programmable forever. EFUSE7: When this bit programmed to 1,EFUSE4~7 higher 128 bits become un-programmable forever.	RW
30:0	DATA [30:0]	EUFSE DATA register.	

EFUSE7 [30]	EFUSE7 [29]	~	EFUSE7 [1]	EFUSE7 [0]
EFUSE6 [30]	EFUSE6 [29]	~	EFUSE6 [1]	EFUSE6 [0]
EFUSE5 [30]	EFUSE5 [29]	~	EFUSE5 [1]	EFUSE5 [0]
EFUSE4	EFUSE4	~	EFUSE4 [1]	EFUSE4 [0]

		[30]	[29]				
		EFUSE3 [30]	EFUSE3 [29]	~	EFUSE3 [1]	EFUSE3 [0]	
		EFUSE2 [30]	EFUSE2 [29]	~	EFUSE2 [1]	EFUSE2 [0]	
		EFUSE1 [30]	EFUSE1 [29]	~	EFUSE1 [1]	EFUSE1 [0]	
		EFUSE0 [30]	EFUSE0 [29]	~	EFUSE0 [1]	EFUSE0 [0]	

NOTE:

*[1]: DATA[31]/PRTCT bits take higher priorities than HI_WEN & LOW_WEN.

12.3 Flow

12.3.1 Write EFUSE Flow

1 Write full-256 bit.
- a Config AHB2 freq to 166Mhz.
- b Connect VDDQ pin to 2.5V. *1
- c Write register EFUSE0~EFUSE7.
- d Write register EFSCTL.HI_WEN and EFSCTL.LO_WEN to 1 at same time.
- e Wait till register EFSCTL.HI_WEN and EFSCTL.LO_WEN is set to 0.
- f Disconnect VDDQ pin from 2.5V and finish writing EFUSE.

2 Write low-128 bit.
- a Config AHB2 freq to 166Mhz.
- b Connect VDDQ pin to 2.5V. *1
- c Write register EFUSE0~EFUSE3.
- d Write register EFSCTL.LO_WEN to 1.
- e Wait till register EFSCTL.LO_WEN is set to 0.
- f Disconnect VDDQ pin from 2.5V and finish writing EFUSE.

3 Write high-128 bit.
- a Config AHB2 freq to 166Mhz.
- b Connect VDDQ pin to 2.5V. *1
- c Write register EFUSE3~EFUSE7.
- d Write register EFSCTL.HI_WEN to 1.
- e Wait till register EFSCTL.HI_WEN is set to 0.
- f Disconnect VDDQ pin from 2.5V and finish writing EFUSE.

NOTE:

*1: Do NOT pre-charge VDDQ too early before fuse program started.
The maximum 2.5V supply time to VDDQ must be strictly controlled less than 1sec.
Programming window is below 3ms.
Use on-chip counter and GPIO to coordinate external supply source.

12.3.2 Read EFUSE Flow

Read register EFUSE0~EFUSE7 after chip is reset.