T8307 VoWLAN Processor

1 Description

The T8307 is a VoIP processor IC designed for handheld wireless terminals that operate over 802.11 *Wi-Fi*® wireless local area networks. It is one chip of a four-chip complete solution comprising Agere's VoWLAN product (see Figure 5.1-1 and Figure 5.1-2). The T8307 leverages Agere's leading wireless handset technology to provide ultra-small form factors, extended battery life, and low implementation costs. The T8307 includes a digital signal processor (DSP) core optimized for low-power communications applications and a powerful high-performance, industry-standard microcontroller core along with a rich set of peripherals specifically designed for wireless handsets.

The T8307 is targeted at 802.11/*Wi-Fi* wireless enterprise communications in a variety of industries where robust, private, nontariffed communications provide significant advantages. Several examples include health care, retail, education, manufacturing, and distribution. The T8307 achieves best-in-class signal processing performance while maintaining the efficient software code density, low power consumption, small physical size, and low cost required for broad deployment of handheld 802.11/*Wi-Fi* handsets.

2 Features

- *ARM*®946E-S microcontroller core:
  - 91 MHz system bus, 16 Kbyte instruction and 4 Kbyte data caches.
  - 8 Kbyte tightly coupled zero wait-state instruction and 4 Kbyte tightly coupled zero wait-state data memory.
  - Direct memory access (DMA) controller for transparent transfers between memory and peripherals.
  - External memory interface (EMI) with asynchronous burst mode support.
  - Programmable interrupt controller (PIC), programmable 48-bit general-purpose I/O unit, keyboard interface, programmable interval timer, and real-time clock (RTC).
  - Synchronous serial port (SSP) supporting *Motorola*® SPI, *Texas Instruments*® *SSI*™, *National Semiconductor*® *MICROWIRE*®, and *Philips*® I²S serial bus formats.
  - Multiple UARTs with automatic baud rate detection and configuration, modem flow control, and IrDA.
  - On-chip 12 Mbits/s USB 1.1 slave device controller hardware.
  - SD/MMC controller that supports interfacing to secure digital/multimedia memory cards.
- DSP16000 dual-MAC DSP core:
  - Up to 182 million MACs per second at 91 MHz.
  - Memory complement: 144K x 16 bit ROM, 24K x 16 bit RAM.
  - Vocoder support: G.711 ∝-law, G.711 A-law, G.723, G.729A, G.729B, and G.729AB.
  - Programmable interval timer unit, programmable 2-bit general-purpose I/O unit, parallel memory style interface to CSP8307 analog conversion and power management IC.
  - Synchronous serial port supporting SPI, synchronous serial interface (SSI), and PCM formats for easy interface to external audio processor chips.
- JTAG boundary-scan and integrated hardware development system (HDS).
- Low power:
  - Ultralow leakage process technology for best-in-class standby power.
  - Flexible power management modes to allow for maximum active power management.
  - 1.5 V internal supply for power efficiency.
  - 1.8 V I/O pin supplies for compatibility.
- Interprocessor communication hardware support between *ARM* and DSP16000:
  - Shared 512 x 32-bit memory.
  - Programmable interrupt and status.
- Two on-chip, programmable, PLL clock synthesizers: one for *ARM* and DSP, the other for USB.
- Supported by T8307 industry-standard *ARM* software and hardware development tools.
- Fine pitch ball grid array package for small footprint:
  - 224-ball FSBGAC (10 mm x 10 mm; 0.5 mm ball pitch).

Table of Contents

Contents Page

Table of Contents (continued)

Contents **Page**

Table of Contents (continued)

Contents **Page**

Table of Contents (continued)

Contents **Page**

Table of Contents (continued)

Contents **Page**

Table of Contents (continued)

Contents **Page**

Table of Contents (continued)

Contents **Page**

Table of Contents (continued)

Contents **Page**

Table of Contents (continued)

Contents **Page**

Table of Contents (continued)

Contents **Page**

List of Figures

Figure **Page**

List of Figures (continued)

Figure **Page**

List of Figures (continued)

Figure **Page**

List of Tables

Table — Page

List of Tables (continued)

Table **Page**

List of Tables (continued)

Table **Page**

List of Tables (continued)

Table **Page**

List of Tables (continued)

Table **Page**

List of Tables (continued)

Table **Page**

List of Tables (continued)

Table **Page**

List of Tables (continued)

Table **Page**

3 Notation Conventions

The following notation conventions apply to this data sheet.

lower-case	Registers that are directly writable or readable by DSP16000 core instructions (register-mapped registers) are lower-case.
UPPER-CASE	Device flags, I/O pins, control register fields, and memory-mapped registers are upper-case.
boldface	Register names and DSP16000 core instructions are printed in boldface when used in text descriptions.
italics	Documentation variables that are replaced are printed in italics.
`courier`	DSP16000 program examples or C-language representations are printed in courier font.
[ ]	Square brackets enclose a range of numbers that represents multiple bits in a single register or bus. The range of numbers is delimited by a colon. For example, **imux**[11:10] are bits 11 and 10 of the program-accessible **imux** register.
< >	Angle brackets enclose a list of items delimited by commas or a range of items delimited by a dash (—), one of which is selected if used in an instruction. For example, **SADD**<**0—3**> represents the four memory-mapped registers **SADD0**, **SADD1**, **SADD2**, and **SADD3**, and the general instruction **aTE**<**h,l**> **= RB** can be replaced with **a0h = timer**.

4 Pinout Information

4.1 224-Pin FSBGAC (Top See-Through)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		VSS	A_A10			A_A3		A_D0				A_D11		A_CS0N			A_BE1N	VSS	
	A_A12		A_A11	VDD_IO		VDD_IO	A_A0	VDD_CORE				VDD_IO	VDD_CORE	A_OEN		VDD_IO	A_CS7N		USB_SUSP
	VDD_IO	A_A9		A_A8	A_A5	A_A4		A_A1				A_D10		VDD_IO	A_CS1N	A_CS5N		USB_VMI	USB_VPI
		A_A19	A_A15		A_A14	A_A13	A_A6	A_A2	VDD_IO		A_D8	A_D13	A_D15	A_WEN	A_CS4N		USB_VPO	VDD_IO	
			A_A21	A_A20		A_A18	A_A7	A_A16	A_D1		A_D6	A_D9	A_D14	PIO30_ WAITN		USB_VMO	USB_DATA		
F	VDD_IO	A_A25	A_A24	A_A23	A_A22			A_A17	A_D2		A_D5	A_D12			A_CS6N	USB_OEN	VDDA_U	VSSA_U	X2RTC
		VDD_CORE		KEYBRD10	PWM1_PI O46	KEYBRD9		VSS	VSS	A_D3	A_D4	A_D7		A_CS2N	RESETN	OSC32OUT		VRTC	
	KEYBRD7	KEYBRD11	KEYBRD8	KEYBRD6		KEYBRD5_ PSW1_BUF		VSS	VSS	VSS	VSS	VSS		FLASH- RSTN		A_BE0N	RTC- ALARMN	VDD_CORE	X1RTC
J				KEYBRD3	KEYBRD2	KEYBRD1	KEYBRD0	KEYBRD4				VSS	MCI_CMD	A_CS3N	MCI_CLK	PIO47			
							TX1	VSS				ATMS_ PIO45	MCI_DAT0						
L				VDD_IO	ATDI_ RTS1	RX1	VSS	VSS				MCI_DAT1	MCI_DAT2	MCI_CMD_ EN	MCI_DAT3	VDD_IO			
	RX0	VDD_CORE	TX0	ATCK_ CTS1		CTS0		VSS	VSS	VSS	VSS	VSS		MCI_DAT_ EN		IRQ1	SPRXD1	SPCLK1	IRQ2
		RTS0		DTR0	DCD0	D_A7		D_D12	D_D13	D_D14	IOBIT1	CKO_IACK		MCI_DAT0_ EN	SPFS1	SPTXD1		VDD_CORE	
	VDD_IO	DSR0	RI0	SIMCLK	SIMRST			D_D10	D_D11		D_D15	IRQ6			PWRKEEP	IRQ5	IRQ3	IRQ4	SYSCLK- REQ
			SPCLK0	SPRXD0		D_A8	D_D2	D_D4	D_D9		VSSA_D	TRSTN	PIO32	PIO34		IRDATX	IRDARX		
T		SIMIO	SPFS0		D_A5	D_D1	D_D3	D_D7	D_D8		VDDA_D	TDO	ATDO_ PWM2	TEST2	INT0		CPTST- STOP_CKO	VDD_IO	
	VDD_IO	SPTXD0		D_A4	VDD_IO	D_D0		D_D6				VDD_IO		TCK	TEST1	PIO26		PIO33	PIO27
	D_A0		D_A3	D_A6		RWN	VDD_CORE	D_D5				IOBIT0	VDD_CORE	TDI		TEST3	PIO10		PIO25
		D_A1	D_A2			IO		VDD_IO				CKI		TMS			PIO28	VDD_IO	

4 Pinout Information (continued)

4.2 224-Pin FSBGAC Pin Information

Table 4.2-1 T8307 Pinout

#	Function	MUX Control	Direction (Default_ Alternate)	Pull-Up/ Pull-Down (200 k.)	Ball	Reset Value
Call Processor						
External Memory Interface (SMC)						
1	A_A0	—	Out	—	B7	0*
2	A_A1	—	Out	—	C8	0
3	A_A2	—	Out	—	D8	0
4	A_A3	—	Out	—	A6	0
5	A_A4	—	Out	—	C6	0
6	A_A5	—	Out	—	C5	0
7	A_A6	—	Out	—	D7	0
8	A_A7	—	Out	—	E7	0
9	A_A8	—	Out	—	C4	0
10	A_A9	—	Out	—	C2	0
11	A_A10	—	Out	—	A3	0
12	A_A11	—	Out	—	B3	0
13	A_A12	—	Out	—	B1	0
14	A_A13	—	Out	—	D6	0
15	A_A14	—	Out	—	D5	0
16	A_A15	—	Out	—	D3	0
17	A_A16	—	Out	—	E8	0
18	A_A17	—	Out	—	F8	0
19	A_A18	—	Out	—	E6	0
20	A_A19	—	Out	—	D2	0
21	A_A20	—	Out	—	E4	0
22	A_A21	—	Out	—	E3	0
23	A_A22	—	Out	—	F5	0
24	A_A23	—	Out	—	F4	0
25	A_A24	—	Out	—	F3	0
26	PIO35_A_A25_BOOTSEL	ALTPINC[9]	In/Out_Out	Pull-up	F2	Input
27	A_D0	—	In/Out	—	A8	Input
28	A_D1	—	In/Out	—	E9	Input
29	A_D2	—	In/Out	—	F9	Input
30	A_D3	—	In/Out	—	G10	Input
31	A_D4	—	In/Out	—	G11	Input
32	A_D5	—	In/Out	—	F11	Input

* For A_A0, A_A1, A_A2, . . . A_A24, the reset value is not valid until after RESETN is deasserted.
† FLASHRSTN is 0 when RESETN is asserted and becomes 1 when RESETN is deasserted.
‡ PIO00_IRQ5 and PIO31_IRQ6 are always connected internally to *ARM* programmable interrupt controller (PIC) interrupt request inputs.

4 Pinout Information (continued)

4.2 224-Pin FSBGAC Pin Information (continued)

Table 4.2-1 T8307 Pinout (continued)

#	Function	MUX Control	Direction (Default_ Alternate)	Pull-Up/ Pull-Down (200 k.)	Ball	Reset Value
Call Processor (continued)						
External Memory Interface (SMC) (continued)						
33	A_D6	—	In/Out	—	E11	Input
34	A_D7	—	In/Out	—	G12	Input
35	A_D8	—	In/Out	—	D11	Input
36	A_D9	—	In/Out	—	E12	Input
37	A_D10	—	In/Out	—	C12	Input
38	A_D11	—	In/Out	—	A12	Input
39	A_D12	—	In/Out	—	F12	Input
40	A_D13	—	In/Out	—	D12	Input
41	A_D14	—	In/Out	—	E13	Input
42	A_D15	—	In/Out	—	D13	Input
43	A_WEN	—	Out	—	D14	1
44	A_OEN	—	Out	—	B14	1
45	PIO30_WAITN	ALTPINC[7]	In/Out_In	Pull-up	E14	Input
46	FLASHRSTN	—	Out	—	H14	1†
47	A_CS0N	—	Out	—	A14	1
48	A_CS1N	—	Out	—	C15	1
49	A_CS2N	—	Out	—	G14	1
50	A_CS3N	—	Out	—	J14	1
51	A_CS4N	—	Out	—	D15	1
52	A_CS5N	—	Out	—	C16	1
53	A_CS6N	—	Out	—	F15	1
54	A_CS7N	—	Out	—	B17	1
55	A_BE0N	—	Out	—	H16	1
56	A_BE1N	—	Out	—	A17	1
SIM Card Interface						
57	SIMCLK	—	Out	—	P4	0
58	SIMIO	—	In/Out	Pull-up	T2	0
59	PIO14 (SIMRST)	—	In/Out	Pull-up	P5	Input

* For A_A0, A_A1, A_A2, . . . A_A24, the reset value is not valid until after RESETN is deasserted.
† FLASHRSTN is 0 when RESETN is asserted and becomes 1 when RESETN is deasserted.
‡ PIO00_IRQ5 and PIO31_IRQ6 are always connected internally to *ARM* programmable interrupt controller (PIC) interrupt request inputs.

4 Pinout Information (continued)

4.2 224-Pin FSBGAC Pin Information (continued)

Table 4.2-1 T8307 Pinout (continued)

#	Function	MUX Control	Direction (Default_ Alternate)	Pull-Up/ Pull-Down (200 k.)	Ball	Reset Value
Call Processor (continued)						
USB						
60	PIO07_USB_SUSP	ALTPINC[2]	In/Out_Out	Pull-up	B19	Input
61	PIO09_USB_VPI	ALTPINC[12]	In/Out_In	Pull-up	C19	Input
62	PIO11_USB_VMI	ALTPINC[12]	In/Out_In	Pull-up	C18	Input
63	PIO12_USB_VPO	ALTPINC[2]	In/Out_In/Out	Pull-up	D17	Input
64	PIO13_USB_VMO	ALTPINC[2]	In/Out_In/Out	Pull-up	E16	Input
65	PIO36_USB_OEN	ALTPINC[2]	In/Out_Out	Pull-up	F16	Input
66	PIO37_USB_DATA	ALTPINC[2]	In/Out_In	Pull-up	E17	Input
CP-Side Synchronous Serial Port/I²S Interface (SSP0)						
67	SPCLK0	—	In/Out	Pull-up	R3	0
68	SPRXD0	—	In/Out	Pull-up	R4	Input
69	SPTXD0_I2SD	—	In/Out	Pull-up	U2	Input
70	SPFS0	—	In/Out	Pull-up	T3	1
UART0 (Full-Feature)						
71	RI0_PIO01	ALTPINC[0]	Out_In/Out	Pull-up	P3	Z-State
72	DSR0_PIO02	ALTPINC[0]	Out_In/Out	Pull-up	P2	Z-State
73	DTR0_PIO03	ALTPINC[0]	In_In/Out	Pull-up	N4	Input
74	RTS0_PIO04	ALTPINC[0]	In_In/Out	Pull-up	N2	Input
75	TX0	—	Out	Pull-up	M3	Z-State
76	RX0_IRQ28	—	In	Pull-up	M1	Input
77	CTS0_PIO29	ALTPINC[0]	In/Out_In/Out	Pull-up	M6	Input
78	DCD0_PIO44	ALTPINC[0]	In/Out_In/Out	Pull-up	N5	Input
IrDA						
79	PIO41_IRDATX	ALTPINC[6]	In/Out_Out	Pull-up	R16	Input
80	PIO42_IRDARX	ALTPINC[6]	In/Out_In	Pull-up	R17	Input
UART1						
81	TX1	—	Out	Pull-up	K7	1
82	RX1_IRQ28	—	In	Pull-up	L6	Input
	For UART1 H/W flow control signals, See *ARM* JTAG ATDI and ATCK pins.	—	—	—	—	—

* For A_A0, A_A1, A_A2, . . . A_A24, the reset value is not valid until after RESETN is deasserted.
† FLASHRSTN is 0 when RESETN is asserted and becomes 1 when RESETN is deasserted.
‡ PIO00_IRQ5 and PIO31_IRQ6 are always connected internally to *ARM* programmable interrupt controller (PIC) interrupt request inputs.

4 Pinout Information (continued)

4.2 224-Pin FSBGAC Pin Information (continued)

Table 4.2-1 T8307 Pinout (continued)

#	Function	MUX Control	Direction (Default_ Alternate)	Pull-Up/ Pull-Down (200 k.)	Ball	Reset Value
Call Processor (continued)						
Keyboard Matrix						
83	KEYBRD0	—	In/Out	Pull-up	J7	Input
84	KEYBRD1	—	In/Out	Pull-up	J6	Input
85	KEYBRD2	—	In/Out	Pull-up	J5	Input
86	KEYBRD3	—	In/Out	Pull-up	J4	Input
87	KEYBRD4	—	In/Out	Pull-up	J8	Input
88	KEYBRD5(PSW1_BUF)	—	In/Out	Pull-up	H6	Input
89	KEYBRD6	—	In/Out	Pull-up	H4	Input
90	KEYBRD7	—	In/Out	Pull-up	H1	Input
91	KEYBRD8	—	In/Out	Pull-up	H3	Input
92	KEYBRD9	—	In/Out	Pull-up	G6	Input
93	KEYBRD10	—	In/Out	Pull-up	G4	Input
94	KEYBRD11	—	In/Out	Pull-up	H2	Input
SD/MMC Card Interface						
95	PIO38_MCI_CLK	ALTPINC[3]	In/Out_Out	Pull-up	J15	Input
96	PIO08_MCI_CMD	ALTPINC[3]	In/Out_In/Out	Pull-up	J13	Input
97	PIO21_MCI_DAT0	ALTPINC[3]	In/Out_In/Out	Pull-up	K13	Input
98	PIO22_MCI_DAT1	ALTPINC[3]	In/Out_In/Out	Pull-up	L12	Input
99	PIO23_MCI_DAT2	ALTPINC[3]	In/Out_In/Out	Pull-up	L13	Input
100	PIO24_MCI_DAT3	ALTPINC[3]	In/Out_In/Out	Pull-up	L15	Input
101	PIO39_MCI_CMD_EN	ALTPINC[3]	In/Out_Out	Pull-up	L14	Input
102	PIO40_MCI_DAT_EN	ALTPINC[3]	In/Out_Out	Pull-up	M14	Input
103	PIO43_MCI_DAT0_EN	ALTPINC[3]	In/Out_Out	Pull-up	N14	Input
RTC (32 kHz)						
104	X1RTC	—	Special (V_{RTC})	—	H19	—
105	X2RTC	—	Special (V_{RTC})	—	F19	—
106	RTCALARMN	—	Out (V_{RTC})	—	H17	Undefined
107	OSC32OUT	—	Out (1.8 V)	—	G16	Undefined
Analog Baseband Control Pins						
108	PIO19 (PWRKEEP)	—	In/Out	Pull-up	P15	Input
109	PIO20_SYSCLKREQ	ALTPINC[5]	In/Out_Out	Pull-up	P19	Input
Pulse-Width Modulated Port						
110	PWM1_PIO46	ALTPINC[10]	Out_In/Out	Pull-up	G5	0

* For A_A0, A_A1, A_A2, . . . A_A24, the reset value is not valid until after RESETN is deasserted.
† FLASHRSTN is 0 when RESETN is asserted and becomes 1 when RESETN is deasserted.
‡ PIO00_IRQ5 and PIO31_IRQ6 are always connected internally to *ARM* programmable interrupt controller (PIC) interrupt request inputs.

4 Pinout Information (continued)

4.2 224-Pin FSBGAC Pin Information (continued)

Table 4.2-1 T8307 Pinout (continued)

#	Function	MUX Control	Direction (Default_ Alternate)	Pull-Up/ Pull-Down (200 k.)	Ball	Reset Value
Call Processor (continued)						
General-Purpose Input/Output Pins						
111	PIO10	—	In/Out	Pull-up	V17	Input
112	PIO25	—	In/Out	Pull-up	V19	Input
113	PIO26	—	In/Out	Pull-up	U16	Input
114	PIO27	—	In/Out	Pull-up	U19	Input
115	PIO28	—	In/Out	Pull-up	W17	Input
116	PIO32	—	In/Out	Pull-up	R13	Input
117	PIO33	—	In/Out	Pull-up	U18	Input
118	PIO34	—	In/Out	Pull-up	R14	Input
119	PIO47	—	In/Out	Pull-up	J16	Input
Interrupt Pins						
120	IRQ1	—	In	Pull-up	M16	Input
121	IRQ2	—	In	Pull-up	M19	Input
122	IRQ3	—	In	Pull-up	P17	Input
123	IRQ4	—	In	Pull-up	P18	Input
124	PIO00_IRQ5‡	—	In/Out	Pull-up	P16	Input
125	PIO31_IRQ6‡	—	In/Out	Pull-up	P12	Input
***ARM* JTAG Debug Pins**						
126	ATMS_PIO45	TEST1—3	In_In/Out	Pull-up	K12	Input
127	ATCK_CTS1	TEST1—3	In_In/Out	Pull-up	M4	Input
128	ATDI_RTS1	TEST1—3	In_In/Out	Pull-up	L5	Input
129	ATDO_PWM2	TEST1—3	Out_Out	—	T13	Unknown
Test Pins						
130	CPTSTSTOP_CKO	ALTPINC[11]	In_Out	Pull-down	T17	Input
131	TEST1	—	In	Pull-up	U15	Input
132	TEST2	—	In	Pull-up	T14	Input
133	TEST3	—	In	Pull-up	V16	Input

* For A_A0, A_A1, A_A2, . . . A_A24, the reset value is not valid until after RESETN is deasserted.
† FLASHRSTN is 0 when RESETN is asserted and becomes 1 when RESETN is deasserted.
‡ PIO00_IRQ5 and PIO31_IRQ6 are always connected internally to *ARM* programmable interrupt controller (PIC) interrupt request inputs.

4 Pinout Information (continued)

4.2 224-Pin FSBGAC Pin Information (continued)

Table 4.2-1 T8307 Pinout (continued)

#	Function	MUX Control	Direction (Default_ Alternate)	Pull-Up/ Pull-Down (200 k.)	Ball	Reset Value
Digital Signal Processor (continued)						
CSP Interface						
134	D_A0	—	Out	—	V1	0
135	D_A1	—	Out	—	W2	0
136	D_A2	—	Out	—	W3	0
137	D_A3	—	Out	—	V3	0
138	D_A4	—	Out	—	U4	0
139	D_A5	—	Out	—	T5	0
140	D_A6	—	Out	—	V4	0
141	D_A7	—	Out	—	N6	0
142	D_A8	—	Out	—	R6	0
143	D_D0	—	In/Out	—	U6	Z-state
144	D_D1	—	In/Out	—	T6	Z-state
145	D_D2	—	In/Out	—	R7	Z-state
146	D_D3	—	In/Out	—	T7	Z-state
147	D_D4	—	In/Out	—	R8	Z-state
148	D_D5	—	In/Out	—	V8	Z-state
149	D_D6	—	In/Out	—	U8	Z-state
150	D_D7	—	In/Out	—	T8	Z-state
151	D_D8	—	In/Out	—	T9	Z-state
152	D_D9	—	In/Out	—	R9	Z-state
153	D_D10	—	In/Out	—	P8	Z-state
154	D_D11	—	In/Out	—	P9	Z-state
155	D_D12	—	In/Out	—	N8	Z-state
156	D_D13	—	In/Out	—	N9	Z-state
157	D_D14	—	In/Out	—	N10	Z-state
158	D_D15	—	In/Out	—	P11	Z-state
159	RWN	—	Out	—	V6	1
160	IO	—	Out	—	W6	1
161	INT0	—	In	—	T15	Input
DSP-Side Synchronous Serial Port/I²S Interface (SSP1)						
162	SPCLK1_PIO18	ALTPINC[4]	In/Out_In/Out	Pull-up	M18	0
163	SPRXD1_PIO17	ALTPINC[4]	In/Out_In/Out	Pull-up	M17	Input
164	SPTXD1_I2SD_PIO16	ALTPINC[4]	In/Out_In/Out	Pull-up	N16	Input
165	SPFS1_PIO15	ALTPINC[4]	In/Out_In/Out	Pull-up	N15	1

* For A_A0, A_A1, A_A2, . . . A_A24, the reset value is not valid until after RESETN is deasserted.
† FLASHRSTN is 0 when RESETN is asserted and becomes 1 when RESETN is deasserted.
‡ PIO00_IRQ5 and PIO31_IRQ6 are always connected internally to *ARM* programmable interrupt controller (PIC) interrupt request inputs.

4 Pinout Information (continued)

4.2 224-Pin FSBGAC Pin Information (continued)

Table 4.2-1 T8307 Pinout (continued)

#	Function	MUX Control	Direction (Default_ Alternate)	Pull-Up/ Pull-Down (200 k.)	Ball	Reset Value
Digital Signal Processor (continued)						
CSP Interface (continued)						
Test Pins						
166	IOBIT0_PIO05	ALTPINC[1]	In/Out_In/Out	Pull-up	V12	Input
167	IOBIT1_PIO06	ALTPINC[1]	In/Out_In/Out	Pull-up	N11	Input
168	CKO_IACK	—	Out	—	N12	CLK
DSP JTAG and Chip Boundary-Scan Pins						
169	TCK	—	In	Pull-up	U14	Input
170	TDI	—	In	Pull-up	V14	Input
171	TDO	—	Out	—	T12	Unknown
172	TRSTN	—	In	Pull-up	R12	Input
173	TMS	—	In	Pull-up	W14	Input
Common Functions						
174	RESETN	—	In	—	G15	Input
175	CKI	—	Special (VDDA_D)	—	W12	—
Power Pins						
176	VRTC	—	1.5 V PWR	—	G18	—
177	VDD_IO_1P8	—	1.8 V PWR	—	B4	—
178	VDD_IO_1P8	—	1.8 V PWR	—	B6	—
179	VDD_IO_1P8	—	1.8 V PWR	—	B12	—
180	VDD_IO_1P8	—	1.8 V PWR	—	B16	—
181	VDD_IO_1P8	—	1.8 V PWR	—	C1	—
182	VDD_IO_1P8	—	1.8 V PWR	—	C14	—
183	VDD_IO_1P8	—	1.8 V PWR	—	D9	—
184	VDD_IO_1P8	—	1.8 V PWR	—	D18	—
185	VDD_IO_1P8	—	1.8 V PWR	—	F1	—
186	VDD_IO_1P8	—	1.8 V PWR	—	L4	—
187	VDD_IO_1P8	—	1.8 V PWR	—	L16	—
188	VDD_IO_1P8	—	1.8 V PWR	—	P1	—
189	VDD_IO_1P8	—	1.8 V PWR	—	T18	—
190	VDD_IO_1P8	—	1.8 V PWR	—	U1	—
191	VDD_IO_1P8	—	1.8 V PWR	—	U5	—
192	VDD_IO_1P8	—	1.8 V PWR	—	U12	—
193	VDD_IO_1P8	—	1.8 V PWR	—	W8	—
194	VDD_IO_1P8	—	1.8 V PWR	—	W18	—

* For A_A0, A_A1, A_A2, . . . A_A24, the reset value is not valid until after RESETN is deasserted.
† FLASHRSTN is 0 when RESETN is asserted and becomes 1 when RESETN is deasserted.
‡ PIO00_IRQ5 and PIO31_IRQ6 are always connected internally to *ARM* programmable interrupt controller (PIC) interrupt request inputs.

4 Pinout Information (continued)

4.2 224-Pin FSBGAC Pin Information (continued)

Table 4.2-1 T8307 Pinout (continued)

#	Function	MUX Control	Direction (Default_ Alternate)	Pull-Up/ Pull-Down (200 k.)	Ball	Reset Value
Digital Signal Processor (continued)						
Power Pins (continued)						
195	VDD_CORE	—	1.5 V PWR	—	B8	—
196	VDD_CORE	—	1.5 V PWR	—	B13	—
197	VDD_CORE	—	1.5 V PWR	—	G2	—
198	VDD_CORE	—	1.5 V PWR	—	H18	—
199	VDD_CORE	—	1.5 V PWR	—	M2	—
200	VDD_CORE	—	1.5 V PWR	—	N18	—
201	VDD_CORE	—	1.5 V PWR	—	V7	—
202	VDD_CORE	—	1.5 V PWR	—	V13	—
203	VSS	—	GND	—	A2	—
204	VSS	—	GND	—	A18	—
205	VSS	—	GND	—	G8	—
206	VSS	—	GND	—	G9	—
207	VSS	—	GND	—	H8	—
208	VSS	—	GND	—	H9	—
209	VSS	—	GND	—	H10	—
210	VSS	—	GND	—	H11	—
211	VSS	—	GND	—	H12	—
212	VSS	—	GND	—	J12	—
213	VSS	—	GND	—	K8	—
214	VSS	—	GND	—	L7	—
215	VSS	—	GND	—	L8	—
216	VSS	—	GND	—	M8	—
217	VSS	—	GND	—	M9	—
218	VSS	—	GND	—	M10	—
219	VSS	—	GND	—	M11	—
220	VSS	—	GND	—	M12	—
Analog Power						
221	VDDA_D	—	1.5 V PWR	—	T11	—
222	VSSA_D	—	GND	—	R11	—
223	VDDA_U	—	1.5 V PWR	—	F17	—
224	VSSA_U	—	GND	—	F18	—

* For A_A0, A_A1, A_A2, . . . A_A24, the reset value is not valid until after RESETN is deasserted.
† FLASHRSTN is 0 when RESETN is asserted and becomes 1 when RESETN is deasserted.
‡ PIO00_IRQ5 and PIO31_IRQ6 are always connected internally to *ARM* programmable interrupt controller (PIC) interrupt request inputs.

4 Pinout Information (continued)

4.3 Signal Description

Table 4.3-1 T8307 Signal Description

Signal	Type	Pin Description
Call Processor		
External Memory Interface (SMC)		
A_A0—A_A25	Out	*ARM* address lines.
A_D0—A_D15	In/Out	*ARM* data lines.
BOOTSEL	In	*ARM* boot location selection. This input is sampled upon external pin reset. If the sampled input is low, the core will boot from external ROM. Otherwise, the core will boot from internal ROM.
A_WEN	Out	*ARM* write enable (active-low).
A_OEN	Out	*ARM* output enable (active-low).
PIO30_WAITN	In	*ARM* wait-state request (active-low).
FLASHRSTN	Out	Reset output. Can be used to reset external memory chips and CSP8307 analog conversion IC.
A_CS0N—A_CS7N	Out	*ARM* external chip select outputs (active-low).
A_BE0N	Out	*ARM* byte lane enable 0 (active-low).
A_BE1N	Out	*ARM* byte lane enable 1 (active-low).
SIM Card Interface		
SIMCLK	Out	SIM clock output.
SIMIO	In/Out	SIM data input/output.
PIO14 (SIMRST)	Out	General-purpose I/O pin 14. Its recommended usage is SIM card reset output.
USB		
USB_SUSP	Out	USB suspension signal output (active-low).
USB_VPI	In	USB V+ input signal from external USB transceiver.
USB_VMI	In	USB V– input signal from external USB transceiver.
USB_VPO	In/Out	For single-ended type transceiver: bi zero. For differential type transceiver: USB V+ output signal. For bidirectional differential type transceiver: USB V+ input and output signal.
USB_VMO	In/Out	For single-ended type transceiver: USB V+/V- voltage output signal. For differential type transceiver: USB V– output signal. For bidirectional differential type transceiver: USB V– input and output signal.
USB_OEN	Out	USB output buffer enable (active-low).
USB_DATA	In	USB data input from external USB transceiver.
CP-Side Synchronous Serial Port/I^{2}S Interface (SSP0)		
SPCLK0	In/Out	SSP serial clock input (in slave mode) or output (in master mode).
SPRXD0	In/Out	SSP serial data input.
SPTXD0_I2SD	In/Out	SSP serial data output (or I^{2}S serial data input/output).
SPFS0	In/Out	SSP serial frame input (in slave mode) or output (in master mode).

4 Pinout Information (continued)

4.3 Signal Description (continued)

Table 4.3-1 T8307 Signal Description (continued)

Signal	Type	Pin Description
UART0 (Full Feature) and IrDA		
RI0	Out	UART0 ring indication modem status output.
DSR0	Out	UART0 data set ready modem status output.
DTR0	In	UART0 data terminal ready modem status input.
RTS0	In	UART0 request-to-send modem status input.
TX0	Out	UART0 transmitted serial data output.
RX0_IRQ28	In	UART0 received serial data input with Rx line wake-up interrupt capability. Note that IRQ28 is only for Rx line wake-up purpose. It is not a general interrupt request input.
CTS0	Out	UART0 clear-to-send modem status output.
DCD0	Out	UART0 data carrier detection modem status output.
IRDATX	Out	UART0 IrDA transmitted serial data output.
IRDARX	In	UART0 IrDA received serial data input.
UART1		
TX1	Out	UART1 transmitted serial data output.
RX1_IRQ28	In	UART1 received serial data input with Rx line wake-up interrupt capability. Note that IRQ28 is only for Rx line wake-up purpose. It is not a general interrupt request input.
RTS1	In	UART1 request-to-send modem status input.
CTS1	Out	UART1 clear-to-send modem status output.
Keyboard Matrix		
KEYBRD0—KEYBRD11	In/Out	Keyboard interface pins.
KEYBRD5 (PSW1_BUF)	In/Out	Keyboard interface pin 5. Its recommended usage is PSW1_BUF input, which accepts buffered power switch signal from CSP8307 analog conversion IC.
SD/MMC Card Interface Pins		
MCI_CLK	Out	SD/MMC card clock output.
MCI_CMD	In/Out	SD/MMC card command output/response input.
MCI_DAT0—MCI_DAT3	In/Out	SD/MMC card data input/output lines.
MCI_CMD_EN	Out	SD/MMC card command output enable (active-low).
MCI_DAT_EN	Out	Secure digital card data lines [3:1] enable (active-low).
MCI_DAT0_EN	Out	SD/MMC card data line 0 enable (active-low).
Real-Time Clock (32 kHz)		
X1RTC	—	External 32 kHz crystal connector 1 (crystal mode) or 32 kHz real-time clock input (bypass mode).
X2RTC	—	External 32 kHz crystal connector 2 (crystal mode). In bypass mode, this pin will be grounded.
RTCALARMN	Out	Power-on request to the power management section of CSP8307 analog baseband IC (active-low). This pin operates from V_{RTC} power supply (1.5 V).
OSC32OUT	Out	32 kHz real-time clock output. This pin operates from V_{DD}_IO_1P8 power supply (1.8 V).

4 Pinout Information (continued)

4.3 Signal Description (continued)

Table 4.3-1 T8307 Signal Description (continued)

<table>
<tr><th>Signal</th><th>Type</th><th>Pin Description</th></tr>
<tr><td colspan="3">Analog Baseband Control Pins</td></tr>
<tr><td>PIO19 (PWRKEEP)</td><td>Out</td><td>General-purpose input/output pin 19. Its recommended usage is PWR-KEEP output (which connects to CSP8307 analog conversion IC and, when asserted high, keeps the whole system in power-on state).</td></tr>
<tr><td>SYSCLKREQ</td><td>Out</td><td>System clock request output (active-low). This pin is intended to connect to the XOENAQ pin of CSP8307 analog conversion IC.</td></tr>
<tr><td colspan="3">Pulse-Width Modulation Ports</td></tr>
<tr><td>PWM1—PWM2</td><td>Out</td><td>Pulse-width modulated signal outputs.</td></tr>
<tr><td colspan="3">General-Purpose Input/Output Pins</td></tr>
<tr><td>PIO00—PIO47</td><td>In/Out</td><td>General-purpose input/output pins.</td></tr>
<tr><td colspan="3">Interrupt Pins</td></tr>
<tr><td>IRQ1—IRQ6</td><td>In</td><td>Interrupt request inputs.</td></tr>
<tr><td colspan="3">ARM JTAG Debug Pins</td></tr>
<tr><td>ATMS</td><td>In</td><td>ARM JTAG mode selection input.</td></tr>
<tr><td>ATCK</td><td>In</td><td>ARM JTAG clock input.</td></tr>
<tr><td>ATDI</td><td>In</td><td>ARM JTAG data input.</td></tr>
<tr><td>ATDO</td><td>Out</td><td>ARM JTAG data output.</td></tr>
<tr><td colspan="3">Test Pins</td></tr>
<tr><td>CPTSTSTOP</td><td>In</td><td>CP block test stop input.
1: ARM system bus clock and peripheral bus clock are stopped.
0: ARM system bus clock and peripheral bus clock are enabled.</td></tr>
<tr><td>CKO</td><td>Out</td><td>CP block clock output for test purpose. When enabled through ALT-PINC[11], this pin outputs either 1/2 of the ARM system bus clock frequency, or 48 MHz USB module clock (depending on CKOEN and UPLL2CKO bit settings). Note that this CKO output is strictly for test purpose only. To avoid excessive RF interference, the user should not enable the CKO output in the application circuit.</td></tr>
<tr><td>TEST1—TEST3</td><td>In</td><td>
<table>
<tr><th>TEST1</th><th>TEST2</th><th>TEST3</th><th>Description</th></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Normal mode (ARM JTAG pins selected; PIO45/PWM2/CTS1/RTS1 disabled).</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Reserved.</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Reserved.</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Reserved.</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>PIO45/PWM2/CTS1/RTS1 selected;
ARM and DSP JTAGs daisy chained (accessible through DSP JTAG pins).</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>PIO45/PWM2/CTS1/RTS1 selected;
ARM JTAG disabled.</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Reserved.</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Reserved.</td></tr>
</table>
</td></tr>
</table>

4 Pinout Information (continued)

4.3 Signal Description (continued)

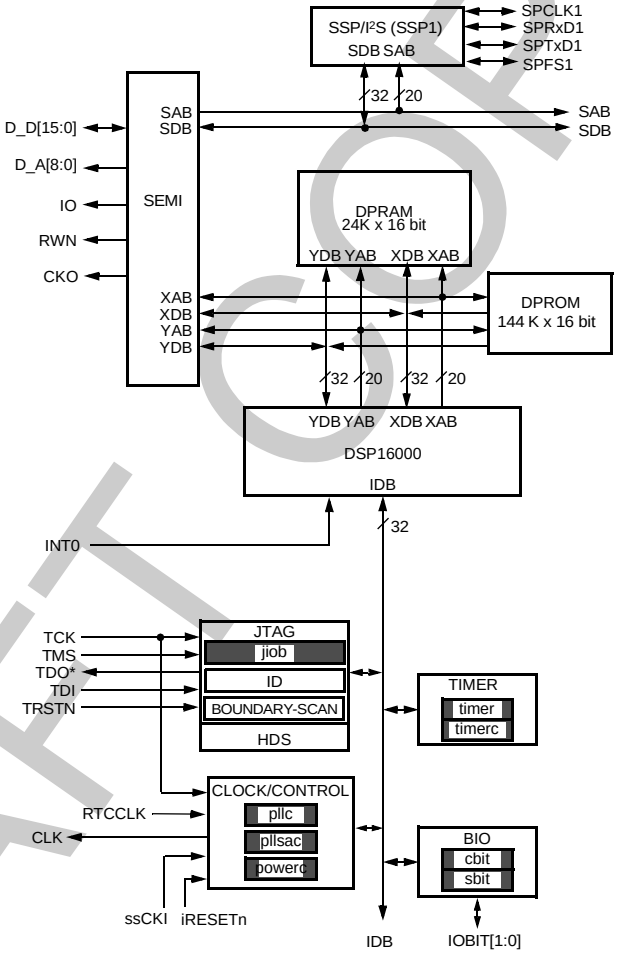
Table 4.3-1 T8307 Signal Description (continued)

Signal	Type	Pin Description
Digital Signal Processor		
CSP Interface		
D_A0—D_A8	Out	DSP address lines.
D_D0—D_D15	In/Out	DSP data lines.
RWN	Out	DSP external read/write signal output (low for write and high for read).
IO	Out	DSP EIO component enable output (active-low).
INT0	In	DSP external interrupt 0 input.
DSP-Side Synchronous Serial Port/I²S Interface (SSP1)		
SPCLK1	In/Out	SSP serial clock input (in slave mode) or output (in master mode).
SPRXD1	In/Out	SSP serial data input.
SPTXD1_I2SD	In/Out	SSP serial data output (or I²S serial data input/output).
SPFS1	In/Out	SSP serial frame input (in slave mode) or output (in master mode).
Test Pins		
IOBIT0—IOBIT1	In/Out	DSP general-purpose bit input/output lines.
CKO_IACK	Out	DSP clock output or interrupt acknowledgement output for test purpose.
DSP JTAG and Chip Boundary-Scan Pins		
TCK	In	JTAG clock input.
TDI	In	JTAG data input.
TDO	Out	JTAG data output.
TRSTN	In	JTAG reset input (active-low).
TMS	In	JTAG mode selection input.
Common Functions		
RESETN	In	Chip reset input (active-low).
CKI	—	Small-signal clock input.
Power Pins		
V_{RTC}	—	1.5 V power supply to RTC module.
V_{DD}_IO_1P8	—	1.8 V digital power supply for I/O ring.
V_{DD}_CORE	—	1.5 V digital power supply for core logic.
V_{SS}	—	Digital common ground.
V_{DD}A_D	—	1.5 V analog power supply for *ARM* and DSP PLL (ADPLL), as well as small signal clock buffer.
V_{SS}A_D	—	1.5 V analog ground for *ARM* and DSP PLL (ADPLL), as well as small signal clock buffer.
V_{DD}A_U	—	1.5 V analog power supply for USB PLL (UPLL).
V_{SS}A_U	—	1.5 V analog ground for USB PLL (UPLL).

5 Hardware Architecture

T8307 is composed of three major blocks: a DSP block using the DSP16000 core and peripherals, a CP block using the *ARM*946E-S core and peripherals, and an ICP/IDP block for communication between the DSP and the CP. Figure 5.1-1 and Figure 5.1-2 show the system block diagrams of the IC.

5.1 Device Architecture



* Internal daisy chain can be enabled through TEST1-3 pins.

Figure 5.1-1 T8307 Block Diagram—Top Half (DSP Block)

5 Hardware Architecture (continued)

5.1 Device Architecture (continued)

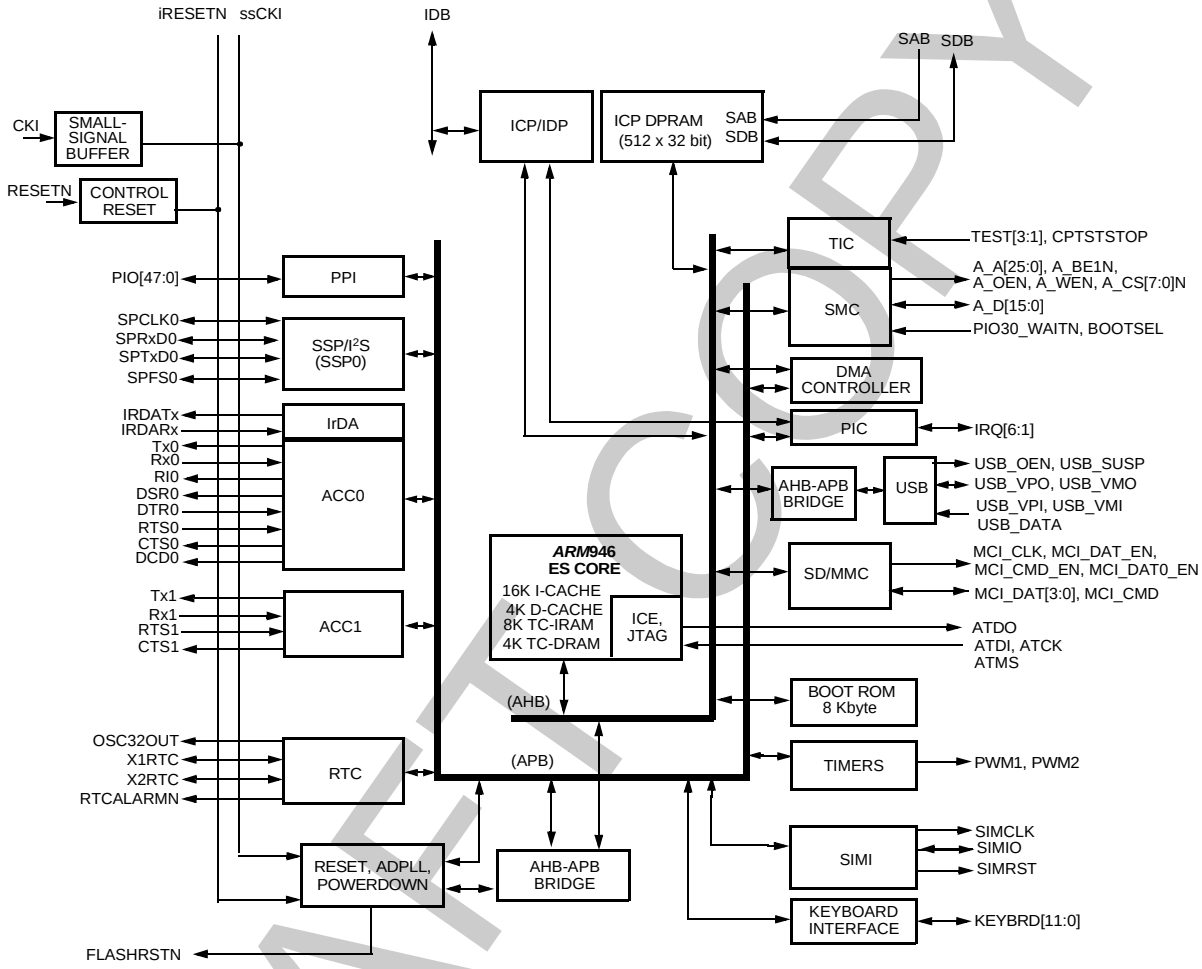


Figure 5.1-2 T8307 Block Diagram—Bottom Half (CP Block)

5 Hardware Architecture (continued)

5.1 Device Architecture (continued)

5.1.1 Digital Signal Processor (DSP) Blocks

The DSP block contains a DSP core with mask-programmable ROM, dual-port RAM, one timer, one 2-bit bit input/output unit (BIO), and JTAG with hardware development system (HDS) debug units. In addition, the DSP block contains a clock divider and associated control, and an system and external memory interface unit (SEMI) to interface to CSP8307 analog conversion IC.

5.1.2 Microcontroller/Call Processor (CP) Block

The CP block contains an *ARM*946E-S core with mask-programmable ROM, RAM, SMC, DMA controller, PIC, test interface controller (TIC), JTAG, ICE debug unit embedded in the core, reset/powerdown/PLL unit (ADPLL), programmable I/O (PPI), two asynchronous communications controllers (ACC0 with IrDA, and ACC1), one SSP/I^2S, RTC, programmable timers, USB, SD/MMC controller, SIM, and keyboard interface.

5.1.3 Interprocessor Communication Port (ICP)/Interprocessor Debug Port (IDP)

Internally, the DSP and the CP communicate via a 512 x 32-bit shared dual-port RAM (ICP DPRAM) module by using an interrupt-based protocol. These blocks are referred to as the Interprocessor Communication Port (ICP).

The IDP implements interprocessor breakpointing and debugging features between DSP16000 and the *ARM* microcontroller.

5.2 Device Reset, Clock Sources, and Boot Procedure

This section describes the different ways in which the device is reset, the clock sources for the various blocks in the device, and the booting procedure (i.e., the execution of the code after reset deassertion).

5.2.1 Device Reset Setup

Figure 5.2-1 shows T8307 device reset setup. T8307 has an active-low asynchronous device reset pin, RESETN. When this pin is forced to a logic 0, the DSP, CP, and ICP blocks are forced into their reset state.

The watchdog timer of the CP block has the capability to generate a watchdog time-out reset. This reset signal is directly ANDed with RESETN input. When watchdog reset occurs, the DSP, CP, and ICP blocks are forced into their reset state.

An active-low device test logic reset pin, referred to as TRSTN, has also been included in the test access port (TAP) set of pins for the DSP module. The test logic reset pin TRSTN, when asserted (i.e., driven to a logic 0 state), resets the JTAG and HDS in the DSP block. It does not, however, reset the DSP core and its peripherals.

ICP is reset upon device power-on and whenever the RESETN input is forced to a logic 0. The DRESETN bit in the ICP's DCCON register (see Table 9.1-1) powers up in a zero state. This causes the DSP to be held in reset state. The DSP is held in reset even after RESETN is deasserted. The DSP is released from the reset state by the CP only after it writes a 1 to the DRESETN bit (assuming the DSP JTAG is not asserting JRESET).

The DSP and its peripherals can also be reset by the DSP-JTAG commands.

5 Hardware Architecture (continued)

5.2 Device Reset, Clock Sources, and Boot Procedure (continued)

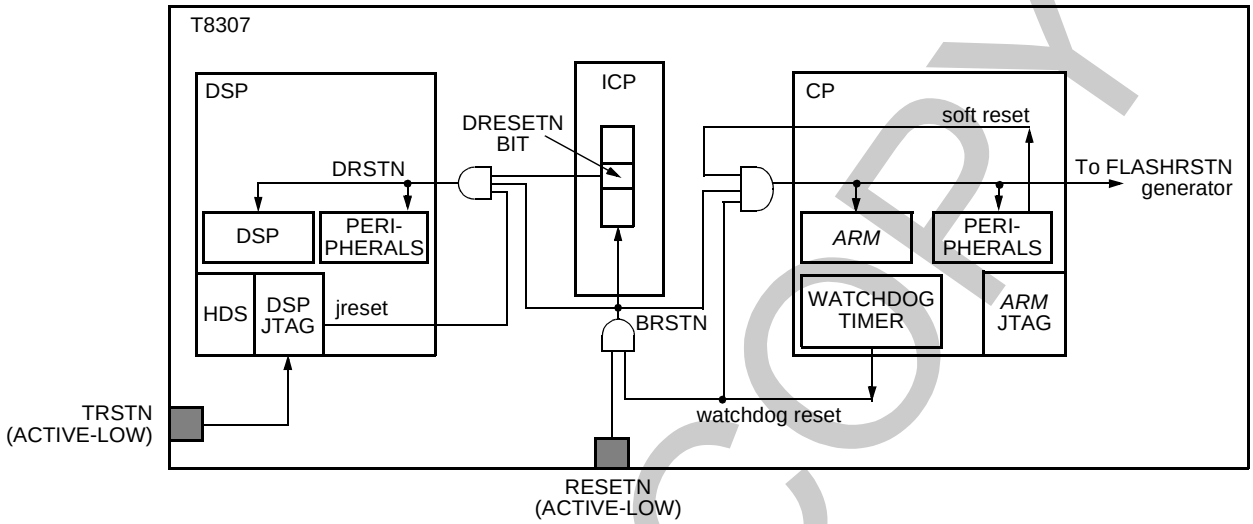


Figure 5.2-1 T8307 Device Reset Setup

5.2.2 Clock Sources

A number of clock sources are available for driving the various blocks in T8307. The device receives four different clock inputs on four separate device pins. They include the following:

- 13 MHz—30 MHz system clock on CKI pin.
- 32 kHz clock on pin X1RTC.
- DSP JTAG clock on TCK.
- CP JTAG clock on ATCK.

The system clock CKI is derived from a small-signal clock buffer. A small-signal sine wave is applied to the CKI pin. The output of the small-signal buffer is a square wave that is used as the clock to the device.

Apart from the clock supplied to the device from external sources, the device incorporates clock generators and clock synthesizers (PLLs) for the DSP and the CP portions. The PLL that generates USB clock is referred to as the UPLL, and the PLL that generates *ARM* and DSP clocks is called the ADPLL. The blocks in the device have the option of using clocks from these sources.

The DSP system clock, DCLK, is chosen from one of the following:

- CKI from the small-signal buffer.
- X1RTC—32 kHz clock.
- ADPLL (with the DSP-side postdivider).
- DTCK (DSP-JTAG clock used by DSP tools).

The DSP timer, like all the other DSP peripherals, uses DSP clock DCLK to interface to the DSP16000 core. However, the DSP timer always uses CKI as its time-counting clock.

The CP clock, ACLK, is chosen from one of the following:

- CKI from the small-signal buffer.
- X1RTC—32 kHz clock.
- ADPLL (with the *ARM*-side postdivider).

Unlike the DSP timer, the *ARM* timers use APB clock for both bus interfacing and time counting.

5 Hardware Architecture (continued)

5.2 Device Reset, Clock Sources, and Boot Procedure (continued)

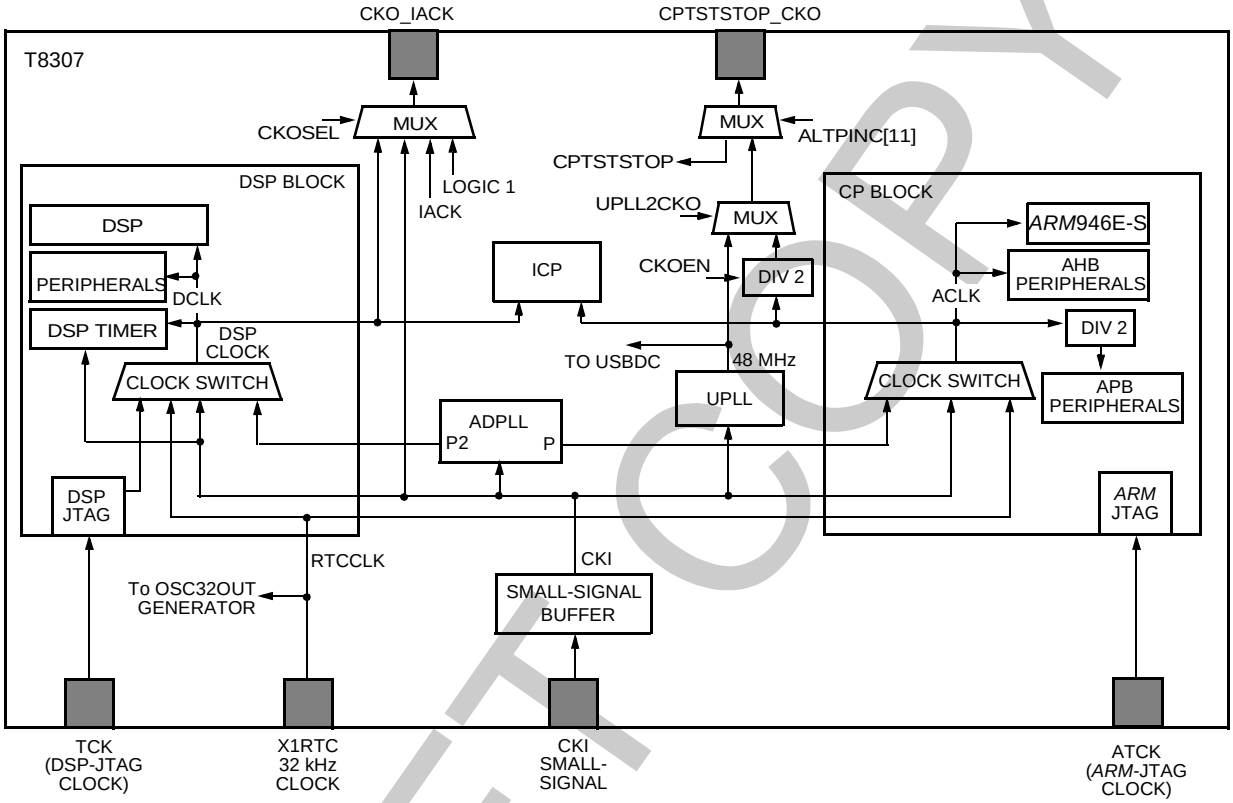


Figure 5.2-2 T8307 Clock Sources

Figure 5.2-2 shows the setup in T8307 for clock sources. For both the DSP and CP blocks, CKI is picked as the clock to execute after reset is deasserted. The selection of any other clocks for the different modules is possible under software control after writing the appropriate control information in the control registers.

5 Hardware Architecture (continued)

5.2 Device Reset, Clock Sources, and Boot Procedure (continued)

5.2.2.1 Small-Signal Clock Input Buffer

The small-signal clock input buffer generates the main input clock to the DSP and the CP. The small-signal clock buffer is intended to be used so that an ac waveform (e.g., sine, square, clipped sine) will be applied to the CKI pin through an internal ac coupling capacitor as shown in Figure 5.2-3.

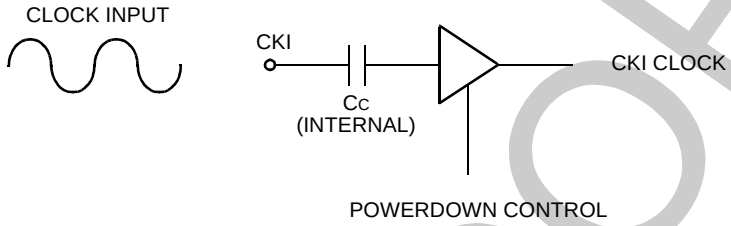


Figure 5.2-3 Simplified Block Diagram of Small-Signal Input Buffer

Due to the low amplitude of the input signal, care must be taken in PC board design to avoid crosstalk from other signals to the clock input. The clock buffer can be turned off and placed into a low-power sleep mode under software control. When turned on (or enabled), a start-up time is incurred while the small-signal buffer settles.

In order to provide fast start-up from a disabled state, the small-signal buffer circuit contains a low-power bias source to maintain the CKI pin voltage at approximately its active voltage (around 0.6 V) when disabled. Because of this bias circuit, the small-signal buffer will continue to consume up to 1 ∝A when disabled. Specifications for the small-signal input buffer are listed in Table 11.2.

5 Hardware Architecture (continued)

5.2 Device Reset, Clock Sources, and Boot Procedure (continued)

5.2.3 Boot Procedure

T8307 uses various clock sources for its operation. These include the clocks from the small-signal buffer and the on-chip PLLs—ADPLL and UPLL.

The small-signal buffer and the PLLs require a start-up time before their outputs become stable (from the time when they are enabled to generate clocks).

The PLLs use the device input clock CKI as their reference. Therefore, the external reset must be asserted for a period at least equal to or greater than the start-up time (with the assumption that the external clock input to the CKI pin is stable and running) of the small-signal clock buffer. After the device boots up, the DSP and CP can switch over to their respective PLL clocks after enabling the PLL and waiting for a period equal to their respective lock times. The following steps illustrate the procedure for proper execution of code after the device is powered on or after reset is deasserted.

5.2.3.1 Booting After RESETN Assertion

The following steps illustrate the device boot procedure after power-on. None of T8307 clock sources is producing stable signals. However, CKI pin input to the small-signal buffer is assumed to be stable and running. The DSP always boots from DSP-*ARM* shared dual-port RAM (ICP DPRAM).

1. Device reset pin RESETN is held at logic 0 state on powerup. The DSP, CP, and all the other blocks are held in reset state. TRSTN is also held at logic 0 to keep the TAP controller in reset state. The external reset must be asserted for a period at least equal to or greater than the CKI small-signal buffer start-up time.
2. RESETN and TRSTN are raised to logic 1 state. CP boots up with CKI and begins code execution from the memory segment defined by the PIO35_A_A25_BOOTSEL pin. DSP is still held in reset state because the DRESETN bit in ICP is in logic 0 state.
3. CP writes the op codes for the appropriate DSP boot-up routine in the dual-port RAM of ICP, starting from its first address.
4. CP sets the DRESETN bit in the ICP to release the DSP from reset state. DSP begins execution of boot code from the beginning of dual-port RAM in ICP using the small-signal clock as the system clock. DSP sets up its PLL postdivider now. After this, it switches over to its PLL clock and executes code at the programmed clock rate.

5 Hardware Architecture (continued)

5.2 Device Reset, Clock Sources, and Boot Procedure (continued)

5.2.3.2 Booting DSP After DSP-Reset Asserted by CP

The following steps illustrate the DSP boot procedure after DSP-reset is asserted by CP (i.e., by writing a 0 to DRESETN bit in the ICP). CKI from the small-signal buffer is assumed to be stable and running.

1. DSP is forced to reset state because the DRESETN bit in ICP is asserted by CP.
2. CP initializes the dual-port RAM of ICP with the appropriate boot code.
3. CP deasserts the DRESETN bit in the ICP to release the DSP from reset state. DSP begins execution of boot code from the beginning of dual-port RAM of ICP using the small-signal clock input CKI. The DSP sets up its PLL postdivider now. After this, it switches over to the PLL clock and executes code at the programmed clock rate.

The difference between this procedure and the previous one is that the CP is already executing code and the CKI is already stable and running.

5.3 Device Power Management

T8307 requires a number of clocks at different frequencies for its operation. These clocks are listed in the previous section. Operating different blocks of the device at different clock frequencies results in a varying amount of power consumption. The clock sources such as the PLL and the small-signal buffer also consume significant amounts of power when compared with the blocks that are operating on the clocks from these sources. Therefore, proper control and operation of the different blocks and their clock sources can result in conservation of the system power and can ensure longer battery life.

In order to shut off power in blocks that are not required to operate during specified periods of time in a system application, clock gating methodology is employed. Furthermore, software control is provided to turn on or turn off clocks on-the-fly to blocks such as peripherals. This allows the software to manage power consumption for different applications.

A number of approaches are available in T8307 for reducing power consumption. Some of these approaches include putting the device to sleep, during which all the clock circuits are off and only an interrupt is required to wake up the device. Software control of switching clocks to various blocks (in order to manage power consumption in the device) is also available. Examples for these include switching over to the PLL clock for faster operation (and, hence, more power consumption), or slow RTC (for lower power consumption) on-the-fly, shutting off clocks to various blocks (in low-power consumption mode), disabling the PLL, small-signal buffer, and powering down clock sources. This is possible by writing appropriate control bits to the control registers in the DSP and the CP. The small-signal clock buffer is controlled by the DSP and the CP. This buffer is enabled to run when either the DSP is executing code using the CKI or ADPLL clocks, or when CP is executing code using CKI or ADPLL.

A more general approach followed in T8307 to consume less power is to keep the DSP in reset state or in a low-power state (with no clocks running) until the CP initiates an operation in the DSP by issuing an interrupt through the ICP.

A general overview on power management in the DSP and the CP side is described in Section 5.3.1.

5.3.1 General Overview

The DSP section has various programming options for reducing power consumption. All of these options use on-the-fly clock selection methodology to conserve power consumption. The programming options include the following:

- Low-power standby mode.
- Standby with slow internal clock.
- Software stop with small-signal clock running.
- Software stop with small-signal clock disabled.
- Low-power standby mode with PLL enabled and selected.
- Low-power standby mode with PLL enabled and not selected.
- Software stop with PLL enabled and not selected.
- Software stop with PLL disabled and not selected.

5 Hardware Architecture (continued)

5.3 Device Power Management (continued)

The CP section has many different options for reducing the power consumption of the device. There are six different clocking modes:

- FAST mode.
- FAST-WFI mode.
- SLOW mode.
- SLOW-WFI mode.
- FAST-CLKOFF mode.
- SLOW-CLKOFF mode.

Power can also be controlled by turning off individual peripherals. These modes are configured by registers in the reset/power/clock management block.

5.3.2 CP Mode Descriptions

Table 5.3-1 shows a summary of the various powerdown modes for Call Processor block.

Table 5.3-1 Powerdown Modes for CP

Mode	Clock Source	Core Mode	Peripherals Active	Exit Method
FAST	CKI	Normal	Yes	Switch to one of the other modes.
	PLL	Normal	Yes	
FAST-WFI	CKI	WFI	Yes	Any interrupt.
	PLL	WFI	Yes	
SLOW	32 kHz Crystal	Normal	Yes	Switch back to fast mode or switch to SLOW-WFI or SLOW-CLKOFF modes.
SLOW-WFI	32 kHz Crystal	WFI	Yes	Any interrupt.
FAST-CLKOFF	CKI	No Clock	No	Any interrupt from an external interrupt source configured as asynchronous.
	PLL	No Clock	No	
SLOW-CLKOFF	32 kHz Crystal	No Clock	No	Any interrupt from an external interrupt source configured as asynchronous.

5.3.2.1 FAST Mode

FAST mode is the normal operating mode of the device. The fast clock is selected from either the clock input on CKI or the output of the PLL.

5.3.2.2 FAST-WFI Mode

In FAST-WFI mode, the fast clock is being used to control the device. The system is in a wait-for-interrupt configuration, which means that the core and the DMA are prohibited from making any memory requests until an interrupt occurs on one of the interrupt request lines. The peripherals are still active and may generate interrupts to wake up the core. The fast clock is selected from either the clock input on CKI or the output of the PLL.

5.3.2.3 SLOW Mode

SLOW mode uses the slow clock to run the device. The core and any enabled peripherals are still running, but at a reduced clock rate. The slow clock is the 32 kHz crystal input from the RTC.

5 Hardware Architecture (continued)

5.3 Device Power Management (continued)

5.3.2.4 SLOW-WFI Mode

In SLOW-WFI mode, the slow clock is being used to control the device. The system is in a wait-for-interrupt configuration, which means that the core and the DMA are prohibited from making any memory requests until an interrupt occurs on one of the interrupt request lines. The peripherals are still active and may generate interrupts to wake up the core. The slow clock is the 32 kHz crystal input from the RTC.

5.3.2.5 FAST-CLKOFF Mode

In FAST-CLKOFF mode, the clocks are off to most of the system. There is just a small piece of logic in the clock switching circuit that is being clocked. The clock to this circuit is selected from either the clock input on CKI or the output of the PLL. The core and peripherals are not being clocked, except for the RTC when it is being clocked by the 32 kHz crystal. The only way to get out of this mode is by asserting an interrupt on an external interrupt pin that has been configured as asynchronous, or a keyboard interrupt configured as asynchronous.

5.3.2.6 SLOW-CLKOFF Mode

In SLOW-CLKOFF mode, the clocks are off to most of the system. There is just a small piece of logic in the clock switching circuit that is being clocked. The clock to this circuit is the 32 kHz crystal input from the RTC. The core and peripheral are not being clocked, except for the RTC when it is being clocked by the 32 kHz crystal. The only way to get out of this mode is by asserting an interrupt on an external interrupt pin that has been configured as asynchronous, or a keyboard interrupt configured as asynchronous.

5.3.2.7 Mode Switching

Switching between the clocking modes consists of setting bits in configuration registers in the reset/power/clock management block. To select the clock source, the appropriate bit in the clock management register is set and the clock switching logic automatically switches to the clock source selected. To put the chip into WFI mode, bit 0 in the pause register should be written to 1. WFI mode will be entered when all outstanding memory operations are complete. The CLKOFF modes require the OFF bit in the clock control register to be set prior to entering WFI mode.

When either the WFI modes or CLKOFF modes are exited due to the assertion of an interrupt, the clock switching logic will automatically switch back to FAST mode to process the interrupt at the fastest possible clock rate. An example of a switching mode follows.

5.3.2.8 Switching from FAST Mode (CKI) to SLOW-WFI Mode (32 kHz Clock)

Write a 1 to bit 2 of the clock management register in the reset/power/clock management block. This will cause the clock to be switched over to the 32 kHz crystal.

Write a 1 to the pause register in the reset/power/clock management block. This will cause the chip to go into WFI mode after it completes all outstanding transactions. The chip will remain in WFI mode until an interrupt is received, at which time the clock will automatically be switched back to FAST mode using CKI.

5.4 Device Test Port and Debug

Two JTAG ports are available on T8307. One port is reserved for the DSP, and the other for the CP.

5.4.1 DSP-JTAG Test Port

DSP-JTAG is an on-chip hardware module that controls the HDS. All communication between the HDS software, running on the host computer, and the on-chip HDS is in a bit-serial manner through the TAP (test access port) of the device. The TAP pins, which are the means of communicating test information into and out of the device, consist of TDI (test data input), TDO (test data output), TMS (test mode select), TCK (test clock), and TRSTN (TAP controller reset). The registers in the HDS are connected in different scan paths between the TDI (input port) and TDO (output port) pins of the TAP. JTAG instructions are reserved to allow read and write operations to be performed between JTAG and the register chains of the HDS.

The set of test registers includes the device identification register (ID, see Table 8.9-1) and the T8307 IC boundary-scan register. All of the device's inputs and outputs are incorporated in the JTAG boundary-scan path.

5 Hardware Architecture (continued)

5.4 Device Test Port and Debug (continued)

5.4.2 CP-JTAG Test Port

The CP-JTAG port is connected only to the *ARM*946E-S processor. It consists of ATCK, ATDI, ATDO, and ATMS pins. To enable CP-JTAG port, TEST1—TEST3 pins should be left unconnected or tied to high.

The CP-JTAG pins are multiplexed with secondary functions such as PWM2 and UART1 hardware flow control. When these secondary functions are selected and CP-JTAG debugging is still desired, TEST1—TEST3 pins can be set to 011, so that CP-JTAG is daisy chained to DSP-JTAG. In such a case CP-JTAG and DSP-JTAG are both accessible through the DSP-JTAG pins. See also Section 8.9.5 for details.

When the secondary functions of CP-JTAG pins are selected and CP-JTAG debugging is not desired, TEST1—TEST3 pins can be set to “010”, so that CP-JTAG is disabled and the secondary functions on CP-JTAG pins are enabled.

For proper functioning of CP-JTAG port, ATCK frequency should be less than or equal to one-sixth (1/6) of the *ARM*946E-S system clock frequency. Specifically, ATCK frequency should be less than or equal to 1/6 of CKI frequency upon external pin reset.

6 Memory and Register Maps

6.1 Call Processor Block Memory Map

T8307 can boot from internal ROM or external ROM, depending on the value applied to PIO35_A_A25_BOOTSEL during external reset. Bit 1 of the BOOTS_ID register (Table 7.2-5) will reset to the INVERTED value that is applied to the pin PIO35_A_A25_BOOTSEL on an external pin reset. This bit is unaffected by other resets. If this bit is 0, the *ARM* core will boot from internal ROM. Otherwise, the core will boot from external ROM. For these two cases, the memory map is described in Table 6.1-1 as the Boot Map and External Memory Remap columns, respectively.

After reset, the tightly coupled memories (TC I-RAM and TC D-RAM) can be enabled by programming *ARM* coprocessor 15 (CP15) control register 1. When tightly coupled memories are enabled, accesses within the area occupied by them (0x00000000—0x1FFFFFFF) stays on-chip, effectively masking off any off-chip memory in that area. The TC I-RAM and TC D-RAM are single ported. They cannot be accessed by other AHB devices (e.g., the DMAC).

6 Memory and Register Maps (continued)

6.1 Call Processor Block Memory Map (continued)

Table 6.1-1 shows T8307 CP block memory map.

Table 6.1-1 Populated T8307 CP Block Memory Map

Address	Boot Map	External Memory Remap	Tightly Coupled Memory Remap
0x00000000—0x00001FFF	BOOTROM (8 KB)	SMC Banks 0—7 (512MB)	TC I-RAM (8 KB)
0x00002000—0x03FFFFFF	Reserved		Reserved
0x04000000—0x04000FFF	Reserved		TC D-RAM (4 KB)
0x04001000—0x1FFFFFFF	Reserved		Reserved
0x20000000—0x3FFFFFFF	SMC Banks 0—7 (512 MB)	SMC Banks 0—7 (512 MB)	SMC Banks 0—7 (512 MB)
0x40000000—0x5FFFFFFF	Reserved	Reserved	Reserved
0x60000000—0x60001FFF	BOOTROM (8 KB)	BOOTROM (8 KB)	BOOTROM (8 KB)
0x60002000—0x64017FFF	Reserved	Reserved	Reserved
0x64018000—0x6401FFFF	USB (32 KB)	USB (32 KB)	USB (32 KB)
0x64020000—0x6FFFFFFF	Reserved	Reserved	Reserved
0x70000000—0x70000FFF	SMC Registers (4 KB)	SMC Registers (4 KB)	SMC Registers (4 KB)
0x70001000—0x70002FFF	Reserved	Reserved	Reserved
0x70003000—0x70003FFF	DMAC Registers (4 KB)	DMAC Registers (4 KB)	DMAC Registers (4 KB)
0x70004000—0x700BFFFF	Reserved	Reserved	Reserved
0x700C0000—0x700DFFFF	Peripherals (128 KB)	Peripherals (128 KB)	Peripherals (128 KB)
0x700E0000—0x8000FFFF	Reserved	Reserved	Reserved
0x80010000—0x8001FFFF	Reserved (64 KB)	Reserved (64 KB)	Reserved (64 KB)
0x80020000—0xFBFFFFFF	Reserved	Reserved	Reserved
0xFC000000—0xFC0007FF	ICP DPRAM (2 KB)	ICP DPRAM (2 KB)	ICP DPRAM (2 KB)
0xFC000800—0xFFFEFFDF	Reserved	Reserved	Reserved
0xFFFEFFE0—0xFFFEFFEC	Reserved	Reserved	Reserved
0xFFFEFFF0	DCCON	DCCON	DCCON
0xFFFEFFF4	DCSTAT	DCSTAT	DCSTAT
0xFFFEFFF8	DHCON	DHCON	DHCON
0xFFFEFFFC	DHSTAT	DHSTAT	DHSTAT
0xFFFF0000— 0xFFFFFFFF	Reserved (64 KB)	Reserved (64 KB)	Reserved (64 KB)

6 Memory and Register Maps (continued)

6.1 Call Processor Block Memory Map (continued)

The following is the address map for the call processor block peripherals:

The actual physical address is the peripheral base address (0x700C0000) + offset. Registers for peripherals are populated from the lower addresses towards the upper addresses.

Table 6.1-2 *ARM* Peripheral Address Map

Offset	Peripheral	Comments
0x00000000	Reset/Pwr/Clk Control	Reset/Power Management/Clock Control
0x00001000	PIC	Programmable Interrupt Controller
0x00002000	Reserved	Reserved
0x00003000	SSP0	CP-side SSP/I^2S (SSP0)
0x00004000	Reserved	Reserved
0x00005000	Timers	Timer
0x00006000	PPI	Parallel Peripheral Interface (Pins[31:0])
0x00007000	KBI	Keyboard Interface
0x00008000	ACC0	UART0/IrDA
0x00009000	ACC1	UART1
0x0000A000	SD/MMC	SD/MMC Controller
0x0000B000	SIM	SIM Interface
0x0000C000	RTC	Real-Time Clock
0x0000D000	Reserved	Reserved
0x0000E000	Reserved	Reserved
0x0000F000	PMUX	Pin Multiplexor
0x00010000	Reserved	Reserved
0x00011000	Reserved	Reserved
0x00012000	Reserved	Reserved
0x00013000	PPI2	Parallel Peripheral Interface 2 (Pins[47:32])
0x00014000—0x0001FFFF	Reserved (12 x 4KB)	Reserved

6 Memory and Register Maps (continued)

6.2 Call Processor Block Register Table

Table 6.2-1 CP Block Register Table

Address	Name	Width	Description	RW	Reset Value	Table #
Reset, Power, and Clock Management						
0x700C0000	PAUSER	32	Pause register.	RW	0x0	7.2-2
0x700C0004	CLKM	32	Clock management register.	RW	0x0	7.2-3
0x700C0008	PWRM Set	32	Power management set register.	RW	0x0	7.2-4
0x700C000C	PWRM Clear	32	Power management clear register.	RW	0x0	7.2-4
0x700C0010	BOOTS_ID	32	Boot select/ID register.	RW	Depends on BOOTSEL pin and reset type	7.2-5
0x700C0014	CLKS	32	Clock status register.	R	0x0	7.2-6
0x700C0018	CLKC	32	Clock control register.	RW	0x0	7.2-8
0x700C001C	RSVD	32	Reserved.	—	—	—
0x700C0020	SOFTRST	32	Soft reset register (automatic self-clear).	RW	0x0	7.2-9
0x700C0024	PLLCR	32	PLL control register.	RW	0x20000	7.2-10
0x700C0028	RSTEXT	32	Reset extend register.	RW	0x32C9, only upon external pin reset	7.2-16
0x700C002C	SCLKEN	32	System clock enable register.	RW	0x02	7.2-12
0x700C0030	RSTS	32	Reset status register.	R	reflects reset type	7.2-11
0x700C0034	RSTSC	32	Reset status clear register.	RW	reflects reset type	7.2-11
0x700C0038	USBFWC Set	32	USB firmware control register set address.	RW	0x44	7.2-17
0x700C003C	USBFWC Clear	32	USB firmware control register clear address.	RW	0x44	7.2-17
0x700C0040—0x700C0048	RSVD	—	Reserved.	—	—	—
0x700C004C	WUTO	32	Wake-up time-out register.	RW	0x0	7.2-13
0x700C0050	WFCTO	32	Wait for clock time-out register.	RW	0x0	7.2-14
0x700C0054	KBTC	32	Keyboard bounce timer control register.	RW	0x0	7.2-15
SMC Control and Status Registers						
0x70000000	SMBIDCYR0	32	Idle cycle control register for memory bank 0.	RW	0x0000000F	7.3-6
0x70000004	SMBWST1R0	32	Wait-state 1 control register for memory bank 0.	RW	0x0000001F	7.3-7
0x70000008	SMBWST2R0	32	Wait-state 2 control register for memory bank 0.	RW	0x0000001F	7.3-8
0x7000000C	SMBWSTOENR0	32	Output enable assertion delay control register for memory bank 0.	RW	0x00000000	7.3-9

* Indexed by EPINDEX.
† Contains shared bits. See Section 7.14.12.6.

6 Memory and Register Maps (continued)

6.2 Call Processor Block Register Table (continued)

Table 6.2-1 CP Block Register Table (continued)

Address	Name	Width	Description	RW	Reset Value	Table #
SMC Control and Status Registers (continued)						
0x70000010	SMBWSTWENR0	32	Write enable assertion delay control register for memory bank 0.	RW	0x00000001	7.3-10
0x70000014	SMBCR0	32	Control register for memory bank 0.	RW	0x00000040	7.3-12
0x70000018	SMBSR0	32	Status register for memory bank 0.	RW	0x00000000	7.3-13
0x7000001C—0x70000034	—	—	SMC registers for memory bank 1 (same mapping as bank 0).	—	—	—
0x70000038—0x70000050	—	—	SMC registers for memory bank 2 (same mapping as bank 0).	—	—	—
0x70000054—0x7000006C	—	—	SMC registers for memory bank 3 (same mapping as bank 0).	—	—	—
0x70000070—0x70000088	—	—	SMC registers for memory bank 4 (same mapping as bank 0).	—	—	—
0x7000008C—0x700000A4	—	—	SMC registers for memory bank 5 (same mapping as bank 0).	—	—	—
0x700000A8—0x700000C0	—	—	SMC registers for memory bank 6 (same mapping as bank 0).	—	—	—
0x700000C4—0x700000DC	—	—	SMC registers for memory bank 7 (same mapping as bank 0).	—	—	—
0x700000E4	SMBWST2OENR0	32	Output enable deassertion to chip select deassertion hold delay control register for memory bank 0.	RW	0x00000000	7.3-14
0x700000E8	SMBWST2WENR0	32	Write enable deassertion to chip select deassertion hold delay control register for memory bank 0.	RW	0x00000000	7.3-15
0x700000EC	SMBWST2OENR1	32	Output enable deassertion to chip select deassertion hold delay control register for memory bank 1.	RW	0x00000000	7.3-14
0x700000F0	SMBWST2WENR1	32	Write enable deassertion to chip select deassertion hold delay control register for memory bank 1.	RW	0x00000000	7.3-15
0x700000F4	SMBWST2OENR2	32	Output enable deassertion to chip select deassertion hold delay control register for memory bank 2.	RW	0x00000000	7.3-14
0x700000F8	SMBWST2WENR2	32	Write enable deassertion to chip select deassertion hold delay control register for memory bank 2.	RW	0x00000000	7.3-15
0x700000FC	SMBWST2OENR3	32	Output enable deassertion to chip select deassertion hold delay control register for memory bank 3.	RW	0x00000000	7.3-14
0x70000100	SMBWST2WENR3	32	Write enable deassertion to chip select deassertion hold delay control register for memory bank 3.	RW	0x00000000	7.3-15
0x70000104	SMBWST2OENR4	32	Output enable deassertion to chip select deassertion hold delay control register for memory bank 4.	RW	0x00000000	7.3-14

* Indexed by EPINDEX.
† Contains shared bits. See Section 7.14.12.6.

6 Memory and Register Maps (continued)

6.2 Call Processor Block Register Table (continued)

Table 6.2-1 CP Block Register Table (continued)

Address	Name	Width	Description	RW	Reset Value	Table #
0x70000108	SMBWST2WENR4	32	Write enable deassertion to chip select deassertion hold delay control register for memory bank 4.	RW	0x00000000	7.3-15
0x7000010C	SMBWST2OENR5	32	Output enable deassertion to chip select deassertion hold delay control register for memory bank 5.	RW	0x00000000	7.3-14
0x70000110	SMBWST2WENR5	32	Write enable deassertion to chip select deassertion hold delay control register for memory bank 5.	RW	0x00000000	7.3-15
0x70000114	SMBWST2OENR6	32	Output enable deassertion to chip select deassertion hold delay control register for memory bank 6.	RW	0x00000000	7.3-14
0x70000118	SMBWST2WENR6	32	Write enable deassertion to chip select deassertion hold delay control register for memory bank 6.	RW	0x00000000	7.3-15
0x7000011C	SMBWST2OENR7	32	Output enable deassertion to chip select deassertion hold delay control register for memory bank 7.	RW	0x00000000	7.3-14
0x70000120	SMBWST2WENR7	32	Write enable deassertion to chip select deassertion hold delay control register for memory bank 7.	RW	0x00000000	7.3-15
DMAC Registers						
0x70003000	DMACIntStatus	4	This register provides the interrupt status of the DMA controller. A high bit indicates that a specific DMA channel interrupt is active.	R	0x0	7.4-1
0x70003004	DMACIntTCStatus	4	This register is used to determine whether an interrupt was generated due to the transaction completing (terminal count). A high bit indicates that the transaction completed.	R	0x0	7.4-2
0x70003008	DMACIntTCClear	4	When writing to this register, each data bit that is high causes the corresponding bit in the DMACIntTCStatus and DMACRawIntTCStatus registers to be cleared. Data bits that are low have no effect on the corresponding bit in the register.	W	—	7.4-3
0x7000300C	DMACIntErrorStatus	4	This register is used to determine whether an interrupt was generated due to an error being generated.	R	0x0	7.4-4
0x70003010	DMACIntErrClr	4	When writing to this register, each data bit that is high causes the corresponding bit in the DMACIntErrorStatus and DMACRawIntErrorStatus registers to be cleared. Data bits that are low have no effect on the corresponding bit in the register.	W	—	7.4-5

* Indexed by EPINDEX.
† Contains shared bits. See Section 7.14.12.6.

6 Memory and Register Maps (continued)

6.2 Call Processor Block Register Table (continued)

Table 6.2-1 CP Block Register Table (continued)

Address	Name	Width	Description	RW	Reset Value	Table #
DMAC Registers (continued)						
0x70003014	DMACRawInt TCStatus	4	This register provides the raw status of DMA terminal count interrupts prior to masking. A high bit indicates that the interrupt request is active prior to masking.	R	0x0	7.4-6
0x70003018	DMACRawInt ErrorStatus	4	This register provides the raw status of DMA error interrupts prior to masking. A high bit indicates that the interrupt request is active prior to masking.	R	0x0	7.4-7
0x7000301C	DMACEnbldChns	4	This register shows which DMA channels are enabled. A high bit indicates that a DMA channel is enabled.	R	0x0	7.4-8
0x70003020	DMACSoftBReq	16	This register allows DMA burst requests to be generated by software.	RW	0x0000	7.4-9
0x70003024	DMACSoftSReq	16	This register allows DMA single requests to be generated by software.	RW	0x0000	7.4-10
0x70003028	DMACSoftLBReq	16	This register allows DMA last burst requests to be generated by software.	RW	0x0000	7.4-11
0x7000302C	DMACSoftLSReq	16	This register allows DMA last single requests to be generated by software.	RW	0x0000	7.4-12
0x70003030	DMAC Configuration	2	This register is used to configure the DMA controller.	RW	0x0	7.4-13
0x70003034	DMACSync	16	This register enables or disables synchronization logic for the DMA request signals.	RW	0x0000	7.4-14
0x70003100	DMACC0SrcAddr	32	DMA channel 0 source address.	RW	0x00000000	7.4-15
0x70003104	DMACC0DestAddr	32	DMA channel 0 destination address.	RW	0x00000000	7.4-16
0x70003108	DMACC0LLI	32	DMA channel 0 linked list address.	RW	0x00000000	7.4-17
0x7000310C	DMACC0Control	32	DMA channel 0 control.	RW	0x00000000	7.4-18
0x70003110	DMACC0 Configuration	19	DMA channel 0 configuration register.	RW	0x00000	7.4-22
0x70003114—0x7000311C	RSVD	—	Reserved	—	—	—
0x70003120—0x7000313C	—	—	DMA channel 1 registers (same mapping as DMA channel 1).	—	—	—
0x70003140—0x7000315C	—	—	DMA channel 2 registers (same mapping as DMA channel 1).	—	—	—
0x70003160—0x7000317C	—	—	DMA channel 3 registers (same mapping as DMA channel 1).	—	—	—

* Indexed by EPINDEX.
† Contains shared bits. See Section 7.14.12.6.

6 Memory and Register Maps (continued)

6.2 Call Processor Block Register Table (continued)

Table 6.2-1 CP Block Register Table (continued)

Address	Name	Width	Description	RW	Reset Value	Table #
Programmable Interrupt Controller (PIC)						
0x700C1000	IRSR	32	Interrupt request status register.	R	0x0	7.5-4
0x700C1004	RSVD	32	Reserved.	—	—	—
0x700C1008	IRER set	32	Interrupt request enable set register.	RW	0x0	7.5-5
0x700C100C	IRER clear	32	Interrupt request enable clear register.	RW	0x0	7.5-5
0x700C1010	SOFTIRQ	32	Soft interrupt request register.	RW	0x0	7.5-8
0x700C1014	RSVD	32	Reserved.	—	—	—
0x700C1018—0x700C1090	IPCR1—IPCR31	32	Interrupt priority control registers 1—31.	RW	0x0	7.5-3
0x700C1094	ISRI	32	In-service IRQ register.	R	0x0	7.5-1
0x700C1098	ISRF	32	In-service FIQ register.	R	0x0	7.5-1
0x700C109C	IRQCLR	32	Interrupt request source clear register (automatic self-clear).	RW	0x0	7.5-7
0x700C10A0	IPER Set	32	Interrupt priority enable register (set).	RW	0xFFFFFFFF	7.5-6
0x700C10A4	IPER Clear	32	Interrupt priority enable register (clear).	RW	0xFFFFFFFF	7.5-6
0x700C10A8—0x700C10C0	FPIRQC1—FPIRQC7	32	Fully programmable interrupt 1—7 control registers.	RW	0x0	7.5-9
0x700C10C4—0x700C10C8	FPIRQC27—FPIRQC28	32	Fully programmable interrupt 27—28 control registers.	RW	0x0	7.5-9
0x700C10CC	SFCSEL	32	Slow to fast clock select register.	RW	0x0	7.5-10
0x700C10D0	RSVD	32	Reserved.	—	—	—
0x700C10D4	BPWFCC	32	Bypass the wait for clock counter register.	RW	0x0	7.5-11
0x700C10D8—0x700C10DC	FPIRQC29—FPIRQC30	32	Fully programmable interrupt 29—30 control registers.	RW	0x0	7.5-9
0x700C10E0—0x700C10FF	RSVD	—	Reserved.	—	—	—
PPI Registers, Group 1						
0x700C6000	PPI1DIR	32	Port data direction register.	RW	0x0	7.6-1
0x700C6004	RSVD	32	Reserved.	—	—	—
0x700C6008	PPI1IE	32	Port interrupt enable register.	RW	0xFFFFFFFF	7.6-9
0x700C600C	PPI1SEN	32	Port sense register.	RW	0x0	7.6-5
0x700C6010	PPI1POL	32	Port polarity register.	RW	0x0	7.6-7
0x700C6014—0x700C6018	RSVD	32	Reserved.	—	—	—
0x700C601C	PPI1DATA Clear	32	Port data clear address.	RW	0x0	7.6-3
0x700C6020	PPI1DATA Set	32	Port data set address.	RW	0x0	7.6-3
0x700C6024—0x700C607C	RSVD	—	Reserved.	—	—	—
PPI Registers, Group 2						
0x700D3000	PPI2DIR	32	Port data direction register.	RW	0x0	7.6-2

* Indexed by EPINDEX.
† Contains shared bits. See Section 7.14.12.6.

6 Memory and Register Maps (continued)

6.2 Call Processor Block Register Table (continued)

Table 6.2-1 CP Block Register Table (continued)

Address	Name	Width	Description	RW	Reset Value	Table #
PPI Registers, Group 2 (continued)						
0x700D3004	RSVD	32	Reserved.	—	—	—
0x700D3008	PPI2IE	32	Port interrupt enable register.	RW	0xFFFFFFFF	7.6-10
0x700D300C	PPI2SEN	32	Port sense register.	RW	0x0	7.6-6
0x700D3010	PPI2POL	32	Port polarity register.	RW	0x0	7.6-8
0x700D3014—0x700D3018	RSVD	32	Reserved.	—	—	—
0x700D301C	PPI2DATA Clear	32	Port data clear address.	RW	0x0	7.6-4
0x700D3020	PPI2DATA Set	32	Port data set address.	RW	0x0	7.6-4
0x700D3024—0x700D307C	RSVD	—	Reserved.	—	—	—
UART ACC0 Registers						
0x700C8000	ACCMIRA	32	Modem interface register A.	RW	0x0	7.7-31
0x700C8004	ACCMIRB	32	Modem interface register B.	RW	0x0	7.7-32
0x700C8008	ACCFIFOS	32	FIFO status register.	R	0xDB	7.7-18
0x700C800C	ACCS	32	ACC status register.	R	0x0	7.7-19
0x700C8010	ACCRXC	32	Receiver control register.	RW	0x0	7.7-21
0x700C8014	ACCTXC	32	Transmitter control register.	RW	0x0	7.7-27
0x700C8018	IRDAMC	32	IrDA mode control register.	RW	0x0	7.7-35
0x700C801C	ACCFIFO	32	Tx/Rx FIFO register.	RW	Unknown	7.7-29
0x700C8020	ACCFC	32	Feature control register.	RW	0x0	7.7-33
0x700C8024	ACCAC	32	Autoconfiguration control register.	RW	0x0	7.7-1
0x700C8028	ACCBDO	32	Baud divisor overflow register.	RW	0xFFFFFFFF	7.7-7
0x700C802C	ACCBDU	32	Baud divisor underflow register.	RW	0x0	7.7-8
0x700C8030	ACCBRA	32	Baud range register A.	RW	0x0	7.7-9
0x700C8034	ACCBRB	32	Baud range register B.	RW	0x0	7.7-9
0x700C8038	ACCBRC	32	Baud range register C.	RW	0x0	7.7-9
0x700C803C	ACCBRD	32	Baud range register D.	RW	0x0	7.7-9
0x700C8040	ACCBRE	32	Baud range register E.	RW	0x0	7.7-9
0x700C8044	ACCBDA	32	Baud divisor A.	RW	0x0	7.7-10
0x700C8048	ACCBDB	32	Baud divisor B.	RW	0x0	7.7-11
0x700C804C	ACCBDC	32	Baud divisor C.	RW	0x0	7.7-12
0x700C8050	ACCBDD	32	Baud divisor D.	RW	0x0	7.7-13
0x700C8054	ACCBDE	32	Baud divisor E.	RW	0x0	7.7-14
0x700C8058	ACCBDR	32	Baud divisor register.	RW	0x0	7.7-3
0x700C805C	ACCRXBC	32	Rx baud counter.	R	0x0	7.7-16
0x700C8060	ACCTXBC	32	Tx baud counter.	R	0x0	7.7-17
0x700C8064	ACCCICCR	32	Character interval counter control register.	RW	0x0	7.7-24

* Indexed by EPINDEX.
† Contains shared bits. See Section 7.14.12.6.

6 Memory and Register Maps (continued)

6.2 Call Processor Block Register Table (continued)

Table 6.2-1 CP Block Register Table (continued)

Address	Name	Width	Description	RW	Reset Value	Table #
0x700C8068	ACCCIC	32	Character interval counter.	RW	0x0	7.7-25
0x700C806C	ACCCMC0	32	Character match control register 0.	RW	0x0	7.7-26
0x700C8070	ACCCMC1	32	Character match control register 1.	RW	0x0	7.7-26
0x700C8074	ACCCMC2	32	Character match control register 2.	RW	0x0	7.7-26
0x700C8078	ACCBM	32	Baud measurement register.	R	0x0	7.7-15
0x700C807C—0x700C80FF	RSVD	—	Reserved.	—	—	—
UART ACC1 Registers						
0x700C9000—0x700C90FF	—	—	ACC1 registers (same mapping as ACC0, except that there is no Feature Register in ACC1).	—	—	—
Timer Registers						
0x700C5000	PWMMAXCA1	32	PWM maximum count register A1.	RW	0x0	7.9-1
0x700C5004	PWMMAXCB1	32	PWM maximum count register B1.	RW	0x0	7.9-1
0x700C5008	PWMCNT1	32	PWM count register 1.	R	0x0	7.9-2
0x700C500C	PWMMAXCA2	32	PWM maximum count register A2.	RW	0x0	7.9-1
0x700C5010	PWMMAXCB2	32	PWM maximum count register B2.	RW	0x0	7.9-1
0x700C5014	PWMCNT2	32	PWM count register 2.	R	0x0	7.9-2
0x700C5018	TMRCNTRATE	32	Count rate register.	RW	0x0	7.9-3
0x700C501C	WTCNT	32	WT count register.	RW	0x0	7.9-5
0x700C5020	RSVD	32	Reserved.	—	—	—
0x700C5024	TMRSR	32	Status register.	RW	0x0	7.9-8
0x700C5028	TMRIE	32	Interrupt enable register.	RW	0x0	7.9-9
0x700C502C	TMRCR	32	Control register.	RW	0x0	7.9-10
0x700C5030	ITMAXC0	32	IT maximum count register 0.	RW	0x0	7.9-6
0x700C5034	ITCNT0	32	IT count register 0.	RW	0x0	7.9-7
0x700C5038	ITMAXC1	32	IT maximum count register 1.	RW	0x0	7.9-6
0x700C503C	ITCNT1	32	IT count register 1.	RW	0x0	7.9-7
0x700C5040	ITMAXC2	32	IT maximum count register 2.	RW	0x0	7.9-6
0x700C5044	ITCNT2	32	IT count register 2.	RW	0x0	7.9-7
0x700C5048	ITMAXC3	32	IT maximum count register 3.	RW	0x0	7.9-6
0x700C504C	ITCNT3	32	IT count register 3.	RW	0x0	7.9-7
0x700C5050	ITDIV	32	IT divider register.	R	0x0	7.9-11
0x700C5054	WTDIV	32	WT divider register.	R	0x0	7.9-12
0x700C5058	PWMDIV	32	PWM divider register.	R	0x0	7.9-13
0x700C505C	PWMMAXCA3	32	PWM maximum count register A3.	RW	0x0	7.9-1
0x700C5060	PWMMAXCB3	32	PWM maximum count register B3.	RW	0x0	7.9-1
0x700C5064	PWMCNT3	32	PWM count register 3.	R	0x0	7.9-2
0x700C5068—0x700C507C	RSVD	—	Reserved.	—	—	—

* Indexed by EPINDEX.
† Contains shared bits. See Section 7.14.12.6.

6 Memory and Register Maps (continued)

6.2 Call Processor Block Register Table (continued)

Table 6.2-1 CP Block Register Table (continued)

Address	Name	Width	Description	RW	Reset Value	Table #
Keyboard Interface Registers						
0x700C7000	KBDDIR	32	Keyboard data direction register.	RW	0x0	7.10-1
0x700C7004	RSVD	32	Reserved.	—	—	—
0x700C7008	KBDIE	32	Keyboard interrupt enable register.	RW	0x0	7.10-3
0x700C700C	KBDSEN	32	Keyboard sense register.	RW	0x0	7.10-4
0x700C7010	KBDPOL	32	Keyboard polarity register.	RW	0x0	7.10-5
0x700C7014	RSVD	32	Reserved.	—	—	—
0x700C7018	KBDCNTL	32	Keyboard control register.	RW	0x0	7.10-6
0x700C701C	KBDDAT Clear	32	Keyboard data clear address.	RW	0x0	7.10-2
0x700C7020	KBDDAT Set	32	Keyboard data set address.	RW	0x0	7.10-2
RTC Registers						
0x700CC000	RTCCNTL	32	Control register.	RW	Unaffected	7.11-1
0x700CC004	RTCSECA	32	Seconds alarm register.	RW	Unaffected	7.11-4
0x700CC008	RTCSECC	32	Seconds counter register.	RW	Unaffected	7.11-5
0x700CC00C	RTCDIV	32	Divider register.	RW	Unaffected	7.11-6
CP-Side SSPI²S Registers (SSP0)						
0x700C3000	SSPCR0	16	Control register 0.	RW	0x0	7.12-4
0x700C3004	SSPCR1	16	Control register 1.	RW	0x0	7.12-5
0x700C3008	SSPDR	16	Data register.	RW	Unknown	7.12-6
0x700C300C	SSPSR	16	Status register.	R	0x3	7.12-7
0x700C3010	SSPCPSR	16	Clock prescale register.	RW	0x0	7.12-8
0x700C3014	SSPIMSC	16	Interrupt mask set or clear register.	RW	0x0	7.12-9
0x700C3018	SSPRIS	16	Raw interrupt status register.	R	0x8	7.12-10
0x700C301C	SSPMIS	16	Masked interrupt status register.	R	0x0	7.12-11
0x700C3020	SSPICR	16	Interrupt clear register.	W	0x0	7.12-12
0x700C3024	SSPDMACR	16	DMA control register.	RW	0x0	—
SIM Interface Registers						
0x700CB000	SIMBRR	32	Baud rate register.	RW	0x0	7.13-1
0x700CB004	SIMBRC	32	Baud rate counter.	R	0x0	7.13-2
0x700CB008	SIMFIFOS	32	FIFO status register.	R	0x9	7.13-3
0x700CB00C	SIMS	32	SIM status register.	R	0x0	7.13-4
0x700CB010	SIMRXC	32	Receiver control register.	RW	0x0	7.13-5
0x700CB014	SIMTXC	32	Transmitter control register.	RW	0x0	7.13-8
0x700CB018	SIMMODEC	32	Mode control register.	RW	0x0	7.13-11
0x700CB01C	SIMFIFO	32	Tx/Rx FIFO register.	RW	Unknown	7.13-13

* Indexed by EPINDEX.
† Contains shared bits. See Section 7.14.12.6.

6 Memory and Register Maps (continued)

6.2 Call Processor Block Register Table (continued)

Table 6.2-1 CP Block Register Table (continued)

Address	Name	Width	Description	RW	Reset Value	Table #
USB Device Controller Registers						
0x64018000*	TxDAT	8	Transmit FIFO data register.	W	0x0	7.14-22
0x64018004*	TxCNTL	8	Transmit FIFO byte-count low register.	RW	0x0	7.14-23
0x64018008*	TxCNTH	8	Transmit FIFO byte-count high register.	RW	0x0	7.14-23
0x6401800C*	TxCON	8	USB transmit FIFO control register.	RW	0x4	7.14-24
0x64018010*	TxFLG	8	Transmit FIFO flag register.	RW	0x8	7.14-25
0x64018014*	RxDAT	8	Receive FIFO data register.	R	0x0	7.14-26
0x64018018*	RxCNTL	8	Receive FIFO byte-count low register.	R	0x0	7.14-27
0x6401801C*	RxCNTH	8	Receive FIFO byte-count high register.	R	0x0	7.14-27
0x64018020*	RxCON	8	Receive FIFO control register.	RW	0x4	7.14-28
0x64018024*	RxFLG	8	Receive FIFO flag register.	RW	0x8	7.14-29
0x64018028	EPINDEX	8	Endpoint index register.	RW	0x0	7.14-17
0x6401802C*	EPCON†	8	Endpoint control register.	RW	Endpoint 0: 0x35 Others: 0x10	7.14-18
0x64018030*	TxSTAT	8	Endpoint transmit status register.	RW	0x0	7.14-19
0x64018034*	RxSTAT†	8	Endpoint receive status register.	RW	0x0	7.14-20
0x64018038	SOFL†	8	Start of frame low register.	RW	0x0	7.14-16
0x6401803C	SOFH†	8	Start of frame high register.	RW	0x0	7.14-15
0x64018040	FADDR	8	Function address register.	RW	0x0	7.14-21
0x64018044	SCR	8	System control register.	RW	0x0	7.14-30
0x64018048	SSR†	8	System status register.	RW	0x0	7.14-31
0x64018050	SBI†	8	Serial bus interrupt register.	RW	0x0	7.14-13
0x64018054	SBI1†	8	Serial bus interrupt register 1.	RW	0x0	7.14-14
0x64018058	SBIE	8	Serial bus interrupt enable register.	RW	0x0	7.14-11
0x6401805C	SBIE1	8	Serial bus interrupt enable register 1.	RW	0x0	7.14-12
0x64018060	REV	8	Hardware revision register.	R	0x14	7.14-32
0x64018064	LOCK	8	Suspend power-off locking register.	RW	0x1	7.14-33
0x64018068	PEND	8	Pend hardware status update register.	RW	0x0	7.14-34
0x6401806C	SCRATCH	8	Scratch firmware information register.	RW	0x0	7.14-35
0x64018070	MCSR	8	Miscellaneous control/status register.	RW	0x0	7.14-36
0x64018074	DSAV	8	Data set available.	R	0x0	7.14-37
0x64018078	DSAV1	8	Data set available 1.	R	0x0	7.14-38
0x6401807C–0x640180FC	RSVD	32	Reserved.	—	—	—

* Indexed by EPINDEX.
† Contains shared bits. See Section 7.14.12.6.

6 Memory and Register Maps (continued)

6.2 Call Processor Block Register Table (continued)

Table 6.2-1 CP Block Register Table (continued)

Address	Name	Width	Description	RW	Reset Value	Table #
USB Device Controller Registers (continued)						
0x64018100	GC1	16	USB general control register 1.	W	0x0	—
0x64018104	GC2	16	USB general control register 2.	RW	0x0	7.14-4
0x64018108	GC2SET	16	USB general control register 2 set address.	RW	0x0	7.14-4
0x6401810C	GC2CLR	16	USB general control register 2 clear address.	RW	0x0	7.14-4
0x64018110	GC3	16	USB general control register 3 (USB PLL control register).	RW	0x2F	7.14-5
0x64018114	GC3SET	16	USB general control register 3 set address.	RW	0x2F	7.14-5
0x64018118	GC3CLR	16	USB general control register 3 clear address.	RW	0x2F	7.14-5
0x6401811C	GC4	16	USB general control register 4.	R	—	7.14-6
0x64018120	GC5	16	USB general control register 5 (USB Clock control register).	RW	0x0	7.14-7
0x64018124	GC5SET	16	USB general control register 5 set address.	RW	0x0	7.14-7
0x64018128	GC5CLR	16	USB general control register 5 clear address.	RW	0x0	7.14-7
PMUX Module Registers						
0x700CF000	ALTPINC clear	32	ALTPIN control clear register.	RW	0x0	7.15-1
0x700CF004	ALTPINC set	32	ALTPIN control set register.	RW	0x0	7.15-2
0x700CF008	ALTPINC	32	ALTPIN control register.	RW	0x0	7.15-3
0x700CF00C —0x700CF014	RSVD	—	Reserved.	—	—	—
0x700CF018	ARMID	32	*ARM* ID register.	R	Unaffected	7.15-5
0x700CF01C	PMUXFC	32	Feature control register.	RW	0x0	7.15-6
0x700CF020	PURESEN1	32	Pull-up resistor enable control 1 register.	RW	0xFFFFFFFF	7.15-7
0x700CF024	PURESEN2	32	Pull-up resistor enable control 2 register.	RW	0xFFFFFFFF	7.15-9
0x700CF028	PURESEN3	32	Pull-up resistor enable control 3 register.	RW	0xFFFFFFFF	7.15-11

* Indexed by EPINDEX.
† Contains shared bits. See Section 7.14.12.6.

6 Memory and Register Maps (continued)

6.2 Call Processor Block Register Table (continued)

Table 6.2-1 CP Block Register Table (continued)

Address	Name	Width	Description	RW	Reset Value	Table #
SD/MMC Card Controller Registers						
0x700CA000	MCIPower	8	Power control register.	RW	0x00	7.16-10
0x700CA004	MCIClock	12	Clock control register.	RW	0x000	7.16-11
0x700CA008	MCIArgument	32	Argument register.	RW	0x00000000	7.16-12
0x700CA00C	MCICommand	11	Command register.	RW	0x000	7.16-13
0x700CA010	MCIRespCmd	6	Response command register.	R	0x00	7.16-15
0x700CA014	MCIResponse0	32	Response register.	R	0x00000000	7.16-16
0x700CA018	MCIResponse1	32	Response register.	R	0x00000000	7.16-16
0x700CA01C	MCIResponse2	32	Response register.	R	0x00000000	7.16-16
0x700CA020	MCIResponse3	31	Response register.	R	0x00000000	7.16-16
0x700CA024	MCIDataTimer	32	Data timer.	RW	0x00000000	7.16-18
0x700CA028	MCIDataLength	16	Data length register.	RW	0x0000	7.16-19
0x700CA02C	MCIDataCtrl	8	Data control register.	RW	0x00	7.16-20
0x700CA030	MCIDataCnt	16	Data counter.	R	0x0000	7.16-22
0x700CA034	MCIStatus	22	Status register.	R	0x000000	7.16-23
0x700CA038	MCIClear	11	Clear register.	W	—	7.16-24
0x700CA03C	MCIMask0	22	Interrupt 0 mask register.	RW	0x000000	7.16-25
0x700CA040	MCIMask1	22	Interrupt 1 mask register.	RW	0x000000	7.16-25
0x700CA044	MCISelect	4	Secure digital memory card select register.	RW	0x0	7.16-26
0x700CA048	MCIFifoCnt	15	FIFO counter.	R	0x0000	7.16-27
0x700CA04C—0x700CA07C	RSVD	—	Reserved.	—	—	—
0x700CA080—0x700CA0BC	MCIFIFO	32	Data FIFO register.	RW	0x00000000	7.16-28

* Indexed by EPINDEX.
† Contains shared bits. See Section 7.14.12.6.

6 Memory and Register Maps (continued)

6.3 Call Processor Block Interrupt Table

Table 6.3-1 CP Block IRQ Signal Mapping

Interrupt Request Line	Interrupt Type	Comment
IRQ1	IRQ1 pin interrupt.	Fully Programmable
IRQ2	IRQ2 pin interrupt.	Fully Programmable
IRQ3	IRQ3 pin interrupt.	Fully Programmable
IRQ4	IRQ4 pin interrupt.	Fully Programmable
IRQ5	IRQ5 pin interrupt.	Fully Programmable
IRQ6	IRQ6 pin interrupt.	Fully Programmable
IRQ7	Keyboard interrupt.	Fully Programmable
IRQ8	Software interrupt.	—
IRQ9	Reserved.	—
IRQ10	DMA error interrupt.	—
IRQ11	DMA terminal count interrupt.	—
IRQ12	Programmable timer interrupt.	—
IRQ13	RTC interrupt.	—
IRQ14	CP-side SSP/I^2S (SSP0) interrupt.	—
IRQ15	UART0 interrupt.	—
IRQ16	UART1 interrupt.	—
IRQ17	Reserved.	—
IRQ18	Reserved.	—
IRQ19	SIM Interrupt.	—
IRQ20	PIO pin[7:0] I/O interrupt.	—
IRQ21	PIO pin[15:8] I/O interrupt.	—
IRQ22	PIO pin[23:16] I/O interrupt.	—
IRQ23	PIO pin[31:24] I/O interrupt.	—
IRQ24	SD/MMC interrupt 0.	—
IRQ25	PIO pin[39:32] I/O interrupt.	—
IRQ26	PIO pin[47:40] I/O interrupt.	—
IRQ27	ICP interrupt.	Fully Programmable
IRQ28	UART Rx0 or Rx1 pin interrupt for line wake-up.	Fully Programmable
IRQ29	USB core interrupt.	Fully Programmable
IRQ30	USB suspend interrupt.	Fully Programmable
IRQ31	SD/MMC interrupt 1.	—

6 Memory and Register Maps (continued)

6.4 Digital Signal Processor Block Memory Map

The DSP16000 core has a modified Harvard architecture with separate program and data memory spaces (X-memory space and Y-memory space). The core differentiates between the X-memory and Y-memory spaces by the addressing unit used for the access (XAAU vs. YAAU) and not by the physical memory accessed. The core accesses the X-memory space via its 20-bit X address bus (XAB) and 32-bit X data bus (XDB). The core accesses the Y-memory space via its 20-bit Y address bus (YAB) and 32-bit Y data bus (YDB).

Although T8307 digital baseband processor DSP block memory is 16-bit word-addressable, data or instruction widths can be either 16 bits or 32 bits and applications can access the memories 32 bits at a time.

Table 6.4-1 summarizes the components of the T8307 digital baseband processor DSP block memory. The table specifies the name and size of each component, whether it is internal or external, and in which memory space(s) it resides. The two memory spaces are X-memory space and Y-memory space.

Table 6.4-1 T8307 Digital Baseband Processor DSP Block Memory Components

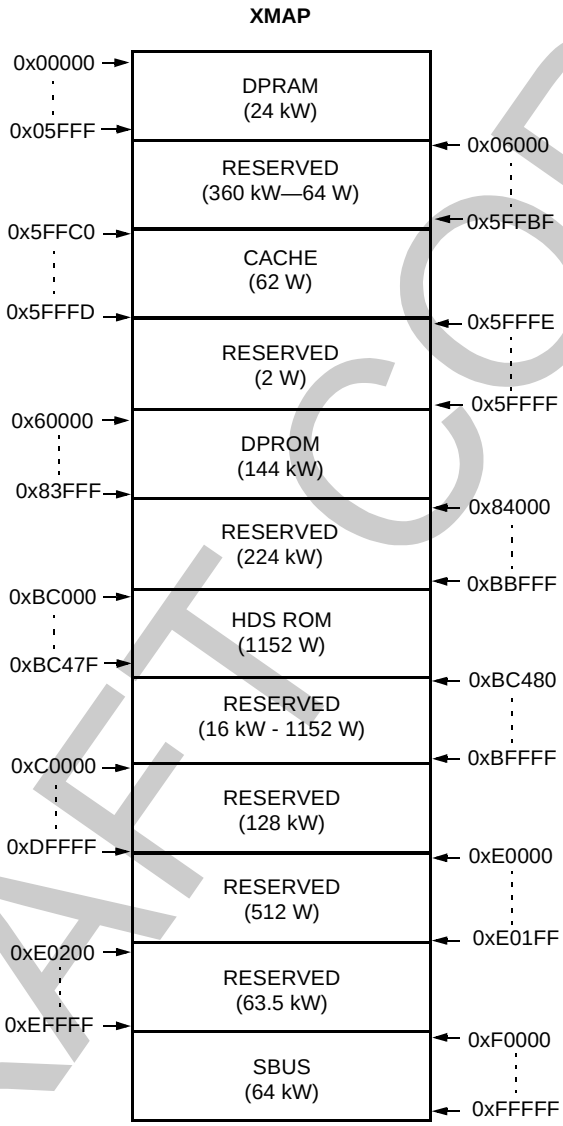
Type	Memory Component	Size	DSP	
			X-Memory Space	Y-Memory Space
Private Internal	DPRAM	24 Kwords	ϖ	ϖ
	CACHE	62 words	ϖ	ϖ
	DPROM	144 Kwords	ϖ	ϖ
Shared External	EIO	512 words	—	ϖ
Shared Internal	SBUS*	64 Kwords	ϖ	ϖ

* The SBUS internal I/O section consists of 1 Kwords of ICP DPRAM and memory-mapped registers in the DSP peripheral blocks. Only a small portion of the 64 Kwords reserved for internal I/O is actually populated with memory or registers.

6 Memory and Register Maps (continued)

6.4 Digital Signal Processor Block Memory Map (continued)

6.4.1 X-Memory Map



2381 (F).b

Figure 6.4-1 X-Memory Map

6 Memory and Register Maps (continued)

6.4 Digital Signal Processor Block Memory Map (continued)

6.4.2 Y-Memory Map

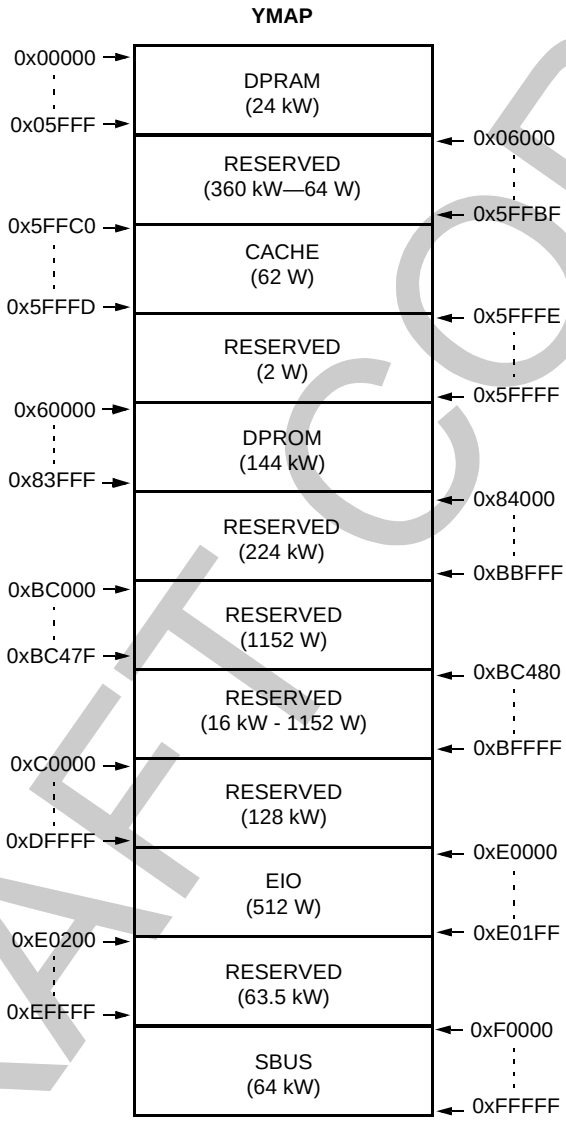


Figure 6.4-2 Y-Memory Map

6 Memory and Register Maps (continued)

6.4 Digital Signal Processor Block Memory Map (continued)

6.4.3 Private Internal Memory

The core has its own private internal memories for program and data storage: DPRAM, CACHE, and DPROM.

DPRAM is described in more detail in Section 8.10. Cache memory is described in detail in the *DSP16000 Digital Signal Processor Core* Information Manual. DPROM may be used to carry executable DSP codes.

6.4.4 Shared Internal I/O (SBUS)

The 64 Kword internal I/O memory component is accessible by the core in its X- and Y-memory spaces. Any access to this memory component is made over the system bus and is arbitrated by the SEMI. The internal shared I/O memory component consists of memory-mapped control and data registers within the following peripherals:

- SEMI.
- ICP DPRAM.
- DSP-side SSP/I^{2}S (SSP1).

Only a small portion of the 64 Kwords reserved for internal I/O is actually populated with memory or registers. An access to the internal I/O memory component takes multiple cycles to complete. DSP core writes take a minimum of two CLK cycles to complete. DSP core reads take a minimum of five CLK cycles to complete.

Table 6.4-2 is a detailed view of the 64 Kword SBUS memory component. It consists of a 4 Kword block for the memory-mapped registers of each peripheral and the ICP DPRAM. The SBUS components are directly accessible by the core. The SEMI controls access to the internal I/O memory component, which is subject to wait-state and contention penalties. An access to the internal I/O memory component causes the core to incur wait-states, and takes multiple clock cycles to complete. See Section 8.12.6.1 for details on system bus performance.

The SBUS address space is allocated for peripherals connected to the SEMI's internal system bus interface. Sixteen selects are provided at 4 kW intervals.

Table 6.4-2 SBUS Address Space

Address	SBUS Peripheral	Peripheral
0xF0000—0xF0FFF	SPERIP0 (4 kW)	SEMI
0xF1000—0xF1FFF	SPERIP1 (4 kW)	Reserved
0xF2000—0xF2FFF	SPERIP2 (4 kW)	Reserved
0xF3000—0xF3FFF	SPERIP3 (4 kW)	SSP/I^{2}S (SSP1)
0xF4000—0xF6FFF	SPERIP[4:6] (12 kW)	Reserved
0xF7000—0xF7FFF	SPERIP7 (4 kW)	ICP DPRAM
0xF8000—0xFFFFF	SPERIP[8:15] (32 kW)	Reserved

6.4.5 Shared External I/O and Memory (EIO)

External I/O and memory consist of the shared component EIO. EIO is accessible in the Y-memory space.

External I/O devices are allocated 512-word address space, as decoded by the SEMI. For T8307, the EIO select signal is expected to be used to interface to CSP8307 analog conversion IC.

Table 6.4-3 EIO Address Space

Address	EIO Chip Select	Comment
0xE0000—0xE01FF	EIO (512 W)	CSP8307

6 Memory and Register Maps (continued)

6.5 Digital Signal Processor Block Register Table

6.5.1 Memory-Mapped Registers

Table 6.5-1 DSP Block Memory-Mapped Register Table

Name	Address	Description	Bits	R/W	Type	Reset Value	Table #
System External Memory Interface (SEMI)							
ECON0	0xF0000	SEMI control.	16	R/W	Control	0x0FFF	8.15-28
DSP-Side SSP/I²S (SSP1)							
SSPCR0	0xF3000	Control register 0.	16	RW	Control	0x0	8.15-30
SSPCR1	0xF3002	Control register 1.	16	RW	Control	0x0	8.15-31
SSPDR	0xF3004	Data register.	16	RW	Data	Unknown	8.15-32
SSPSR	0xF3006	Status register.	16	R	Status	0x3	8.15-37
SSPCPSR	0xF3008	Clock prescale register.	16	RW	Control	0x0	8.15-29
SSPIMSC	0xF300A	Interrupt mask set or clear register.	16	RW	Control	0x0	8.15-34
SSPRIS	0xF300C	Raw interrupt status register.	16	R	Status	0x8	8.15-36
SSPMIS	0xF300E	Masked interrupt status register.	16	R	Status	0x0	8.15-35
SSPICR	0xF3010	Interrupt clear register.	16	W	Control	0x0	8.15-33

6 Memory and Register Maps (continued)

6.5 Digital Signal Processor Block Register Table (continued)

6.5.2 Register-Mapped Registers

For the reset values of the register-mapped registers listed in Table 6.5-2, refer to Section 8.15.3.

Table 6.5-2 DSP Block Register-Mapped Register Table

Register Name	Description	Size (Bits)	R/W†	Type‡	Signed§/ Unsigned	Core/ Off-Core	Function Block
a0, a1, a2, a3, a4, a5, a6, a7	Accumulators 0—7.	40	R/W	data	signed	core	DAU
a0h, a1h, a2h, a3h, a4h, a5h, a6h, a7h	Accumulators 0—7, high halves (bits 31—16).	16	R/W	data	signed	core	DAU
a0l, a1l, a2l, a3l, a4l, a5l, a6l, a7l	Accumulators 0—7, low halves (bits 15—0).	16	R/W	data	signed	core	DAU
a0g, a1g, a2g, a3g, a4g, a5g, a6g, a7g	Accumulators 0—7,. guard bits (bits 39—32)	8	R/W	data	signed	core	DAU
a0_1h, a2_3h, a4_5h, a6_7h	Accumulator vectors (concatenated high halves of two adjacent accumulators).	32	R/W	data	signed	core	DAU
accon	ICP control registers.	16	R/W	control	unsigned	off-core	ICP
acstat	ICP status registers.	16	R/W	c & s	unsigned	off-core	ICP
ahcon	IDP control register.	16	R/W	control	unsigned	off-core	IDP
ahstat	IDP status register.	16	R	status	unsigned	off-core	IDP
alf	AWAIT and flags.	16	R/W	c & s	unsigned	core	SYS
ar0, ar1, ar2, ar3	Auxiliary registers 0—3.	16	R/W	data	signed	core	DAU
auc0, auc1	Arithmetic unit control.	16	R/W	c & s	unsigned	core	DAU
c0, c1	Counters 0 and 1.	16	R/W	data	signed	core	DAU
c2	Counter holding register.	16	R/W	data	signed	core	DAU
cbit	BIO control.	16	R/W	control	unsigned	off-core	BIO
cloop	Cache loop count.	16	R/W	data	unsigned	core	SYS
csave	Cache save.	32	R/W	control	unsigned	core	SYS
cstate	Cache state.	16	R/W	control	unsigned	core	SYS
h	Pointer postincrement.	20	R/W	data	signed	core	XAAU
i	Pointer postincrement.	20	R/W	data	signed	core	XAAU
ioc	Memory configuration register—clock and memory map selection.	16	R/W	control	unsigned	off-core	SEMI
inc0, inc1	Interrupt control 0 and 1.	20	R/W	control	unsigned	core	SYS
ins	Interrupt status.	20	R/C††	status	unsigned	core	SYS
j	Pointer postincrement/offset.	20	R/W	data	signed	core	YAAU
jhb	High byte of **j** (bits 15—8).	8	R	data	unsigned	core	YAAU
jlb	Low byte of **j** (bits 7—0).	8	R	data	unsigned	core	YAAU
jiob	JTAG test.	32	R/W	data	unsigned	off-core	JTAG
k	Pointer postincrement/offset.	20	R/W	data	signed	core	YAAU
p0	Product 0.	32	R/W	data	signed	core	DAU

† R indicates that the register is readable by instructions; W indicates the register is writable by instructions.
‡ c & s means control and status.
§ Signed registers are in two's complement format.
†† C indicates that the register is cleared and not set.
‡‡ The IEN field (bit 14) of the **psw1** register is read only (writes to this bit are ignored).
§§ The VALUE[6:0] field (bits 6—0) are read only (writes to these bits are ignored).

6 Memory and Register Maps (continued)

6.5 Digital Signal Processor Block Register Table (continued)

6.5.2 Register-Mapped Registers (continued)

Table 6.5-2 DSP Block Register-Mapped Register Table (continued)

Register Name	Description	Size (Bits)	R/W†	Type‡	Signed§/ Unsigned	Core/ Off-Core	Function Block
p0h	High half of **p0** (bits 31—16).	16	R/W	data	signed	core	DAU
p0l	Low half of **p0** (bits 15—0).	16	R/W	data	signed	core	DAU
p1	Product 1.	32	R/W	data	signed	core	DAU
p1h	High half of **p1** (bits 31—16).	16	R/W	data	signed	core	DAU
p1l	Low half of **p1** (bits 15—0).	16	R/W	data	signed	core	DAU
pi	Program interrupt return.	20	R/W	address	unsigned	core	XAAU
pllc	Phase-locked loop control (DSP0 only).	16	R/W	control	unsigned	off-core	Clocks
pllsac	Phase-locked loop status.	16	R	status	unsigned	off-core	Clocks
powerc	Power control.	16	R/W	control	unsigned	off-core	Clocks
pr	Subroutine return.	20	R/W	address	unsigned	core	XAAU
psw0, psw1	Program status words 0 and 1.	16	R/W‡‡	c & s	unsigned	core	DAU
pt0, pt1	Pointers 0 and 1 to X-memory space.	20	R/W	address	unsigned	core	XAAU
ptrap	Program trap return.	20	R/W	address	unsigned	core	XAAU
r0, r1, r2, r3, r4, r5, r6, r7	Pointers 0—7 to Y-memory space.	20	R/W	address	unsigned	core	YAAU
rb0, rb1	Circular buffer pointers 0 and 1 (begin address).	20	R/W	address	unsigned	core	YAAU
re0, re1	Circular buffer pointers 0 and 1 (end address).	20	R/W	address	unsigned	core	YAAU
sbit	BIO status/control.	16	R/W§§	c & s	unsigned	off-core	BIO
sp	Stack pointer.	20	R/W	address	unsigned	core	YAAU
timer	Timer running count for timer.	16	R/W	data	unsigned	off-core	Timer
timerc	Timer control.	16	R/W	control	unsigned	off-core	Timer
vbase	Vector base offset.	20	R/W	address	unsigned	core	XAAU
vsw	Viterbi support word.	16	R/W	control	unsigned	core	DAU
x	Multiplier input.	32	R/W	data	signed	core	DAU
xh	High half of **x** (bits 31—16).	16	R/W	data	signed	core	DAU
xl	Low half of **x** (bits 15—0).	16	R/W	data	signed	core	DAU
y	Multiplier input.	32	R/W	data	signed	core	DAU
yh	High half of **y** (bits 31—16).	16	R/W	data	signed	core	DAU
yl	Low half of **y** (bits 15—0).	16	R/W	data	signed	core	DAU

† R indicates that the register is readable by instructions; W indicates the register is writable by instructions.
‡ c & s means control and status.
§ Signed registers are in two's complement format.
†† C indicates that the register is cleared and not set.
‡‡ The IEN field (bit 14) of the **psw1** register is read only (writes to this bit are ignored).
§§ The VALUE[6:0] field (bits 6—0) are read only (writes to these bits are ignored).

6 Memory and Register Maps (continued)

6.6 Digital Signal Processor Block Interrupt Table

Table 6.6-1 Interrupt and Trap Vector Table

Vector Description	Vector Address*		Priority
	Hexadecimal	Decimal	
Reserved	**vbase** + 0x0	**vbase** + 0	—
Reserved	**vbase** + 0x4	**vbase** + 4	—
UTRAP†	**vbase** + 0x8	**vbase** + 8	5 (Highest)
Reserved	**vbase** + 0xC	**vbase** + 12	—
TIMER	**vbase** + 0x10	**vbase** + 16	0—3‡
Reserved	**vbase** + 0x14	**vbase** + 20	—
Reserved	**vbase** + 0x18	**vbase** + 24	—
Reserved	**vbase** + 0x1C	**vbase** + 28	—
Reserved	**vbase** + 0x20	**vbase** + 32	—
Reserved	**vbase** + 0x24	**vbase** + 36	—
Reserved	**vbase** + 0x28	**vbase** + 40	—
Reserved	**vbase** + 0x2C	**vbase** + 44	—
INT0	**vbase** + 0x30	**vbase** + 48	0—3
Reserved	**vbase** + 0x34	**vbase** + 52	—
SSP	**vbase** + 0x38	**vbase** + 56	0—3
Reserved	**vbase** + 0x3C	**vbase** + 60	—
Reserved	**vbase** + 0x40	**vbase** + 64	—
ICP	**vbase** + 0x44	**vbase** + 68	0—3
Reserved	**vbase** + 0x48	**vbase** + 72	—
Reserved	**vbase** + 0x4C	**vbase** + 76	—
Reserved	**vbase** + 0x50	**vbase** + 80	—
Reserved	**vbase** + 0x54	**vbase** + 84	—
Reserved	**vbase** + 0x58	**vbase** + 88	—
Reserved	**vbase** + 0x5C	**vbase** + 92	—
icall 0§	**vbase** + 0x60	**vbase** + 96	—
icall 1	**vbase** + 0x64	**vbase** + 100	—
⋮	⋮	⋮	—
icall 62	**vbase** + 0x158	**vbase** + 344	—
icall 63	**vbase** + 0x15C	**vbase** + 348	—

* **vbase** contains the base address of the 352-word vector table.

† Reserved for HDS.

‡ The programmer specifies the relative priority levels 0—3 for hardware interrupts via **inc0** and **inc1** (see Table 8.15-15). Level 0 indicates a disabled interrupt. If multiple concurrent interrupts with the same assigned priority occur, the core first services the interrupt that has its status field in the relative least significant bit location of the **ins** register (see Table 8.15-16); i.e., the core first services the interrupt with the lowest vector address.

§ Reserved for system services.

7 Call Processor (CP) Block

7.1 Overview

The CP contains the following features:

- *ARM* 946E-S processor core and system bus with on-chip peripherals targeted at cellular phone applications.
- Low-power sleep mode.
- Capability of shutting down clocks to individual peripherals.

Core information is obtained from the data sheets and technical manuals available from *ARM* Limited.

7.1.1 CP System Functions

Referring to Figure 7.1-1, the CP system functions are provided by the following units: on-chip ROM memory, SMC (*ARM PrimeCell* SMC), a PIC, a reset/clock/power unit, a DMA, a TIC, and a peripheral bridge that converts the system bus to peripheral bus. The features of each of the units follow. See Section 6.1 for a CP address map.

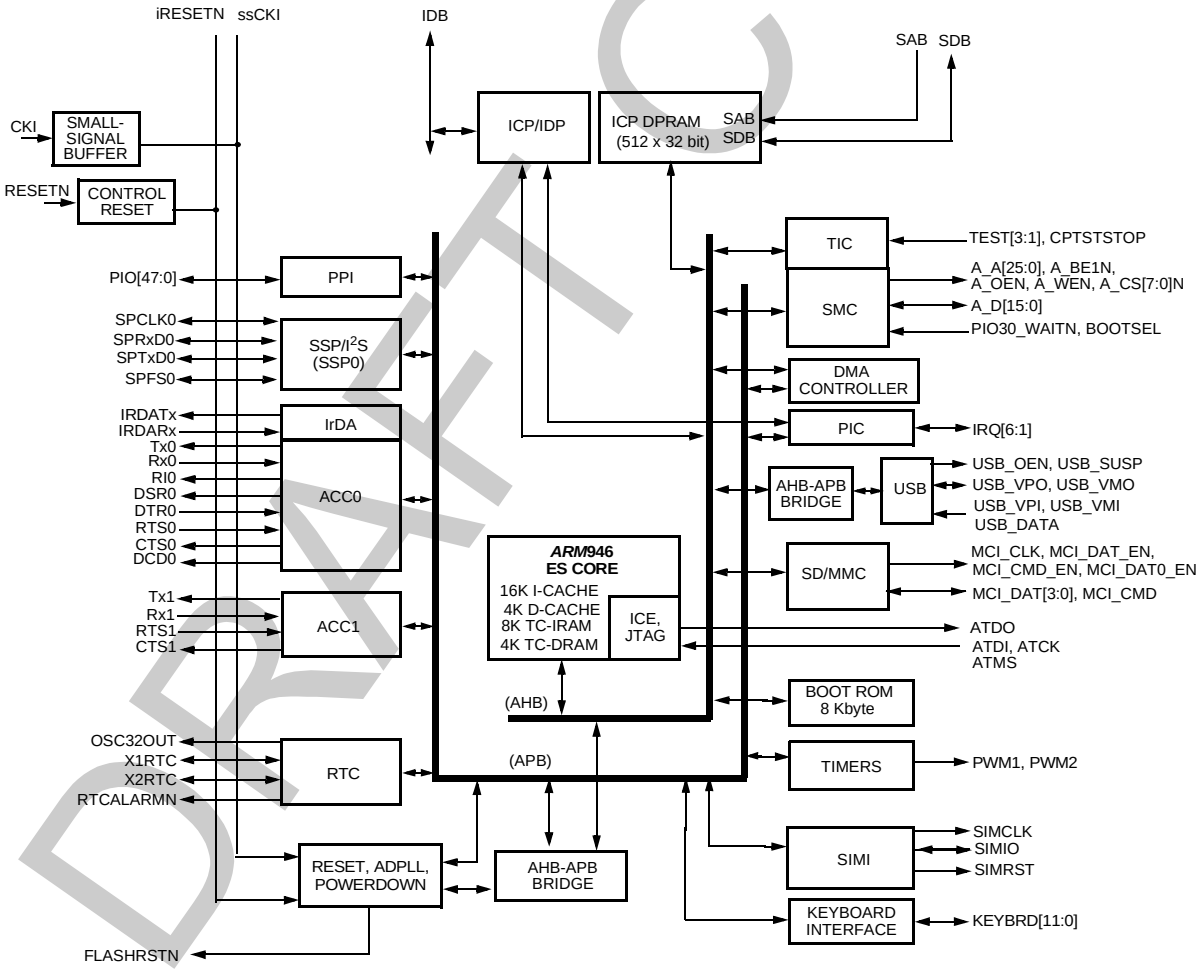


Figure 7.1-1 Block Diagram of the CP-Block

7 Call Processor (CP) Block (continued)

7.1 Overview (continued)

7.1.1.1 Reset/Power/Clock Management Features

- Programmable power control for on-chip peripherals.
- Reset status to identify the source of a reset.
- PLL control.
- Programmable clock source; choose between external CKI pin, PLL, or 32 kHz input.

For more information about reset/power/clock management features, see Section 7.2.

7.1.1.2 Programmable Interrupt Controller (PIC) Features

- 31 maskable interrupt inputs.
- 2 programmable priority groups (IRQ, FIQ).
- 31 programmable priority levels.
- 6 fully programmable external interrupt input pins.

For more information about the PIC, see Section 7.5.

7.1.1.3 External Memory Interface (SMC) Features

The SMC is implemented as the *PrimeCell Static Memory Controller*. This block is well documented in the *ARM PrimeCell Static Memory Controller (PL092)* Technical Reference Manual. This document is accessible from the *ARM* web site.

7.1.1.4 DMA Controller Features

The DMA is documented in the *ARM PrimeCell Dual DMA Master DMA Controller (PL080)* Technical Reference Manual. This document is accessible from the *ARM* web site.

7.1.2 CP-Peripherals

The peripherals available for the CP include the following:

- Parallel peripheral interface (PPI) pins.
- CP-side synchronous serial port (SSP/I^2S).
- Two asynchronous serial communication controllers (ACCs), one with IrDA.
- Four timers.
- Keyboard interface.
- Real-time clock (RTC).
- SIM interface.
- USB device controller.
- Secure digital/multimedia memory card controller.

A description of each of these units follows.

7.1.2.1 Parallel Peripheral Interface (PPI) Features

- Each bit is programmed as either an input or an output.
- Inputs are programmed to be level-sensitive or transition-detect.
- Outputs are programmed to be open-drain or direct-drive.
- Programmable polarity (inverted or not) for inputs and outputs.
- Edges (transitions) on any one of the inputs in the port cause a port specific interrupt request to be asserted.
- Each I/O can be programmed to have an optional internal pull-up resistor connected.

For more details about the PPI, see Section 7.6.

7.1.2.2 Synchronous Serial Port (SSP/I^2S) Features

The SSP is documented in the *ARM PrimeCell Synchronous Serial Port (PL022)* Technical Reference Manual.

For more details about the SSP/I^2S, see Section 7.12.

7.1.2.3 Asynchronous Communications Controller (ACC) Features

- Full-duplex asynchronous communication.
- 32 bytes of FIFO for both receive and transmit.
- FIFO threshold interrupts.
- 1 start bit, 7 or 8 data bits, 1 optional parity bit, 1 or 2 stop bits.
- Programmable baud rate (17-bit system clock divider).
- Complete status reporting capabilities.
- Single interrupt routed to the PIC.
- Support for DMA transfers.
- Autoconfiguration mode with autobaud and autoformat operation.
- Hardware loopback for autoconfiguration mode.

7 Call Processor (CP) Block (continued)

7.1 Overview (continued)

- Character matching interrupts (up to three different characters).
- Modem support (RTS, CTS, DSR, DTR, DCD, RI) for DTE or DCE applications.
- Rx IOCE timer.
- Software flow control.

For more information about the ACC, see Section 7.7.

7.1.2.4 IrDA Features

- Operates at speeds up to 115.2 kbits/s.
- Programmable pulse-width to the IrDA transceiver.

For more information about the IrDA, see Section 7.8.

7.1.2.5 Timer Features

- Pulse-width modulator with one output channel.
- Watchdog timer.
- Interval timer (IT) with four independent timers.
- Generation of a shared interrupt request from the four interval timer channels and the three pulse-width modulators.
- Generation of a watchdog timer reset signal.

For more information about the timer features, see Section 7.9.

7.1.2.6 Keyboard Interface Features

- Maximum 6 x 6 matrix is supported.
- Pins that are not needed for the keyboard can be used as programmable I/O.
- Keyboard inputs must be active for a selectable minimum pulse-width before interrupt generation.
- Each I/O can be programmed to have an optional internal pull-up resistor connected.

Keyboard interface pins that are used as general-purpose programmable I/O have the following features:

- Each bit is programmable as either an input or an output.
- Inputs are programmable to be level-sensitive or transition-detect.
- Outputs are programmable to be open-drain or direct-drive.
- Programmable polarity (inverted or not) for inputs and output.
- Each I/O can be programmed to have an optional internal pull-up resistor connected.

For more information about the keyboard interface, see Section 7.10.

7.1.2.7 Real-Time Clock (RTC) Features

- Can maintain up to 17 years of range.
- Programmed time alarm interrupt.
- Alarm output pin.

For more information about the RTC, see Section 7.11.

7.1.2.8 USB Interface

The USB interface module is an integrated USB 1.1 device controller. USB interface supports the following functions:

- 12 Mbits/s USB 1.1 device operation.
- Each of the 16 unidirectional endpoints supports control, interrupt, isochronous and bulk transfer.

7.1.2.9 SD/MMC Interface

The SD/MMC module is an *ARM PrimeCell* PL180. SD/MMC interface supports the following functions:

- Supports multimedia/secure digital memory card.
- Supports DMA transfers.

7 Call Processor (CP) Block (continued)

7.2 Reset/Power/Clock Management

The reset, power, and clock management registers control clock distribution to the peripherals, identify the source of a RESET condition (external RESET pin, watchdog timer, or software reset), and control the many clock sources (CKI, phase-locked loop (PLL), and 32 kHz external).

7.2.1 Operation

The reset, power, and clock management unit is controlled by sixteen registers. The two power management registers disable the clock signals to individual peripherals in order to save power. Each clock is disabled immediately upon setting one of the register bits to one. The reset, power, and clock management unit has a signal that indicates it has entered the wait-for-interrupt (WFI) sleep state.

The microcontroller clock is switched to an external 32 kHz oscillator by setting appropriate bits in the clock management and clock control registers.

Within the reset status register (see Table 7.2-11), there are three status bits identifying the cause of the most recent full chip reset. In all cases, the core resumes fetching instruction at memory address 0x00000000. One bit indicates that the RESETN pin is active, the second indicates that a device reset is forced by the watchdog timer in the programmable timer unit, and the third indicates a software reset. The three conditions are mutually exclusive, and appropriate actions can be taken within the boot code depending on which bit is set.

7.2.2 Operation of the Clock Switching Logic

There are two modes of operation for the clock switching logic. The first mode is manual mode. When in this mode, the PLL is turned on and off by the user. The second mode is automatic mode. When in this mode, T8307 switches between clocks and automatically turns off the clocks that are not being used.

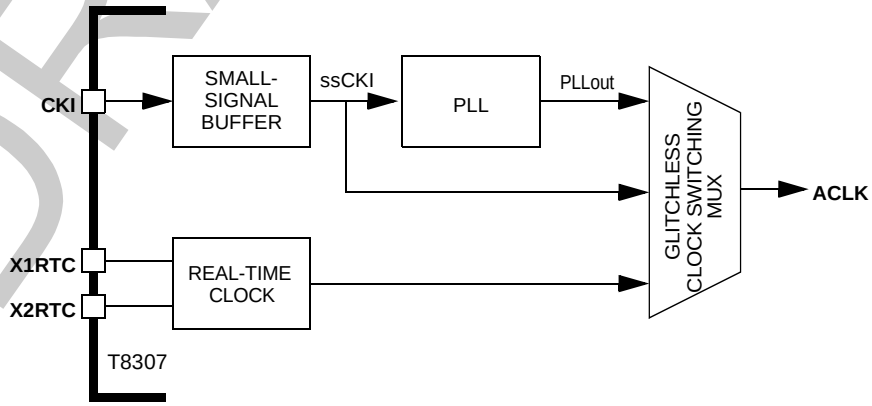
Manual mode is activated by setting the manual bit in the clock control register (see Table 7.2-8) to 1. When the manual bit is set, the clock switching is software controlled. This allows software to control the PLL. For example, switching from the external clock to PLL clock, the PLL enable bit in the clock control register is set to 1, then the PLL clock bit in the clock management register (see Table 7.2-3) is set to 1. With this control, the PLL continues running while using another clock as the source clock.

Automatic mode is activated by setting the manual bit in the clock control register to 0. When in this mode, the clock management register bit corresponding to the new desired clock is set to 1. The system automatically switches to that clock as its source clock. In addition, it turns off any clocks that are not used.

When an interrupt is encountered while in either manual or automatic mode, the system automatically switches back to the last fast clock. If the corresponding bit in the slow to fast clock select register (see Table 7.5-10) is set to 1.

The PLL oscillator is controlled by bit 2 of the clock control register while in manual mode. The PLL generates a clock signal when bit 2 is set to 1. It typically takes about 50 ∝s for the PLL oscillator to restart and lock from the inactive state.

Figure 7.2-1 shows a block diagram of the clock switching logic.



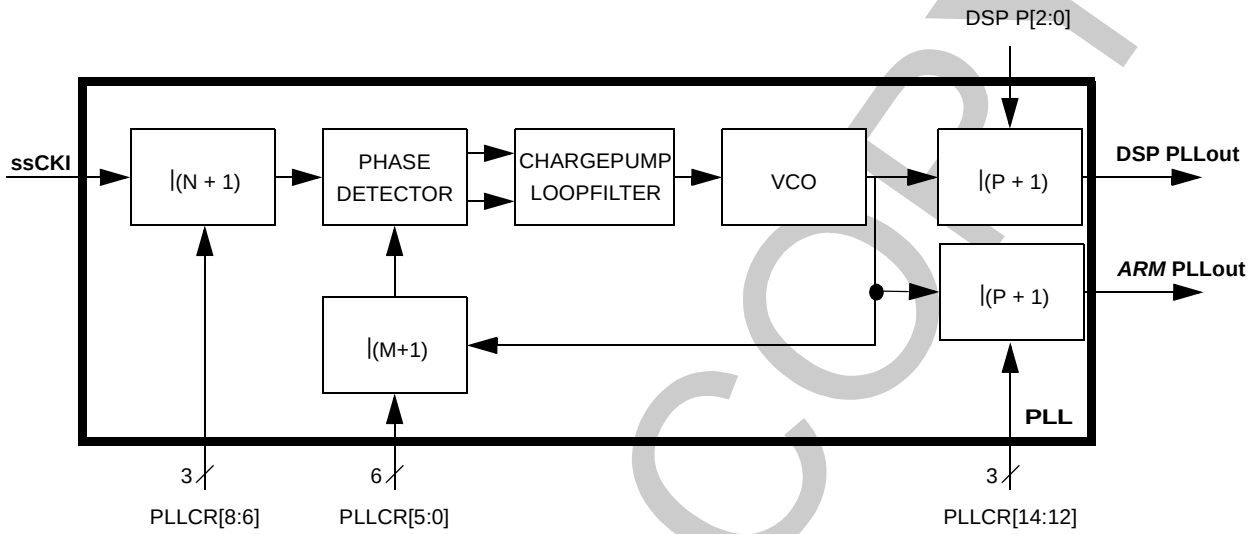
Note: In manual mode, PLL_on is controlled by bit 2 of the clock control register; in automatic mode, PLL_on is controlled by hardware.

Figure 7.2-1 Clock Switching Logic

7 Call Processor (CP) Block (continued)

7.2 Reset/Power/Clock Management (continued)

Figure 7.2-2 shows a block diagram of the PLL.



Notes:
Signals shown in bold are control bits from the PLL control register (See Table 7.2-10).

Other signals from the clock control register also control the clock source.

Figure 7.2-2 Clock Source Block Diagram

T8307 contains two PLLs, but the operation of these PLLs differs from previous Trident devices. One PLL (UPLL) is dedicated to the USB. The other PLL (ADPLL) provides the high-speed master clocks for both the *ARM* and DSP. When the ADPLL is being used, both DSP and *ARM* clocks are generated from the same VCO. Separate postdividers are provided. Thus, the *ARM* and DSP PLL clocks can be set to different frequencies over a limited range. Most of the PLL settings are controlled solely from the *ARM* side. The DSP has control only of the bits for setting its postdivide value.

The input to the PLL comes from the input clock CKI. The PLL cannot operate without this external input clock.

When the PLL is turned on, it will take some time to stabilize and lock to the programmed frequency. The clock switching logic waits until lock occurs before switching to the PLL clock.

The following equation determines the PLL frequency:

$$fPLLout = fssCKI \times \frac{(M + 1)}{(N + 1)\,(P + 1)}$$

The following equation determines the VCO frequency:

$$fVCO = fssCKI \times \frac{(M + 1)}{(N + 1)}$$

7 Call Processor (CP) Block (continued)

7.2 Reset/Power/Clock Management (continued)

The clock switching logic waits for the PLL to lock before switching to the PLL as the system clock.

The following rules govern proper programming and use of the PLL:

- Change the bits in the PLL control register **only** while the PLL is not providing the internal clock source. Bear in mind that changes to the PLL control register may also affect the clock to the DSP.
- To select the PLL as the internal clock, set bit 1 of the clock management register (see Table 7.2-3).
- To deselect the PLL, select another clock in the clock management register.
- The PLL is powered down automatically by the clock switching logic when it is not in use.
- Do not remove the input clock (CKI) before the PLL is powered down.
- With a CKI frequency of 13 MHz, the value of N must be zero.
- Writing to the PLL control register (see Table 7.2-10) causes the PLL to initiate an automatic trim sequence to lock the PLL to the programmed frequency. The PLL should not be used as the clock source until it has locked. The lock time is less than 2 ms.
- The CKI small signal input should not be removed until the PLL has been powered down.

When the PLL is powered down and subsequently powered up again, the time for the PLL to lock is less than 50 ∝s.

7.2.3 Latency

The switch between the CKI-based clock and the PLL-based clock is synchronous. This method results in the actual switch taking place several cycles after the PLLSEL bit is changed. During this time, actual code is executed, but at the precedent clock rate. The PLL is not disabled until the switch back to CKI is complete.

Table 7.2-1 PLL Specifications

Parameters	Min	Max	Unit
Phase Detector Input Frequency	10	100	MHz
VCO Frequency	300	500	MHz
M	3	63	—
N	0	7	—
P	0	7	—
Output Duty Cycle (P + 1 = 2, 4, 6, 8)	48	52	%
Output Duty Cycle (P + 1 = 3, 5, 7)	45	55	%
Output Duty Cycle (P + 1 = 1)	40	60	%
Peak-to-peak Jitter at ACO (t_{jit})	–150	150	ps
Lock Time	—	50	∝s
Autotrim Time	—	2	ms

7 Call Processor (CP) Block (continued)

7.2 Reset/Power/Clock Management (continued)

7.2.4 Registers

The reset, power, and clock management unit consists of sixteen registers to program status of the clock and power configuration of the system.

7.2.4.1 Pause Register (PAUSER)

The pause register (see Table 7.2-2) puts the chip into wait-for-interrupt (WFI) mode. Writing a 1 to this bit causes the system to go into WFI mode immediately. This register automatically clears during an interrupt.

Table 7.2-2 Pause Register (PAUSER), Address (0x700C0000)

Bit	31—1	0
Name	RSVD	PAUSE

Bit	Name	Description
31—1	RSVD	Reserved.
0	PAUSE	Specifies if the system is in wait-for-interrupt (WFI) mode. If 1, the system is in WFI mode. If 0, the system is in normal mode.

Note: Trying to write to this bit during an interrupt will result in a write of 0.

7.2.4.2 Clock Management Register (CLKM)

The clock management register (see Table 7.2-3) selects the source of the clock to the chip blocks. Writing a 1 to a bit in this register causes the clock switching logic to switch to the selected clock. If more than one bit is set, the lowest numbered bit takes precedence. For example, if bits 1 and 0 are both written to 1, the clock switches to the external clock. If all 0s are written, nothing happens.

Table 7.2-3 Clock Management Register (CLKM), Address (0x700C0004)

Bit	31—3	2	1	0
Name	RSVD	CMRT	PLLC	CMEC

Bit	Name	Description
31—3	RSVD	Reserved.
2	CMRT	Switches the clock source to the RTC. If 1, the clock switching logic switches the source clock to the RTC, and then clears this register. If 0, the logic for the RTC is not activated.
1	PLLC	Switches the clock source to the PLL clock. If 1, the clock switching logic switches the source clock to the PLL clock, and then clears this register. If 0, the logic for the PLL is not activated.
0	CMEC	Switches the clock source to the external clock. If 1, the clock switching logic switches the source clock to the external clock, and then clears this register. If 0, the logic for the external clock is not activated.

7 Call Processor (CP) Block (continued)

7.2 Reset/Power/Clock Management (continued)

7.2.4.3 Power Management Registers (PWRM)

The power management registers control the clock to individual devices in the chip. To set the bits, a write is done to the power management set register (see Table 7.2-4). Each data bit that is 1 sets the corresponding bit in the power management register. To clear the bits, a write is done to the power management clear register. Each data bit that is 1 clears the corresponding bit in the power management register. Reading either clear address or set address of this register returns current power management register value.

Table 7.2-4 Power Management Registers (PWRM), Addresses (Clear 0x700C000C/Set 0x700C0008)

Bit	31—21	20	19	18—17	16	15	14	13	12
Name	RSVD	USB	SD/MMC	RSVD	SSP0	RSVD	PPI	RTC	KEYBD

Bit	11	10—9	8	7	6	5	4	3	2	1	0
Name	SIMI	RSVD	ASC1	ASC0	RSVD	PWM3	PWM2	PWM1	PMWT	PMIT	DMAC

Bit	Name	Description
31—21	RSVD	Reserved.
20	USB	Controls the clock to the USB block. If 1, the clock to the USB block is off. If 0, the clock to the USB block is on.
19	SD/MMC	Controls the clock to the SD/MMC card controller block.
18—17	RSVD	Reserved.
16	SSP0	Controls the clock to the CP-side SSP/I^2S (SSP0). If 1, the clock to the SSP0 is off. If 0, the clock to the SSP0 is on.
15	RSVD	Reserved.
14	PPI	Controls the clock to programmable peripheral interface signals 0—47. If 1, the clock to programmable peripheral interface 0—47 is off. If 0, the clock to programmable peripheral interface 0—47 is on.
13	RTC	Controls the clock to the RTC interface. If 1, the clock to the RTC interface is off. If 0, the clock to the RTC interface is on.
12	KEYBD	Controls the clock to the keyboard interface. If 1, the clock to the keyboard interface is off. If 0, the clock to the keyboard interface is on.
11	SIMI	Controls the clock to the SIMI interface. If 1, the clock to the SIMI interface is off. If 0, the clock to the SIMI interface is on.
10—9	RSVD	Reserved.
8	ASC1	Controls the clock to asynchronous serial communications channel 1. If 1, the clock to asynchronous serial communications channel 1 is off. If 0, the clock to asynchronous serial communications channel 1 is on.
7	ASC0	Controls the clock to asynchronous serial communications channel 0. If 1, the clock to asynchronous serial communications channel 0 is off. If 0, the clock to asynchronous serial communications channel 0 is on.
6	RSVD	Reserved.

7 Call Processor (CP) Block (continued)

7.2 Reset/Power/Clock Management (continued)

Table 7.2-4 Power Management Registers, Addresses (Clear 0x700C000C/Set 0x700C0008) (continued)

Bit	Name	Description
5	PWM3	Controls the clock to the PWM3 unit. If 1, the clock to the PWM3 unit is off. If 0, the clock to the PWM3 unit is on.
4	PWM2	Controls clock to the pulse-width modulator PWM2. If 1, the clock to PWM2 is off. If 0, the clock to PWM2 is on.
3	PWM1	Controls the clock to the pulse-width modulator PWM1. If 1, the clock to PWM1 is off. If 0, the clock to PWM1 is on.
2	PMWT	Controls the clock to the watchdog timer. The clock to the watchdog timer cannot be shut off when the watchdog mode is enabled. If 1, the clock to the watchdog timer is off. If 0, the clock to the watchdog timer is on.
1	PMIT	Controls the clock to the interval timer unit. If 1, the clock to the interval timer unit is off. If 0, the clock to the interval timer unit is on.
0	DMAC	Controls the clock to the CP-side DMA controller unit. If 1, the clock to the DMAC is off. If 0, the clock to the DMAC is on.

7 Call Processor (CP) Block (continued)

7.2 Reset/Power/Clock Management (continued)

7.2.4.4 Boot Select/ID Register (BOOTS_ID)

The boot select/ID register (see Table 7.2-5) contains the IDS bit and the ROM boot location bit. Upon an external pin reset, the reset value of bit 1 (BOOT) reflects the inverse of PIO35_A_A25_BOOTSEL pin input. This register is not affected by other types of resets. To specify boot location for a software reset or a watchdog timer reset, simply write the appropriate value to bit 1 (BOOT) prior to the execution of software reset or watchdog timer reset.

Table 7.2-5 Boot Select/ID Register (BOOTS_ID), Address (0x700C0010)

Bit	31—3	2	1	0
Name	RSVD	RSVD	BOOT	IDS

Bit	Name	Description
31—3	RSVD	Reserved.
2	RSVD	Reserved. Always write with 0.
1	BOOT	Specifies whether the core will boot from internal or external ROM. If 1, the core will boot from external ROM using CS0. If 0, the core will boot from internal ROM. This bit will reset to *n* after an external pin reset, where *n* is the inverse of PIO35_A_A25_BOOTSEL pin input during the external pin reset. (If PIO35_A_A25_BOOTSEL pin connects to nothing, the internal pull-up resistor will make this pin to appear as a 1 during the external pin reset, resulting in a boot from internal ROM.) This bit is unaffected by other resets.
0	IDS	Specifies if there is further system ID information. If 0, no further ID information is available. This bit is read-only. It always returns 0 when read. Writing 1 to this bit has no effect.

7 Call Processor (CP) Block (continued)

7.2 Reset/Power/Clock Management (continued)

7.2.4.5 Clock Status Register (CLKS)

The clock status register (see Table 7.2-6) indicates the current clock source for the system clock and the previous fast clock source.

Table 7.2-6 Clock Status Register (CLKS), Address (0x700C0014)

Bit	31—7	6—4	3—2	1—0
Name	RSVD	RSVDTP	PFSC	CSC

Bit	Name	Description
31—7	RSVD	Reserved.
6—4	RSVDTP	Reserved for test purposes: Bit 6 is 1 when the RTC is oscillating. Bit 6 is 0 when the RTC has no clock source. Bit 5 is reserved. Bit 4 is 1 when the PLL is locked. Bit 4 is 0 when the PLL is off or unlocked.
3—2	PFSC	Identifies the previous fast clock source (see Figure 7.2-1). Bit 3 is always 0. Bit 2 is 1 when the PLL clock was the last fast clock. Bit 2 is 0 when the external clock was the last fast clock.
1—0	CSC	Identifies the clock that is the current source of the system clock. See Table 7.2-7.

Table 7.2-7 System Clock Sources

Bits[1:0]	Description
00	External Clock
01	Phase-Locked Loop Clock
10	RTC
11	Reserved

7 Call Processor (CP) Block (continued)

7.2 Reset/Power/Clock Management (continued)

7.2.4.6 Clock Control Register (CLKC)

The clock control register (see Table 7.2-8) configures basic clock functions.

Table 7.2-8 Clock Control Register (CLKC), Address (0x700C0018)

Bit	31—7	6	5	4	3	2	1	0
Name	RSVD	SSBYP	RSVD	CKOEN	MS	PLLE	CKIOV	OFF

Bit	Name	Description
31—7	RSVD	Reserved.
6	SSBYP	Bypass the small-signal buffer. If 1, the small-signal buffer is bypassed. If 0, the small-signal buffer is active.
5	RSVD	Reserved.
4	CKOEN	*ARM* test clock output enable. If 1, enables the 1/2 frequency of *ARM* system clock to be sent toward CPTSTSTOP_CKO. If 0, constant logic level is sent toward CPTSTSTOP_CKO. Note that actual test clock output at CPTSTSTOP_CKO pin depends on the setting of CKOEN, UPLL2CKO, and ALTPINC[11] bits. See Figure 5.2-2 for further detail.
3	MS	Enables the manual mode. If 1, the clock switching is completed by software intervention. If 0, the clock switching is completed automatically.
2	PLLE	Enables the PLL, when in manual mode. If 1, the PLL is enabled. If 0, the PLL is disabled.
1	CKIOV	CKI override. If 1, CKI never turns off. If 0, CKI turns off when not needed by the clock switch logic.
0	OFF	CLKOFF mode. Determines if the CLKOFF mode feature is active. (CLKOFF mode shuts off all of the clocks to the core and peripherals when in WFI mode.) If 1, CLKOFF mode is active. If 0, CLKOFF mode is not active.

7.2.4.7 Soft Reset Register (SOFTRST)

The soft reset register address (see Table 7.2-9), when written, causes a software reset of CP block to occur. When read, all 0s will be returned. This register does not reset T8307 DSP block.

Table 7.2-9 Soft Reset Register (SOFTRST), Address (0x700C0020)

Bit	31—1	0
Name	RSVD	SFT

Bit	Name	Description
31—1	RSVD	Reserved.
0	SFT	Soft reset.

7 Call Processor (CP) Block (continued)

7.2 Reset/Power/Clock Management (continued)

7.2.4.8 PLL Control Register (PLLCR)

The PLL control register (see Table 7.2-10) configures the PLL.

Table 7.2-10 PLL Control Register (PLLCR), Address (0x700C0024)

Bit	31—18	17	16—15	14—12	11—9	8—6	5—0
Name	RSVD	PLLBP	RSVD	P	BITS	N	M

Bit	Name	Description
31—18	RSVD	Reserved.
17	PLLBP	Bypassing PLL. If 1, PLL output is bypassed and the small-signal buffer output is used whenever PLL output is requested. If 0, PLL output is not bypassed.
16—15	RSVD	Reserved for internal use—write with 0.
14—12	P*	PLL VCO frequency postdivider for *ARM* clock. Divides VCO frequency by (P + 1).
11—9	BITS	PLL trim control. This field must be programmed to 0x6 (i.e., 110).
8—6	N	PLL reference frequency predivider. Divides input reference by (N + 1).
5—0	M	PLL VCO frequency multiplier. Multiplies VCO frequency by (M + 1).

* For proper initialization of divider logic, make sure P is odd so that P+1 is even.

7.2.4.9 Reset Status Register (RSTS, RSTSC)

The reset status register (see Table 7.2-11) determines the source of the last chip reset. The register bits only get cleared by writing a 1 to the specified bit when addressing the reset status clear address (RSTSC), or for WR and SFT when an external reset occurs. Reading either RSTS or RSTSC returns current reset status register value.

Table 7.2-11 Reset Status Registers, Addresses (RSTS 0x700C0030, RSTSC 0x700C0034)

Bit	31—4	3	2	1	0
Name	RSVD	SFT	WR	RSVD	ER

Bit	Name	Description
31—4	RSVD	Reserved.
3	SFT	Identifies the last reset as a soft reset. If 1, a soft reset has occurred. If 0, the last reset was not a soft reset or the bit was cleared.
2	WR	Identifies the last reset as a warm reset (caused by watchdog timer). If 1, then a warm reset has occurred. If 0, the last reset was not a warm reset or the bit was cleared.
1	RSVD	Reserved.
0	ER	Identifies the last reset as an external reset. If 1, then an external reset has occurred. If 0, the last reset was not an external reset or the bit was cleared.

Note: The register bits only get cleared by writing a 1 to the specified bit when addressing the clear address, or for WR and SFT when an external reset occurs.

7 Call Processor (CP) Block (continued)

7.2 Reset/Power/Clock Management (continued)

7.2.4.10 System Clock Enable Register (SCLKEN)

This register enables or disables the different ways to request the external clock (CKI).

Table 7.2-12 System Clock Enable Register (SCLKEN), Address (0x700C002C)

Bit	31—3	2	1	0
Name	RSVD	WAKEUP_COUNTER_ENA	INT_ENA	FORCE_SYSCLKREQ_ON

Bit	Name	Description
31—3	RSVD	Reserved.
2	WAKEUP_COUNTER_ENA	The wake-up counter is clocked by the RTC clock and is programmable. The counter is an 8-bit synchronous counter. This bit resets to 0. If 1, enables the wake-up time-out counter from generating the SYSCLKREQ signal. If 0, disables the wake-up time-out counter from generating the SYSCLKREQ signal.
1	INT_ENA	Enables the interrupts from generating the SYSCLKREQ signal. If 1, enables this function. If 0, disables this function.
0	FORCE_SYSCLKREQ_ON	This bit forces the SYSCLKREQ to stay on. If 1, enables this function. If 0, disables this function.

The signal SYSCLKREQ is intended to connect to XOENAQ in CSP8307, which generates the signal XOENA to control the external crystal oscillator.

7 Call Processor (CP) Block (continued)

7.2 Reset/Power/Clock Management (continued)

7.2.4.11 Wake-Up Time-Out Register (WUTO)

The wake-up time-out register is the value that is loaded into the wake-up time-out counter. The wake-up time-out value decrements down to 0. If it reaches 0 before any other source sets the SYSCLKREQ to high, then it will set it to high, allowing the wait for clock timer to kickoff. If that value decrements to 0 without any other sources setting the SYSCLKREQ to high, then the system goes back into sleep mode again. The process then begins again.

Table 7.2-13 Wake-Up Time-Out Register (WUTO), Address (0x700C004C)

Bit	31—8	7—0
Name	RSVD	WTV

Bit	Name	Description
31—8	RSVD	Reserved.
7—0	WTV	This value gets loaded into the wake-up time-out counter, and then gets decremented down to 0. If it reaches 0 before any other source sets the SYSCLKREQ to high, then it will set the SYSCLKREQ to high.

7.2.4.12 Wait for Clock Time-Out Register (WFCTO)

The wait for clock time-out register is the value that is loaded into the wait for clock time-out counter. Once the SYSCLKREQ is set, the state machine jumps to the wait-state, loads this value, then decrements down to 0. Once it reaches 0, the last fast clock gets selected for the system clock.

Table 7.2-14 Wait for Clock Time-Out Register (WFCTO), Address (0x700C0050)

Bit	31—11	10—0
Name	RSVD	WCT

Bit	Name	Description
31—11	RSVD	Reserved.
10—0	WCT	This value gets loaded into the wait for clock time-out counter, and then gets decremented down to 0.

7.2.4.13 Keyboard Bounce Timer Control Register (KBTC)

The keyboard bounce timer control register is used to divide the RTC clock so that the debounce logic uses a lower frequency clock to reduce the effect of the keys repeatedly bouncing.

Table 7.2-15 Keyboard Bounce Timer Control Register (KBTC), Address (0x700C0054)

Bit	31—17	16	15—0
Name	RSVD	KONTON	KEYLD

Bit	Name	Description
31—17	RSVD	Reserved.
16	KONTON	If 0, keyboard bounce counter is disabled. If 1, keyboard bounce counter is enabled.
15—0	KEYLD	This value is loaded into the keyboard bounce counter, which decrements to 0. A value of 0 allows the RTC clock to go directly to the debounce logic. A value of *n* divides the RTC clock by 2 * n prior to going to the debounce logic, where *n* = 0x1—0xFFFF.

7 Call Processor (CP) Block (continued)

7.2 Reset/Power/Clock Management (continued)

7.2.4.14 Reset Extend Register (RSTEXT)

To prevent the CPU from starting to fetch prematurely when the external flash device is still in reset state, the internal reset pulse, regardless of the source of reset, will be N + 2 peripheral clock cycles longer than the reset pulse appearing at the FLASHRSTN pin, where N is programmable through the 16-bit reset extend register (see Table 7.2-16). Note that N must be a nonzero value.

Table 7.2-16 Reset Extend Register (RSTEXT), Address (0x700C0028)

Bit	31—16	15—0
Name	RSVD	REV

Bit	Name	Description
31—16	RSVD	Reserved.
15—0	REV	Number of peripheral clock cycles from the deassertion of FLASHRSTN to the deassertion of internal reset. All 1 for maximum count. 0x0001 for minimum count. Default value after external reset is 0x32C9 (13001 decimal). This register is not affected by a software or a watchdog timer reset.

The dedicated FLASHRSTN output pin is available to provide an active-low output signal for resetting an external flash device, if needed. This reset signal is generated by the RESETN pin reset, watchdog timer reset, or soft reset.

When RESETN pin reset happens, the same reset pulse appears at the FLASHRSTN pin.

If watchdog timer reset or soft reset are the source of the reset, then the reset pulse that appears at the FLASHRSTN pin is 130000 peripheral clock cycles.

Figure 7.2-3 summarizes the reset timing relationship.

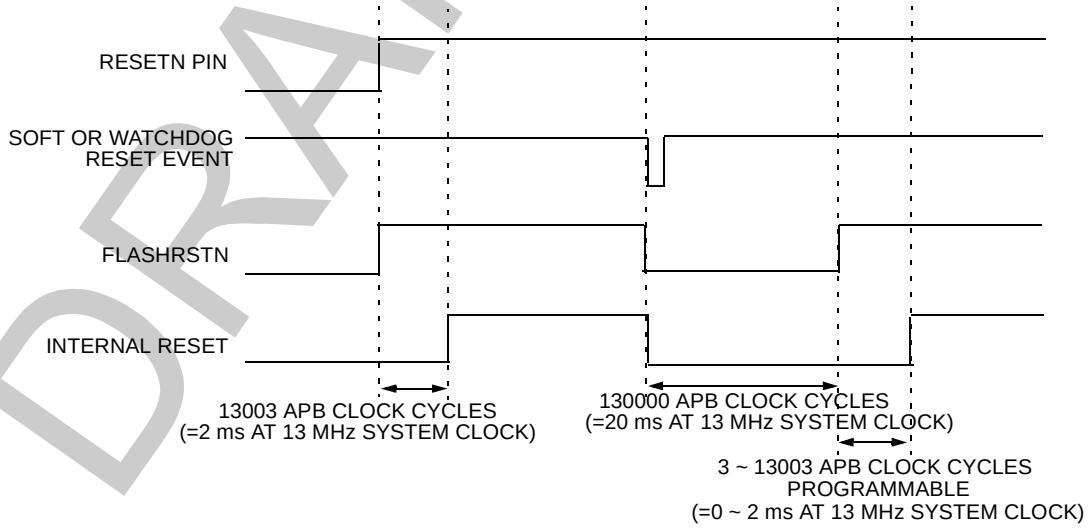


Figure 7.2-3 Reset Timing Relationship

7 Call Processor (CP) Block (continued)

7.2 Reset/Power/Clock Management (continued)

7.2.4.15 USB Firmware Control Register (USBFWC)

The USB firmware control register is used to control the software reset, clock switching, and powerdown mode of the USB interface. When the set address of this register is written, a 1 in the data field indicates the corresponding register bit will be set. When the clear address of this register is written, a 1 in the data field indicates the corresponding register bit will be cleared. A bit 0 in the data field written to either the set or clear address has no effect. Reading the set or the clear address returns the current USBFWC register value.

Table 7.2-17 USB Firmware Control Register (USBFWC), Addresses (Set 0x700C0038/Clear 0x700C003C)

Bit	31—7	6	5—4	3	2	1	0
Name	RSVD	FW_XCVR_SUSP	RSVD	FW_RWUPN	FW_UCORE_RST	RSVD	FW_UCORE_ON

Bit	Name	Description
31—7	RSVD	Reserved.
6	FW_XCVR_SUSP	USB external transceiver suspension. 1: Forces the external USB transceiver into low-power standby mode. This will override the control of USB_SUSP pin from USB device controller. 0: Puts the external USB transceiver into normal working mode (i.e., USB_SUSP pin under the control of USB device controller). Reset value of this bit is 1.
5—4	RSVD	Reserved.
3	FW_RWUPN	Firmware-controlled USB remote wakeup. 1: Sends a firmware wakeup to USB core to execute a remote wakeup sequence. 0: Do nothing. Reset value of this bit is 0.
2	FW_UCORE_RST	USB device controller software reset. 1: Puts the USB device controller in reset state. 0: Does not put the USB device controller in reset state. This bit does not clear itself automatically when set. The user must clear it with a second write to USBFWC clear address. To execute internal hardware resets reliably, allow several cycles in-between the two writes. This bit resets to 1. T8307 firmware must write 0 to FW_UCORE_RST bit before the USB device controller can be used.
1	RSVD	Reserved.
0	FW_UCORE_ON	USB core enable. 1: Puts the USB core in normal working mode. 0: Puts the USB core in low power standby mode.

7 Call Processor (CP) Block (continued)

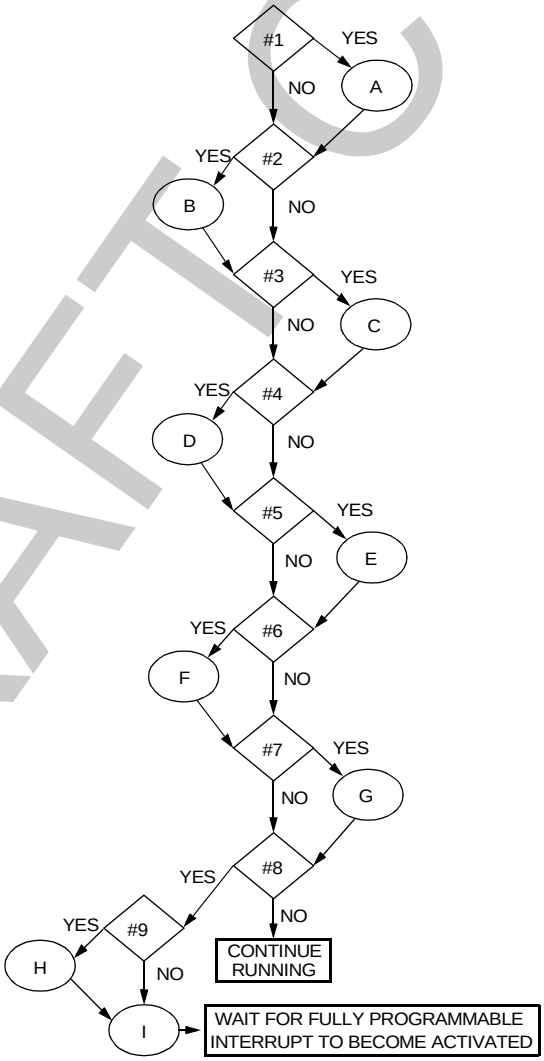
7.2 Reset/Power/Clock Management (continued)

7.2.5 Operation on Reset

Upon all RESETs, the reset, power, and clock management unit performs the following:

- All power management register bits are set to zero, which enables the clocks to all units. The PLL is disabled.
- All status register bits are reset to zero, except the reset status register (see note for Table 7.2-11).
- The source clock is set to the external clock.
- The appropriate status bit is set to one in the reset status register.
- All 10-bit reset extend register bits are reset to one by the RESETN pin reset and will not be reset by watchdog timer reset or soft reset.

7.2.6 Flowchart



1041 (F).e

Figure 7.2-4 Flowchart

7 Call Processor (CP) Block (continued)

7.2 Reset/Power/Clock Management (continued)

The following steps will guide the user through the process of setting up the clock switching registers. The step letters and numbers correspond to the flowchart in Figure 7.2-4.

1. Do you want to program what interrupt will cause slow to fast switching of the source clock?

   A. Write a 1 to each bit that is associated with each interrupt in the slow to fast interrupt register (address 0x700C10CC) that you want to cause a slow to fast clock switch.

2. Do you want to delay the slow to fast clock switching, because the external clock needs to stabilize before switching over to it?

   B. Write the value that you want to count down from before switching over to the external clock into the wait for clock time-out register (address 0x700C0050).

3. Do you want a periodically wake-up of the system, which looks for an interrupt during wait for clock time and, if no interrupt occurs during that time, it will go back to sleep?

   C. Write the value that you want to count down from before waking up into the wake-up time-out register (address 0x700C004C) and enable this feature by writing to the system clock request enable register bit 2 WAKEUP_COUNTER_ENA bit (address 0x700C002C).

4. Do you want to manually control the PLL?

   D. Write bit 3 (MANUAL bit) to a 1 in the clock control register (address 0x700C0018) plus the PLL enable bit for the PLL (bit 2) to turn it on or off.

5. Do you want to bypass the wait for clock time on some of the interrupts?

   E. Write a 1 to each bit that is associated with each interrupt in the bypass wait for clock interrupt register (address 0x700C10D4) that you want to cause the system to bypass the wait for clock counter prior to switching to the last fast clock.

6. Do you want to keep CKI on, even if you are in a slow clock source and in automatic mode?

   F. Write bit 1 (CKI OVERRIDE bit) to a 1 in the clock control register (address 0x700C0018).

7. Do you want to switch to another clock source?

   G. Write the appropriate bit in the clock management register (address 0x700C0004) to a 1.

Note: If in manual mode, make sure the destination clock source is running.

8. Do you want to go into WFI mode?

9. Do you want to go into clock off mode?

   H. Write bit 0 (CLOCK OFF MODE bit) to a 1 in the clock control register (address 0x700C0018).

Note: A fully programmable interrupt must be enabled and set up for asynchronous operation prior to setting this bit.

   I. Write bit 0 (PAUSE bit) to a 1 in the pause register (address 0x700C0000).

7 Call Processor (CP) Block (continued)

Section 7.3 through Section 7.4 detail the CP-system bus functions.

7.3 Static Memory Controller (SMC)

SMC stands for static memory controller. It functions as the CP block external memory interface. SMC is implemented based on the *ARM PrimeCell Static Memory Controller*. The SMC is the interface to Agere's 802.11 media access controller or radio module.

7.3.1 Operation

This section describes the operation of the SMC, including the timing of standard transfers for different memory types and externally waited transfers, and example configurations for different memory device and bank sizes. The functions of SMC are described under the following headings:

- Memory bank select.
- Access sequencing and memory width.
- Wait-state generation.
- Write protection.
- Static memory read control.
- Static memory write control.
- Byte lane control.

7.3.1.1 Memory Bank Select

Eight independently configurable memory banks are supported, with a separate chip select output for each bank. The chip select lines A_CS[7:0]N for all banks are configurable to be either active-high or active-low (default). Table 7.3-1 shows the address mapping for external memory banks where the base address is defined in Section 6.1.

Table 7.3-1 Address Mapping for External Memory Banks

31—29	28—26	25—0
Base address for SMC memory	Chip select address space for eight memory banks (A_CS[7:0]N)	64 MB memory banks address space (A_A[25:0])

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

7.3.1.2 Access Sequencing and Memory Width

The data width of each external memory bank must be configured by programming the appropriate bank configuration register SMBCRx. When the external memory bus is narrower than the transfer initiated from the current AMBA bus master, the internal bus transfer takes several external bus transfers to complete. For example, in the case that bank 0 is configured as an 8-bit wide memory and a 32-bit read is initiated, the AMBA AHB bus stalls while the SMC reads four consecutive bytes from the memory. During these accesses, the data path is controlled (in the external memory data path logic) to demultiplex the 4 bytes into one 32-bit word on the AMBA AHB bus.

The access sequencing supports only little-endian operation.

7.3.1.3 Wait-State Generation

Each bank of the SMC must be configured for external transfer wait-states in read and write accesses. This is achieved by programming the appropriate fields of the bank control registers SMBIDCYRx, SMBWST1Rx, and SMBWST2Rx. The number of cycles in which an AMBA transfer completes is controlled by three other factors:

- Access width.
- External memory width.
- External wait input.

Each bank of the SMC has a programmable enable for the external wait (WaitEn), and a programmable polarity setting (WaitPol), allowing full configuration of the external wait for each bank.

The WST1 wait-state field can be programmed to select up to 31 wait-states for read memory accesses to SRAM and ROM, or the initial burst read access to burst ROM.

The WST2 wait-state field can be programmed to select up to 31 wait-states for write access to SRAM or burst mode reads from burst ROM devices. For example, the configuration for an access to a burst ROM with a 120 ns initial access time followed by a 60 ns burst access time, using a 100 MHz system clock would be 12 wait-states for the first access and 6 wait-states for the subsequent accesses.

7.3.1.4 Write Protection

Each memory bank can be configured for write protection. Normally SRAM is unprotected and ROM devices must be write-protected, but the WP field in the bank configuration registers SMCBCRx can be set to write protect SRAM as well as ROM devices.

If a write access is made to a write protected memory bank, the WriteProtErr bit of the status register is asserted. If a write access is made to a memory bank containing ROM devices and the bank is not write protected, there is no error indication returned.

7.3.1.5 Static Memory Read Control

The static memory read controls are described in the following headings:

- Output enable programmable delay.
- Output enable deassertion to chip select deassertion programmable delay.
- ROM, SRAM, and flash.
- Burst ROM.
- Burst flash.

Output Enable Programmable Delay

The delay between the assertion of the chip select and the output enable is programmable from 0 to 15 cycles using the WSTOEN bits of the bank control registers. This delay is used to reduce the power consumption for memories that are not able to provide valid output data immediately after the chip select is asserted. If the output of the device is enabled before the final read data value is ready, the device drives out two different values, one unknown value, followed by the valid read data. This consumes more power than just driving out the final read data value. The output enable is deasserted at the same time as the chip select, at the end of the transfer, if the WST2OEN bits of the bank control registers are programmed to zero.

Note: The WSTOEN programmed value must be equal to, or less than, the WST1 programmed value, as the access is timed by the wait-states and not by the WSTOEN value. In the external wait enabled mode, the timing of the transfer (controlled by the PIO30_WAITN pin) is not known, so A_OEN is asserted along with A_CS[x]N.

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Output Enable Deassertion to Chip Select Deassertion Programmable Delay

The delay between the deassertion of the output enable and deassertion of the chip select is programmable from 0 to 15 cycles using the WST2OEN bits of the bank control registers. This delay is used to provide additional hold time for memories and devices that require output enable to be deasserted before chip select. The output enable is deasserted at the same time as the chip select, at the end of the transfer, if the WST2OEN bits of the bank control registers are programmed to zero.

Note: The A_CS[x]N assertion period is extended by WST2OEN cycles beyond the WST1/WST2 dictated read cycles. This is significantly different from the case of WSTOEN.

Note: The A_CS[x]N assertion period is extended by WST2OEN cycles, and the A_OEN is deasserted by WST2OEN cycles, for EACH read access within a burst.

ROM, SRAM, and Flash

The SMC uses the same read timing control for ROM, SRAM, and flash devices. Each read starts with the assertion of the appropriate memory bank chip select signals A_CS[x]N and memory address A_A[25:0]. The read access time is determined by the number of wait-states programmed for the WST1 field of the bank control register SMBWST1Rx. The IDCY field in the idle cycle control register SMBIDCYRx determines the number of bus turnaround wait-states added between external read and write transfers.

Figure 7.3-1 shows an external memory read transfer with the minimum zero wait-states (WST1 = 0), and the minimum zero output enable delay states (WSTOEN = 0). A minimum of two AHB wait-states are inserted during all single read transfers.

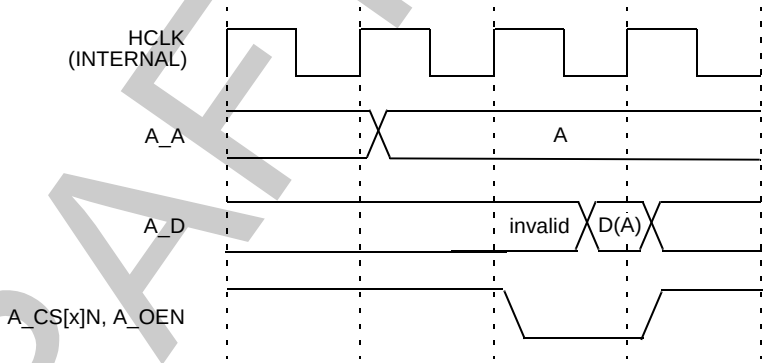


Figure 7.3-1 External Memory Zero Wait-States Read Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-2 shows an external memory read transfer with two wait-states (WST1 = 2), and the minimum zero output enable delay states (WSTOEN = 0). Four AHB wait-states are inserted during the transfer, two for the standard read, and an additional two due to the programmed wait-states added.

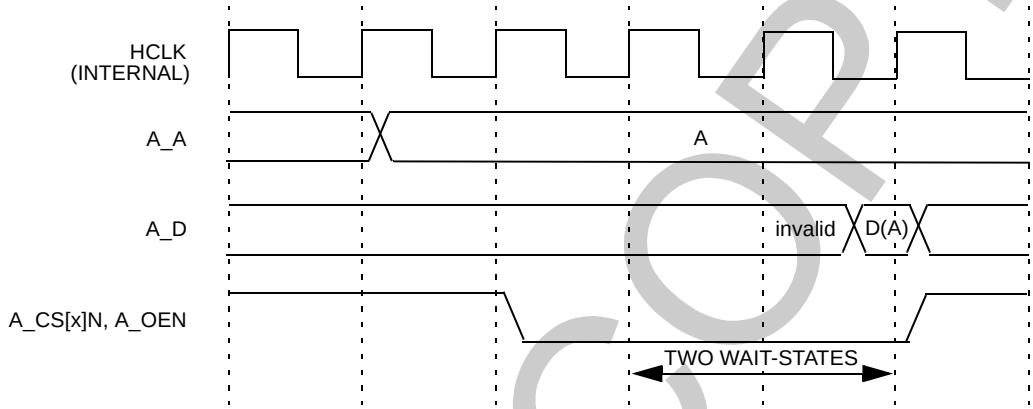


Figure 7.3-2 External Memory Two Wait-States Read Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-3 shows an external memory read transfer with two output enable delay states (WSTOEN = 2) and two wait-states (WST1 = 2). Four AHB wait-states are inserted during the transfer, two for the standard read, and an additional two due to the output enable delay states added.

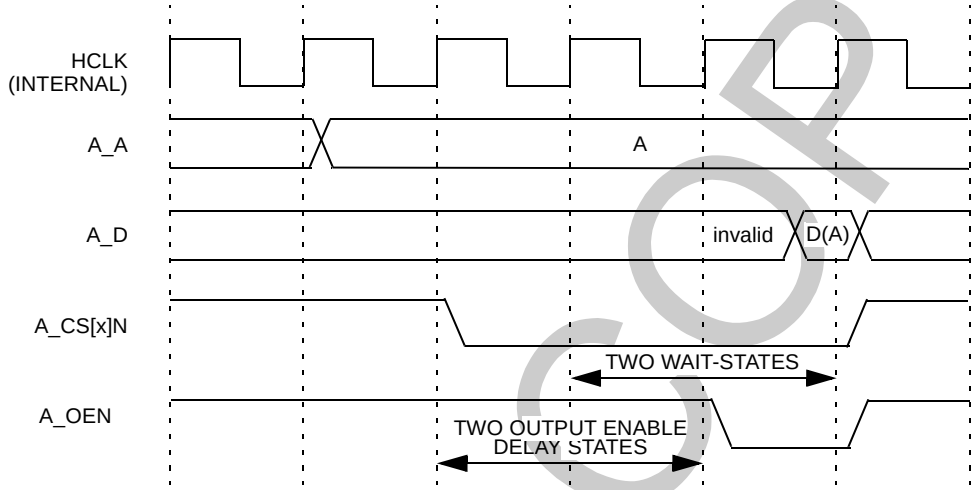


Figure 7.3-3 External Memory Two Output Enable Delays and Two Wait-States Read Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-4 shows an external memory read transfer with one output enable deassertion to chip select deassertion delay state (WST2OEN = 1) and one wait-state (WST1 = 1). Four AHB wait-states are inserted during the transfer, two for the standard read, and an additional two due to WST1 + WST2OEN delay states added.

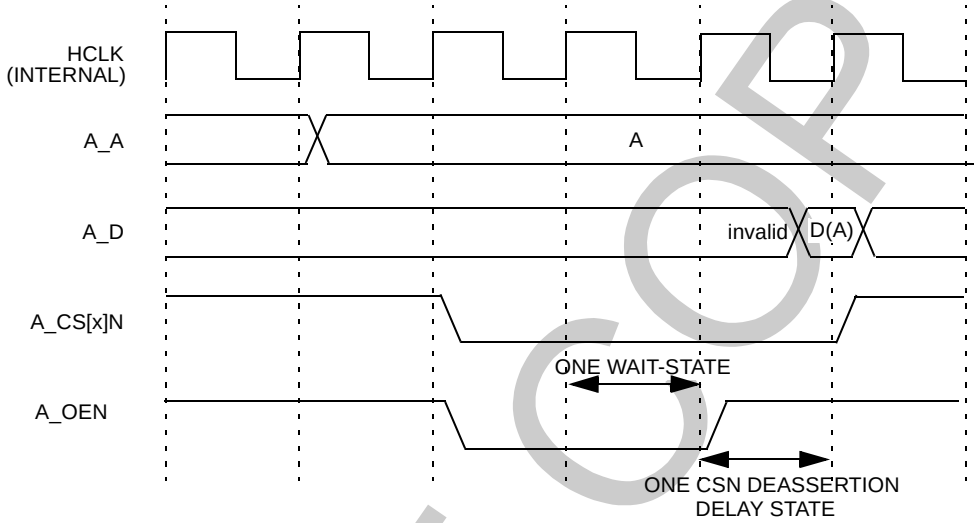


Figure 7.3-4 External Memory One Output Enable Deassertion to Chip Select Deassertion Delay and One Wait-State Read Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-5 shows an external memory read transfer with the minimum zero wait-states where the SMC does not have control of the bus and must request for it. In this example nothing else is requesting the bus, so the SMC is granted straight away, showing the minimum timing when the bus is requested.

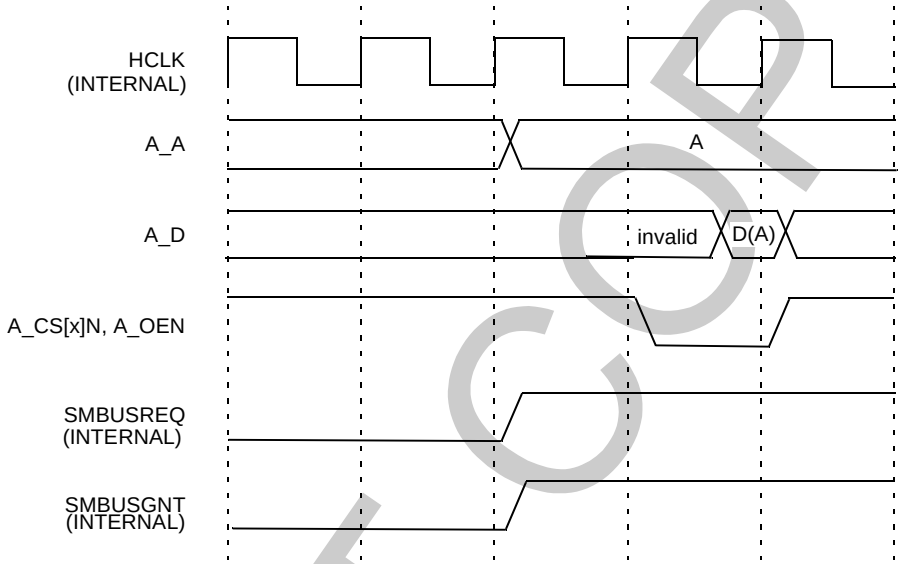


Figure 7.3-5 External Memory Zero Wait-States Read When Not Granted the Bus Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-6 shows external memory read transfers with zero wait-states (WST1 = 0). These might be nonsequential transfers, or sequential transfers of unspecified burst length. All transfers are treated as separate reads, so they have the minimum of two AHB wait-states added.

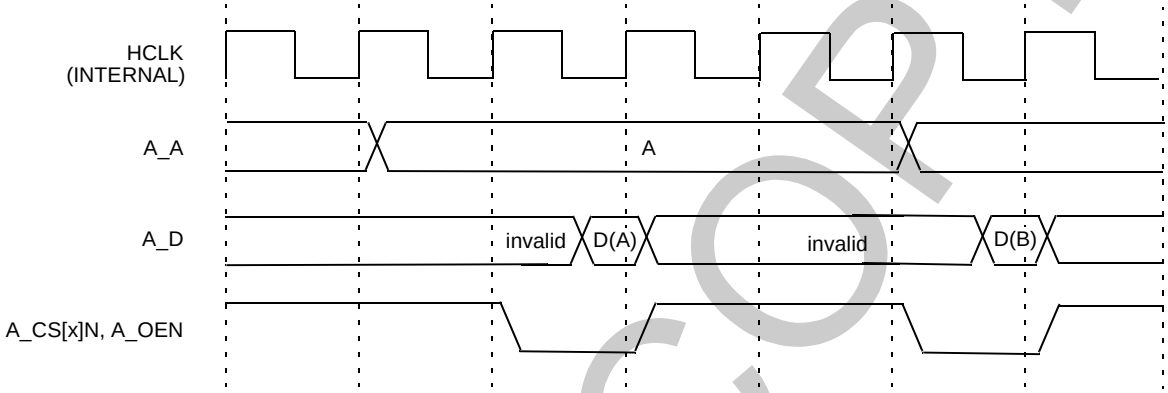


Figure 7.3-6 External Memory Three Zero Wait-States Read Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-7 shows a burst of zero wait-state reads with the length specified. As the length of the burst is known, it is possible to hold the chip select asserted during the whole burst, and generate the external transfers before the current AHB transfer has completed. Therefore, the first read has two AHB wait-states added, and the three following sequential reads have zero AHB wait-states added due to the automatic generation of the external transfers.

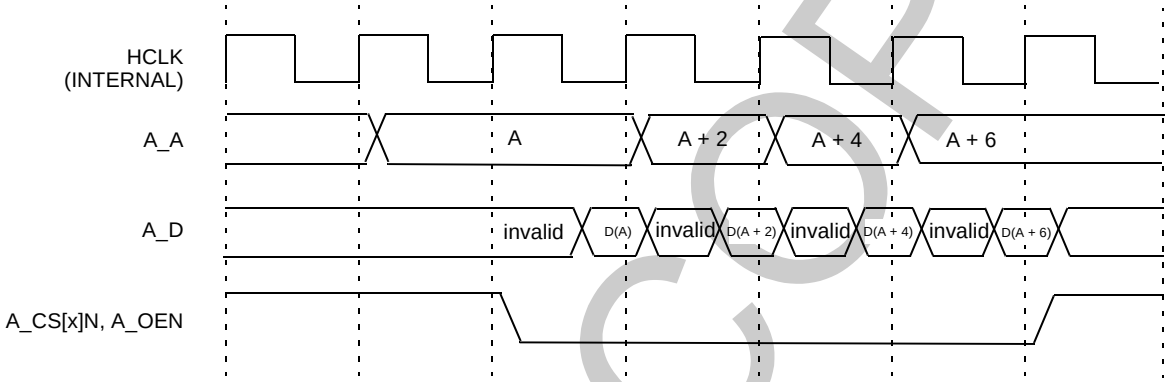


Figure 7.3-7 External Memory Zero Wait-States Fixed-Length Read Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-8 shows a burst of two wait-state reads with the length specified. The WST1 value is used for all transfers in the burst, with the first read having four AHB wait-states inserted, and all sequential transfers having two AHB wait-states.

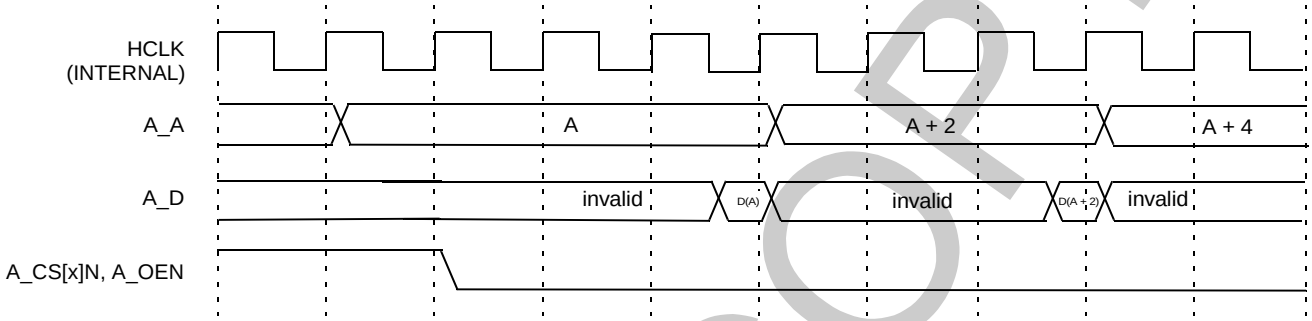


Figure 7.3-8 External Memory Two Wait-States Fixed-Length Burst Read Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Burst ROM

The SMC implemented in T8307 supports sequential access burst reads to a maximum of four consecutive locations in 8-bit or 16-bit memories. This feature supports burst mode ROM devices and increases the bandwidth by using a reduced (configurable) access time for the sequential reads (WST2) following the first read (WST1). The chip select and output enable lines are held during the burst, and only the address changes between subsequent accesses. At the end of the burst, the chip select and output enable lines are deasserted together.

Note: Bursts cannot cross quad boundaries, which are:

- A_A[1:0] = 11 for 8-bit transfers.
- A_A[2:1] = 11 for 16-bit transfers.

They are split up so that the first transfer after the boundary uses the slow read (WST1) timing. For example, a 4-byte transfer starting at address
A_A[1:0] = 01 performs a slow read from address 01, two fast reads from 10 and 11, and then a final slow read from address 00 to finish the burst.
Figure 7.3-9 shows an external memory burst read transfer with two initial wait-states, and one sequential wait-state. The first read has four AHB wait-states inserted, and all additional sequential transfers have only one AHB wait-state. This gives increased performance over the equivalent nonburst ROM timing shown in Figure 7.3-8.

External burst transfers are always split up into bursts of maximum four transfers, with the first read using the slow timing and the subsequent three reads using the fast timing, due to the four transfer burst limit of burst ROM devices. This four transfer limit is only applied to the external transfers. An AHB burst with size greater than the external memory is split up into bursts of four external transfers, taking into account any quad boundary connections.

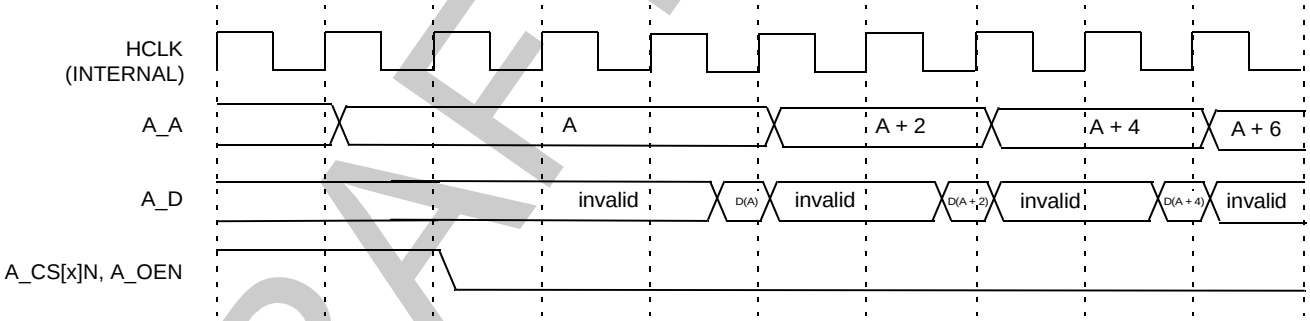


Figure 7.3-9 External Burst ROM WST1 = 2 and WST2 = 1 Fixed-Length Burst Read Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-10 shows a 32-bit read from an 8-bit burst ROM device, causing four burst reads to be performed. A total of five AHB wait-states are added during this transfer, two for the first external read, and then one for each of the subsequent reads.

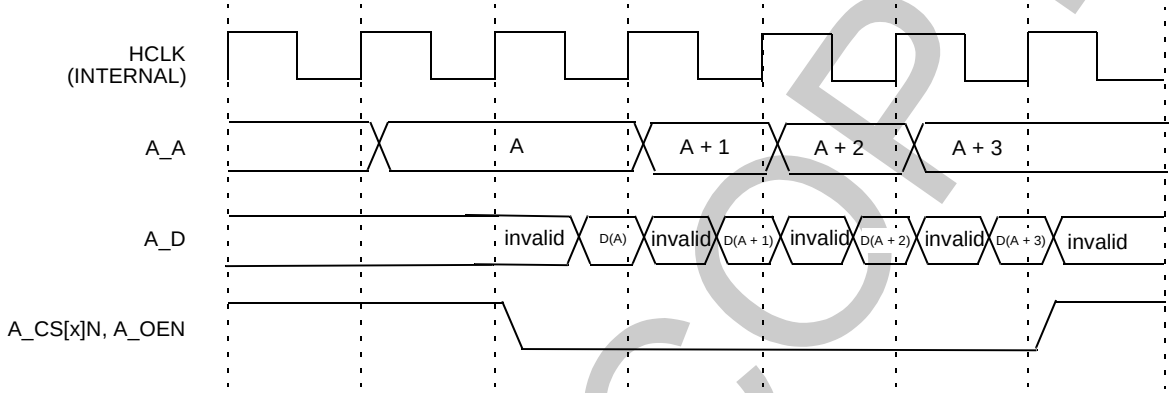


Figure 7.3-10 External Memory 32-Bit Burst Read from 8-Bit Memory Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Burst Flash

The SMC supports sequential access burst reads from burst flash devices, of the same types as for burst ROM. Due to the sharing of the WST2 register between write transfers and burst read transfers, it is only possible to have one setting at a time for burst flash, either the write delay or the burst read delay. This means that for a write transfer the WST2 register must be programmed with the write delay value, and for a burst read transfer the WST2 register must be programmed with the burst access delay value.

7.3.1.6 Static Memory Write Control

Write timing is described in the following sections:

- Write enable programmable delay.
- Write enable deassertion to chip select deassertion programmable delay.
- SRAM.
- Flash memory.

Write Enable Programmable Delay

The delay between the assertion of the chip select and the write enable is programmable from 0 to 15 cycles using the WSTWEN bits of the bank control registers. This delay is used to reduce the power consumption for memories. The write enable is asserted on the falling edge of HCLK (rising edge of nHCLK) after the assertion of the chip select for zero wait-states. The write enable is deasserted half a cycle before the chip select, at the end of the transfer, if the WST2WEN bits of the bank control registers are programmed to zero.

Note: The WSTWEN programmed value must be equal to, or less than the WST2 programmed value, as the access is timed by the wait-states and not by the WSTWEN value. In the external wait enabled mode, the timing of the transfer (controlled by PIO30_WAITN) is not known, so A_WEN is asserted immediately after A_CS[x]N.

Write Enable Deassertion to Chip Select Deassertion Programmable Delay

The delay between the deassertion of the write enable and deassertion of the chip select is programmable from 0 to 15 cycles using the WST2WEN bits of the bank control registers. This delay is used to provide additional hold time for memories and devices that require write enable to be deasserted before chip select. The write enable is deasserted half a cycle before the chip select, at the end of the transfer, if the WST2WEN bits of the bank control registers are programmed to zero.

Note: The A_CS[x]N assertion period is extended by WST2WEN cycles beyond the WST2 dictated write cycles. This is significantly different from the case of WSTWEN.

Byte Lane Enable

Byte lane enable signals A_BE0N and A_BE1N have the same timing as A_WEN for writes to 8-bit devices that use these pins instead of A_WEN. For 16-bit devices, A_BE0N and A_BE1N follow the timing of A_CS[x]N.

SRAM

Write timing for SRAM starts with assertion of the appropriate memory bank chip selects A_CS[x]N and address signals A_A[25:0]. The write access time is determined by the number of wait-states programmed for the WST2 field of the bank control register SMBWST2Rx. The IDCY field in the bank control register determines the number of bus turnaround wait-states added between external read and write transfers.

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-11 shows a single external memory write transfer with minimum zero wait-states (WST2 = 0), and the minimum zero write enable delay states (WSTWEN = 0). No AHB wait-states are added.

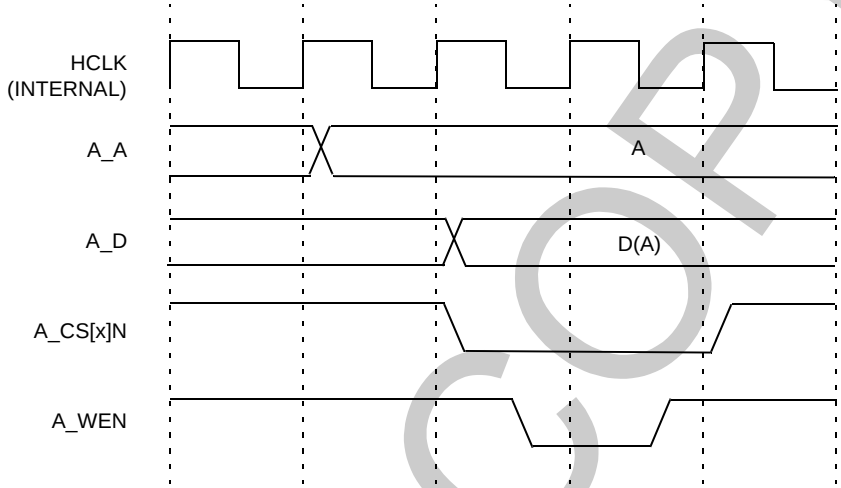


Figure 7.3-11 External Memory Zero Wait-States Write Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-12 shows a single external memory write transfer with two wait-states (WST2 = 2), and the minimum zero write enable delay states (WSTWEN = 0).

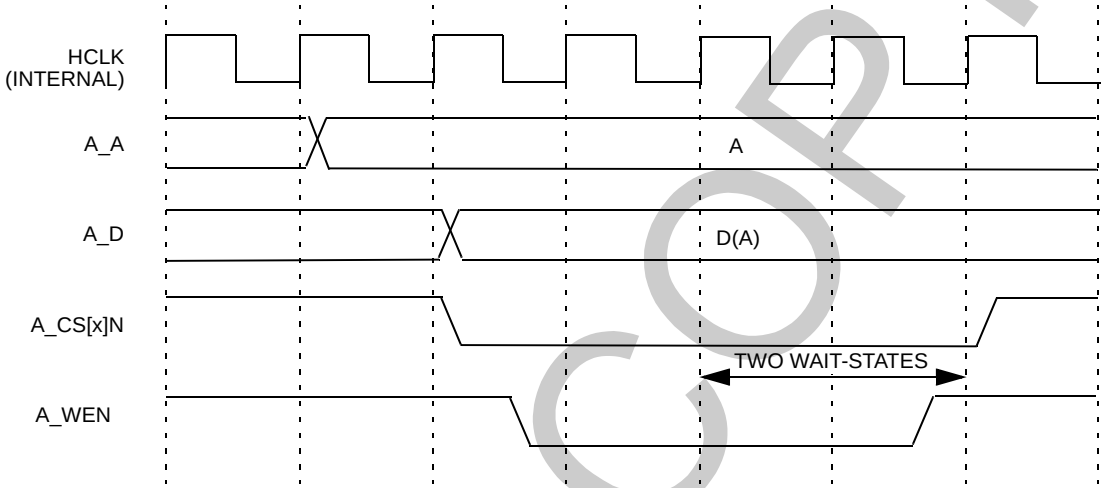


Figure 7.3-12 External Memory Two Wait-States Write Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-13 shows a single external memory write transfer with two write enable delay states (WSTWEN = 2) and two wait-states (WST2 = 2). No AHB wait-states are added.

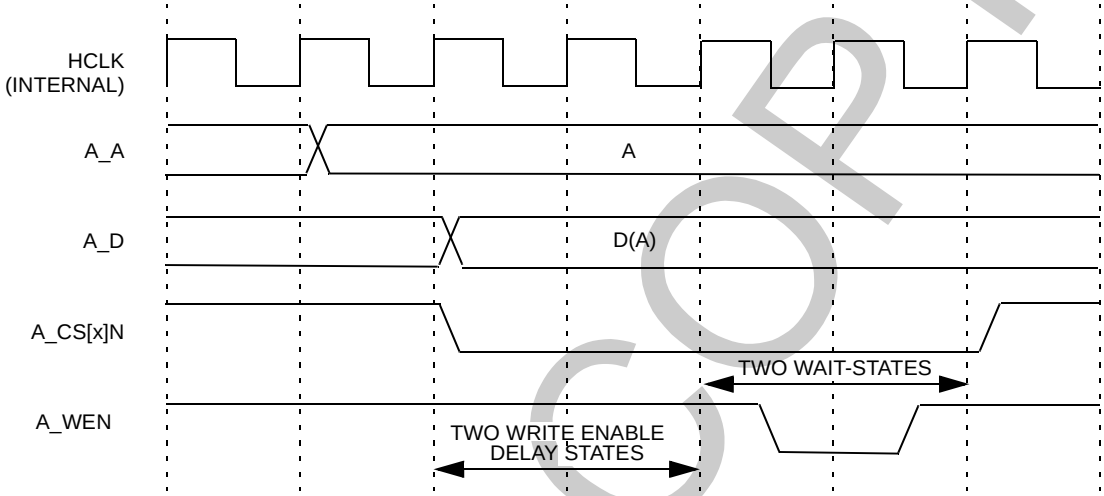


Figure 7.3-13 External Memory Two Write Enable Delays and Two Wait-States Write Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-14 shows a single external memory write transfer with one write enable deassertion to chip select deassertion delay state (WST2WEN = 1) and one wait-state (WST2 = 1). No AHB wait-states are added.

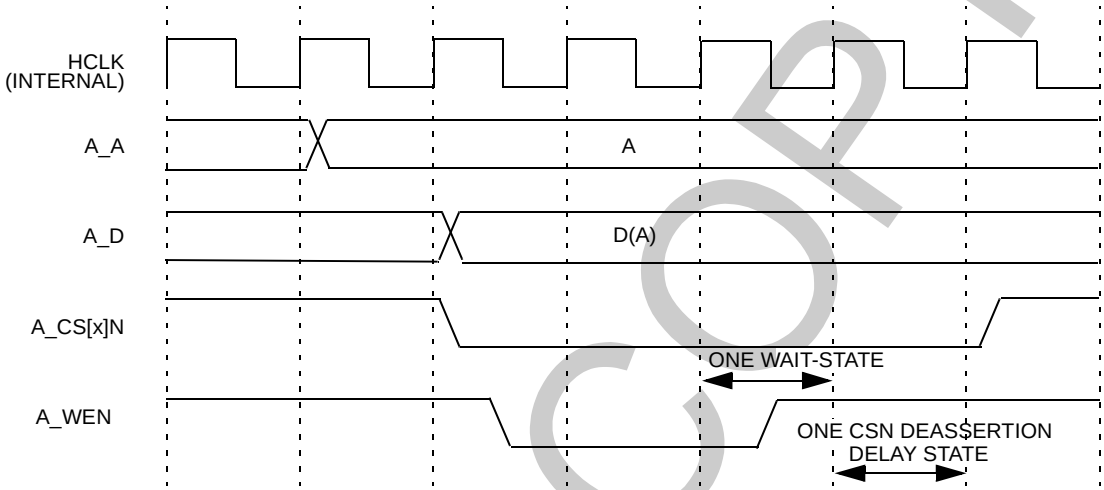


Figure 7.3-14 External Memory One Write Enable Deassertion to Chip Select Deassertion Delay and One Wait-State Write Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-15 shows a single external memory write transfer with minimum zero wait-states where the SMC does not have control of the bus and must request for it. In this example, nothing else is requesting the bus, so the SMC is granted straight away, showing the minimum timing when the bus is requested.

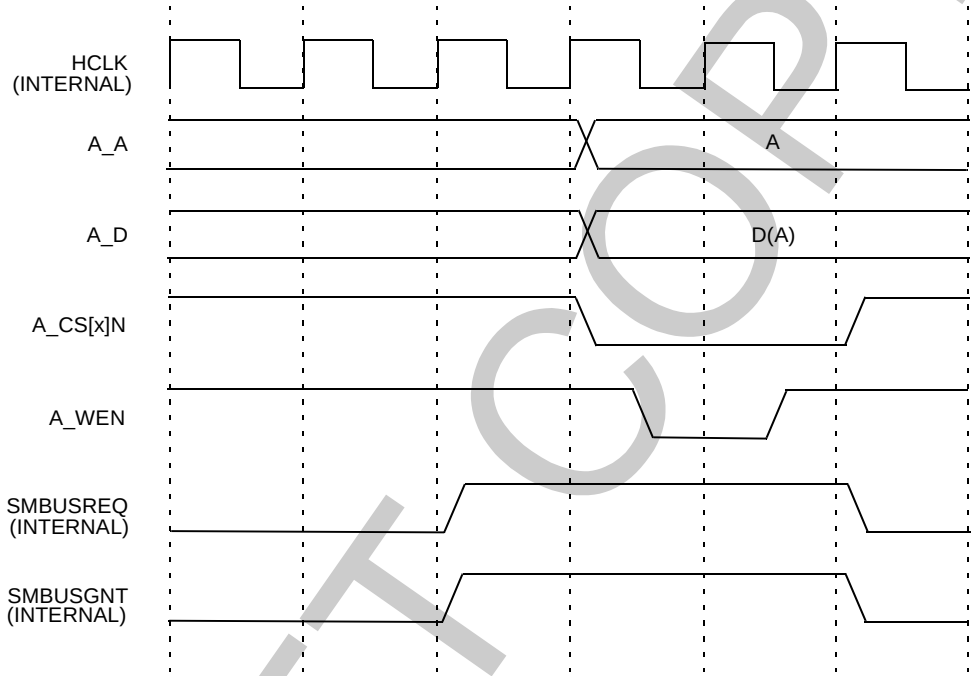


Figure 7.3-15 External Memory Zero Wait-States Write When Not Granted the Bus Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-16 shows two external memory write transfers with zero wait-states (WST2 = 0). AHB wait-states are added until the completion of the second write transfer. This is the timing of any sequence of write transfers:

- Nonsequential to nonsequential.
- Nonsequential to sequential with any value of HBURST (internal signals that indicate whether the transfer forms part of a burst).

The maximum speed of write transfers is controlled by the external timing of the write enable relative to the chip select. All external writes must take a minimum of two cycles to complete, the cycle that write enable is asserted, and the cycle that write enable is deasserted.

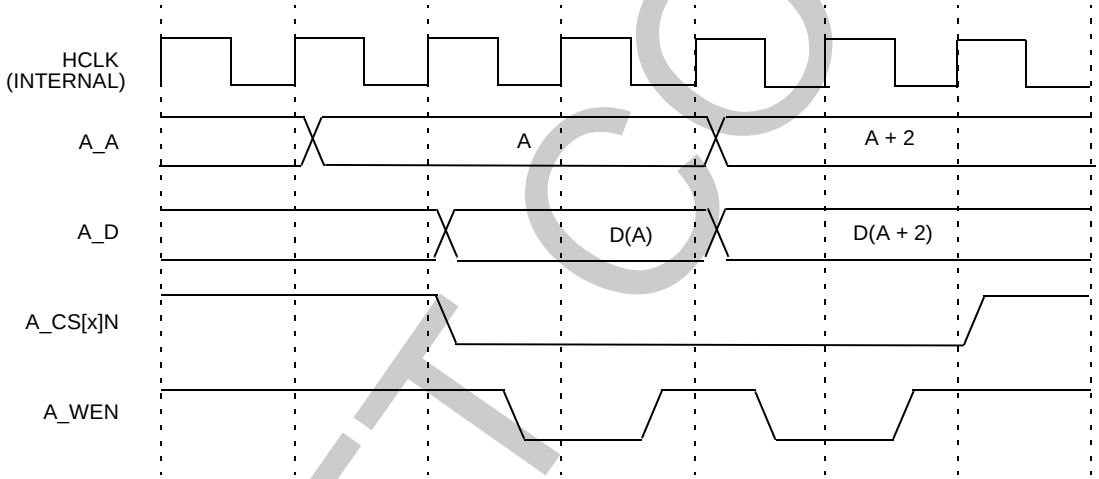


Figure 7.3-16 External Memory Two Zero Wait-Writes Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Flash Memory

Write timing for flash memory devices is the same as for SRAM devices.

7.3.1.7 Bus Turnaround

The SMC can be configured for each memory bank to use external bus turnaround cycles between read and write memory accesses. The IDCY field can be programmed for up to 15 bus turnaround wait-states. This is to avoid bus contention on the external memory data bus. Bus turnaround cycles are generated between external bus transfers as follows:

- Read-to-read, to different memory banks.
- Read-to-write, to the same memory bank.
- Read-to-write, to different memory banks.

Figure 7.3-17 shows a zero wait-read followed by a zero wait-write with default turnaround between the transfers of two cycles due to the timing of the AHB transfers. Standard AHB wait-states are added to the transfers, two for the read, and zero for the write.

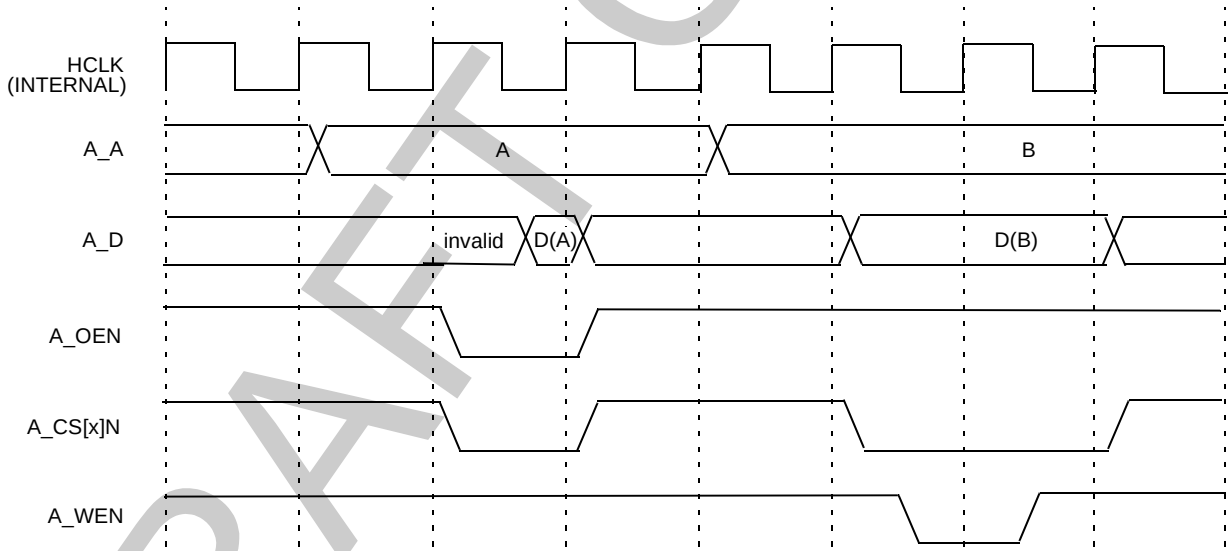


Figure 7.3-17 Read Followed by Write (Both Zero Wait-States) with No Turnaround Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-18 shows a zero wait-write followed by a zero wait-read with default turnaround between the transfers of one cycle. No AHB wait-states are added to the write transfer, but four are added to the read, two to allow the write to complete before the read is started, and then the standard two for the read transfer.

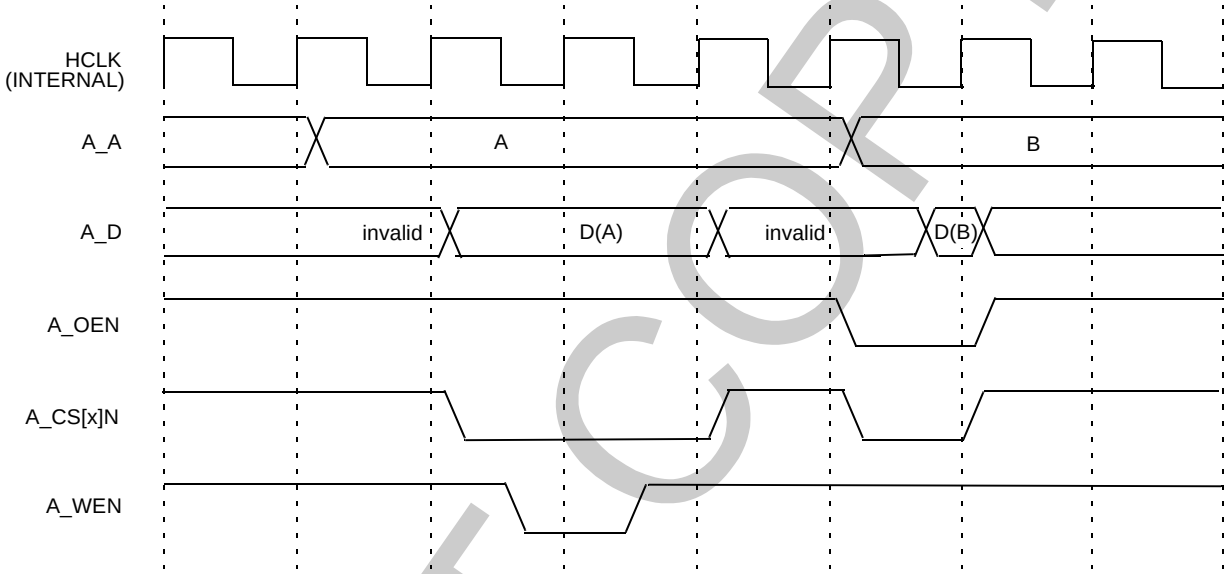


Figure 7.3-18 Write Followed by Read (Both Zero Wait-States) with No Turnaround

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-19 shows a zero wait-read followed by two zero wait-writes with two turnaround cycles added. The standard minimum of two AHB wait-states are added to the read transfer, and none are added to the first write (as for any read-write transfer sequence). Two AHB wait-states are added to the second write due to insertion of the two turnaround cycles that are only generated after the first write transfer is detected.

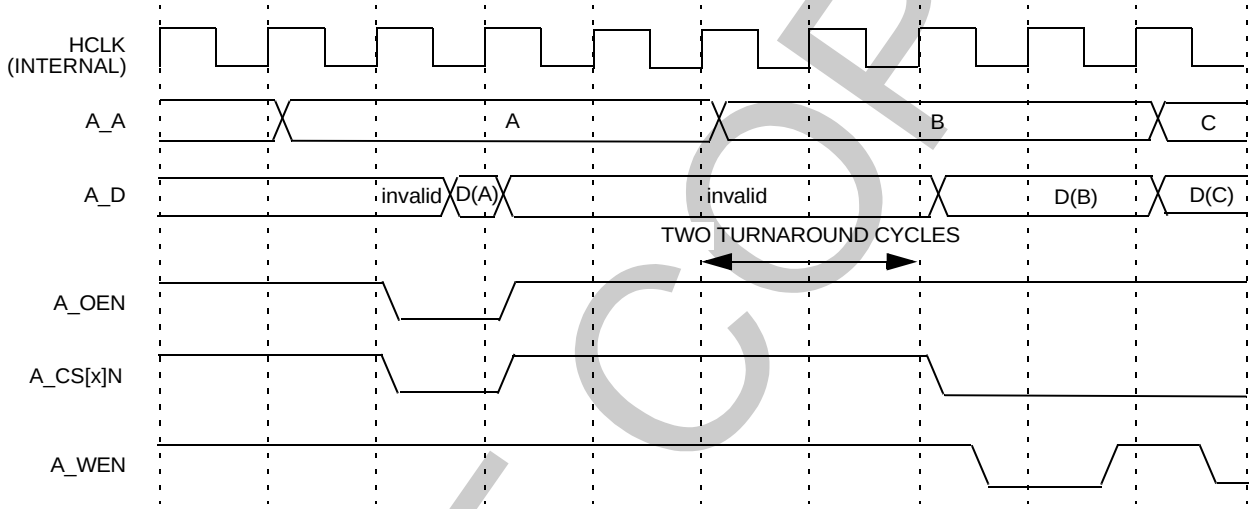


Figure 7.3-19 Read Followed by Two Writes (All Zero Wait-States) with Two Turnaround Cycles Timing Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

7.3.1.8 External Wait Control

The SMC supports the extension of the access cycle by an external device like a memory controller by using the PIO30_WAITN input pin. For this, the WaitEn bit of the bank control registers, SMBCRx must be programmed appropriately. The polarity of the external PIO30_WAITN input is programmed through the WaitPol field of the SMBCRx register.

Note: Since the external wait control input (PIO30_WAITN) is an asynchronous input, it is synchronized before use. This gives all operations using external waits a two-cycle delay due to the synchronization time.

When external wait mode is enabled, the SMC checks for assertion of the PIO30_WAITN input and waits the current transfer while PIO30_WAITN stays asserted. The transaction completes once the PIO30_WAITN line is deasserted (taking into account the two cycle synchronization delay).

If the external wait control mode is disabled, then the SMC ignores the PIO30_WAITN input and the access time is generated normally according to the values programmed in the WST1 and WST2 registers.

PIO30_WAITN Assertion Timing

In the wait-enabled or external-wait control mode, when the SMC is waiting for the PIO30_WAITN assertion, it also starts counting down according to the values programmed in the wait-state count field WST1 or WST2, that are used for read and write transfers, respectively. This feature is used to ensure that adequate time is available to the SMC to detect PIO30_WAITN, since there might be a delay before the external device asserts PIO30_WAITN. If PIO30_WAITN is not asserted during this time, the transfer is assumed to be zero wait.

PIO30_WAITN Deassertion Timing

A waited transfer only ends when the PIO30_WAITN input has been deasserted. If a waited transfer is terminated before it has completed successfully, then an AHB error response is generated, and the WaitToutErr flag in the bank status register is asserted. Since an external wait stops any external transfers being performed on the external bus, the SMC generates an error response when a transfer is requested to any external location and the external wait input is still asserted from a previous transfer. This continues until the waited transfer is complete (PIO30_WAITN deasserted), and then operation continues as normal.

The WaitStatus bit of the bank status register can be used to check the current status of PIO30_WAITN after a terminated transfer.

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

PIO30_WAITN Timing Diagrams

Figure 7.3-20 shows the timing for an externally waited read transfer, taking two cycles for the wait to be asserted, and two cycles for the wait to be deasserted. The synchronization of the asynchronous PIO30_WAITN input adds a further two clock cycles onto the timing of the transfer.

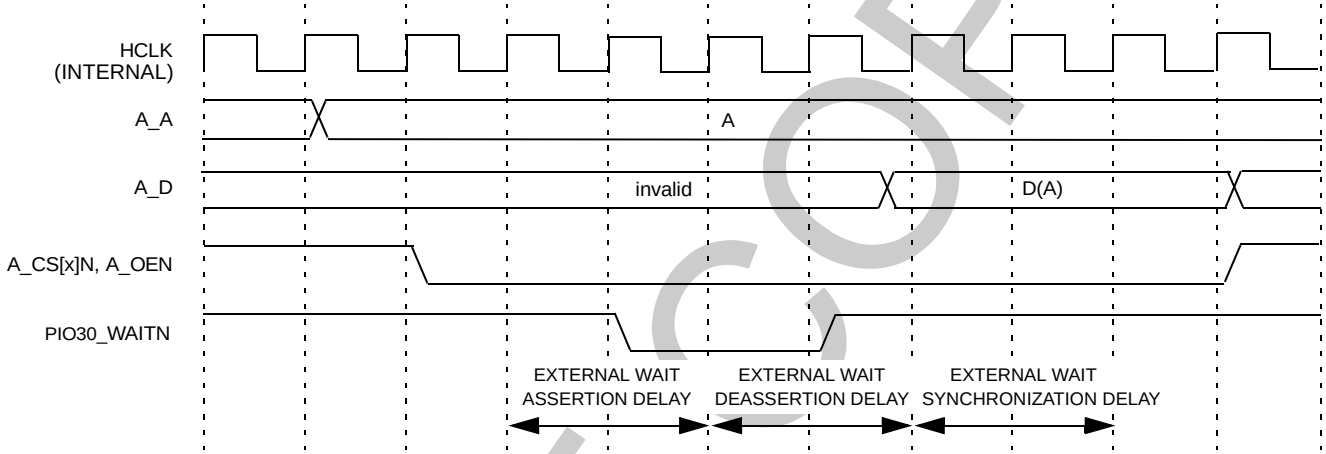


Figure 7.3-20 External Wait-Timed Read Transfer

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-21 shows the timing for an externally waited write transfer, taking two cycles for the wait to be asserted, and two cycles for the wait to be deasserted. An additional cycle is needed at the end of the transfer over a read transfer to allow the deassertion of the write enable before the chip select. An externally waited transfer is also waited on the AHB (unlike a standard write transfer), which allows an error response due to a time-out to be generated correctly.

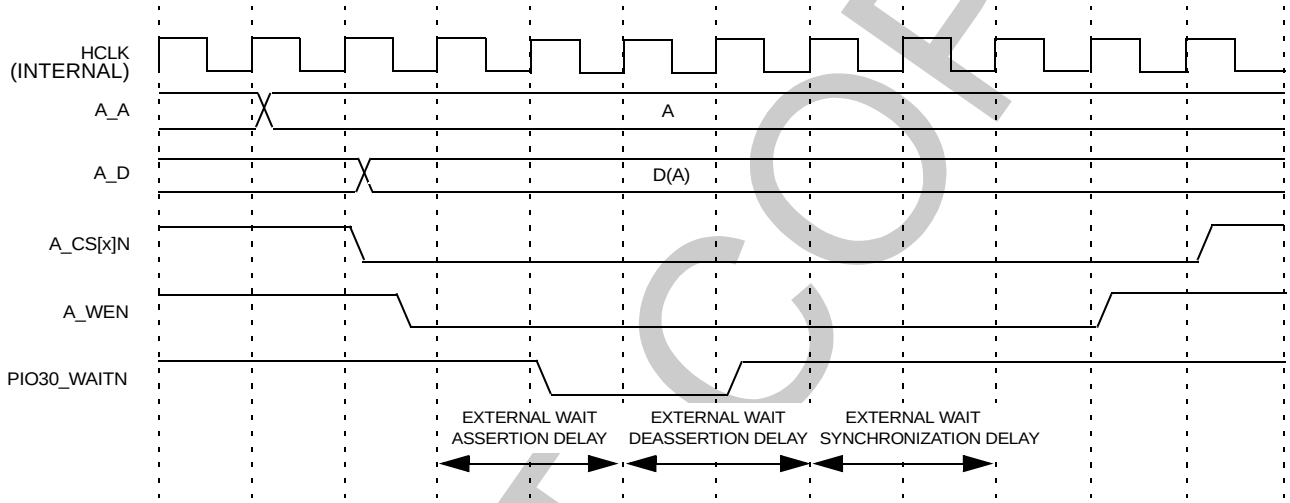


Figure 7.3-21 External Wait-Timed Write Transfer

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

7.3.1.9 Byte Lane Control

T8307 uses A_BE0N and A_BE1N for byte lane control. A_BE0N and A_BE1N are true byte lane control signals generated by *ARM* external memory interface (SMC) block.

External memory bank data bus width can be either 8-bit or 16-bit.

Accesses to Memory Banks Constructed from 8-Bit Memory Devices

Figure 7.3-22 shows the configuration where 8-bit memory devices is used. In this configuration, the RBLE bit must be set to 0.

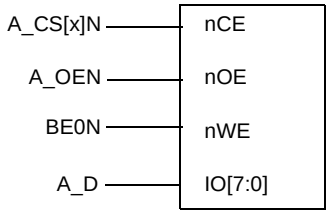


Figure 7.3-22 Memory Banks Constructed from 8-Bit Memory

Accesses to Memory Banks Constructed from 16-Bit Memory Devices

For 16-bit wide memory devices, byte select signals must be appropriately controlled, as shown in Figure 7.3-23. In this case, it is important that the RBLE bit is set to 1 within the respective memory bank control register. This asserts the BE0N and BE1N lines low during a read access to that particular bank, since during a read, all bytes of the devices must be selected to avoid undriven byte lanes on the read data value.

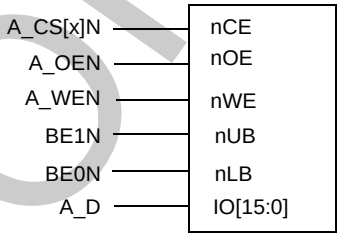


Figure 7.3-23 Memory Banks Constructed from 16-Bit Memory

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Figure 7.3-24 shows connections for a typical memory system with different data width memory devices.

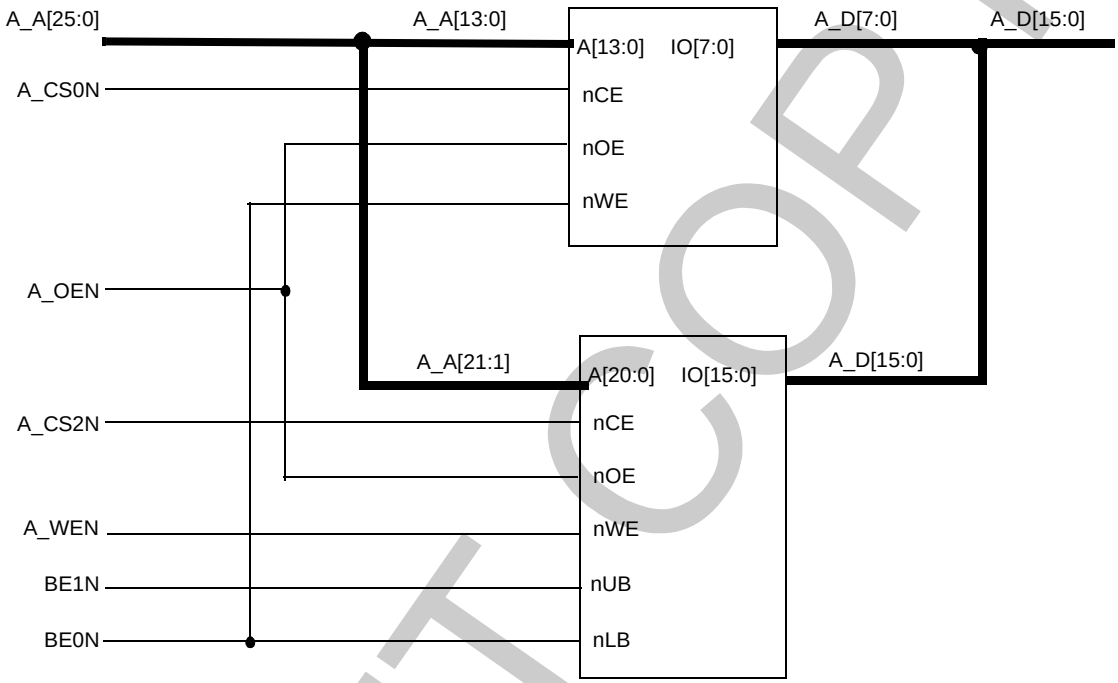


Figure 7.3-24 Typical Memory Connection Diagram

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Byte Lane Control and Data Bus Steering for Little-Endian Configurations

Table 7.3-2 to Table 7.3-5 show the relationship of signals HSIZE[2:0], HADDR[1:0], A_A[1:0], and BE[1:0]N, and mapping of data between the AHB system data bus and external memory data bus. HSIZE[2:0] and HADDR[1:0] are internal signals (part of AHB). HSIZE[1:0] indicates the size of the transfer, where HADDR[1:0] is the two LSBs of the system address bus.

Table 7.3-2 Little-Endian Read, 8-Bit External Bus

Access: Read, Little-Endian, 8-Bit External Bus				System Data Bus Mapping Onto External Data Bus			
Internal Transfer Width	**HSIZE[1:0]**	**HADDR[1:0]**	**A_A[1:0]**	**[31:24]***	**[23:16]***	**[15:8]***	**[7:0]***
Word (four transfers)	10	xx	11	[7:0]	—	—	—
	10	xx	10	—	[7:0]	—	—
	10	xx	01	—	—	[7:0]	—
	10	xx	00	—	—	—	[7:0]
Halfword (two transfers)	01	1x	11	[7:0]	—	—	—
	01	1x	10	—	[7:0]	—	—
Halfword (two transfers)	01	0x	01	—	—	[7:0]	—
	01	0x	00	—	—	—	[7:0]
Byte	00	11	11	[7:0]	—	—	—
Byte	00	10	10	—	[7:0]	—	—
Byte	00	01	01	—	—	[7:0]	—
Byte	00	00	00	—	—	—	[7:0]

* Internal system data bus.

Table 7.3-3 Little-Endian Read, 16-Bit External Bus

Access: Read, Little-Endian, 16-Bit External Bus					System Data Bus Mapping Onto External Data Bus			
Internal Transfer Width	**HSIZE[1:0]**	**HADDR[1:0]**	**A_A[1:0]**	**BE[1:0]N***	**[31:24]†**	**[23:16]†**	**[15:8]†**	**[7:0]†**
Word (two transfers)	10	xx	1x	00	[15:8]	[7:0]	—	—
	10	xx	0x	00	—	—	[15:8]	[7:0]
Halfword	01	1x	1x	00	[15:8]	[7:0]	—	—
Halfword	01	0x	0x	00	—	—	[15:8]	[7:0]
Byte	00	11	1x	00	[15:8]	—	—	—
Byte	00	10	1x	00	—	[7:0]	—	—
Byte	00	01	0x	00	—	—	[15:8]	—
Byte	00	00	0x	00	—	—	—	[7:0]

* BE[1:0]N are driven low as a result of writing 1 to RBLE bit in SMBCRx register.
† Internal system data bus.

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

Table 7.3-4 Little-Endian Write, 8-Bit External Bus

Access: Write, Little-Endian, 8-Bit External Bus				External Data Bus Mapping Onto System Data Bus
Internal Transfer Width	**HSIZE[1:0]**	**HADDR[1:0]**	**A_A[1:0]**	**[7:0]***
Word (four transfers)	10	xx	11	[31:24]
	10	xx	10	[23:16]
	10	xx	01	[15:8]
	10	xx	00	[7:0]
Halfword (two transfers)	01	1x	11	[31:24]
	01	1x	10	[23:16]
Halfword (two transfers)	01	0x	01	[15:8]
	01	0x	00	[7:0]
Byte	00	11	11	[31:24]
Byte	00	10	10	[23:16]
Byte	00	01	01	[15:8]
Byte	00	00	00	[7:0]

* External data bus.

Table 7.3-5 Little-Endian Write, 16-Bit External Bus

Access: Write, Little-Endian, 16-Bit External Bus					External Data Bus Mapping Onto System Data Bus	
Internal Transfer Width	**HSIZE[1:0]**	**HADDR[1:0]**	**A_A[1:0]**	**BE[1:0]N**	**[15:8]***	**[7:0]***
Word (two transfers)	10	xx	1x	00	[31:24]	[23:16]
	10	xx	0x	00	[15:8]	[7:0]
Halfword	01	1x	1x	00	[31:24]	[23:16]
Halfword	01	0x	0x	00	[15:8]	[7:0]
Byte	00	11	1x	01	[31:24]	—
Byte	00	10	1x	10	—	[23:16]
Byte	00	01	0x	01	[15:8]	—
Byte	00	00	0x	10	—	[7:0]

* External data bus.

7.3.2 Booting from ROM After Reset

There are two possible configurations for a system that uses boot ROM:

- Internal ROM.
- External ROM.

T8307 can boot from internal ROM or external ROM, according to the value applied to PIO35_A_A25_BOOTSEL. Bit 1 of the ID register in Section 5.3 will reset to the INVERTED value that is applied to the pin PIO35_A_A25_BOOTSEL on an external pin reset. This bit is unaffected by other resets. If this bit is 1, the core will boot from external ROM. Otherwise, the core will boot from internal ROM.

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

7.3.3 Registers

The SMC control and status registers are shown in Table 6.2-1. These registers are explained in detail in the following sections.

Note: SMC_BANK_ADDR is the base address of the SMC registers for each memory bank. For T8307, SMC_BANK_ADDR = 0x70000000, 0x7000001C, 0x70000038, 0x70000054, 0x70000070, 0x7000008C, 0x700000A8, and 0x700000C4 for memory banks 0 to 7 respectively.

7.3.3.1 Bank Idle Cycle Control Registers (SMBIDCYR0—SMBIDCYR7)

SMBCIDCYR0 to SMBCIDCYR7 are the SMC bank idle cycle control registers, which need to be programmed for the configuration of the SMC memory banks 0 to 7. Each register is identical in structure. The descriptions for the register bits are given in Table 7.3-6.

Table 7.3-6 Bank Idle Cycle Control Registers (SMBIDCYR0—SMBIDCYR7), Addresses (SMC_BANK_ADDR + 0x00)

Bit	31—4	3—0
Name	RSVD	IDCY

Bit	Name	Type	Description
31—4	RSVD	RAZ	Reserved. Do not modify and read as zero.
3—0	IDCY	Read/Write	Idle/turnaround cycles. Defaults to 1111 at reset. This field controls the number of bus turnaround cycles added between read and write accesses, to prevent bus contention on the external memory data bus. The turn around time is (IDCY + 1) x tHCLK*.

* THCLK = Period of HCLK.

7.3.3.2 Bank Wait-State 1 Control Registers (SMBWST1R0—SMBWST1R7)

SMBWST1R0 to SMBWST1R7 are the SMC bank wait-state one control registers, which need to be programmed for the configuration of the SMC memory banks 0 to 7. Each register is identical in structure. The descriptions for the register bits are given in Table 7.3-7.

Table 7.3-7 Bank Wait-State 1 Control Registers (SMBWST1R0—SMBWST1R7), Addresses (SMC_BANK_ADDR + 0x04)

Bit	31—5	4—0
Name	RSVD	WST1

Bit	Name	Type	Description
31—5	RSVD	RAZ	Reserved. Do not modify and read as zero.
4—0	WST1	Read/Write	Wait-state 1. Defaults to 11111 at reset. SRAM and ROM: the WST1 field controls the number of wait-states for read accesses. Burst ROM: the WST1 field controls the number of wait-states for the first read access only. All: the WST1 field controls the external wait assertion timing for reads. Wait-state time = (WST1 + 1) x tHCLK*.

* THCLK = Period of HCLK.

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

7.3.3.3 Bank Wait-State 2 Control Registers (SMBWST2R0—SMBWST2R7)

SMBWST2R0 to SMBWST2R7 are the SMC bank wait-state two control registers, which need to be programmed for the configuration of the SMC memory banks 0 to 7. Each register is identical in structure. The descriptions for the register bits are given in Table 7.3-8.

Table 7.3-8 Bank Wait-State 2 Control Registers (SMBWST2R0—SMBWST2R7), Addresses (SMC_BANK_ADDR + 0x08)

Bit	31—5	4—0
Name	RSVD	WST2

Bit	Name	Type	Description
31—5	RSVD	RAZ	Reserved. Do not modify and read as zero.
4—0	WST2	Read/Write	Wait-state 2. Defaults to 11111 at reset. SRAM: the WST2 field controls the number of wait-states for write accesses, and the external wait assertion timing for writes. This wait-state time is (WST2 + 1) x tHCLK in the case of SRAM. Burst ROM: the WST2 field controls the number of wait-states for the burst read accesses after the first read. This wait-state time is (WST2) x tHCLK* in the case of burst ROM. ROM: WST2 does not apply to ROM devices.

* THCLK = Period of HCLK.

7.3.3.4 Bank Output Enable Assertion Delay Control Registers (SMBWSTOENR0—SMBWSTOENR7)

SMBWSTOENR0 to SMBWSTOENR7 are the SMC bank output enable assertion delay control registers, which need to be programmed for the configuration of the SMC memory banks 0 to 7. Each register is identical in structure. The descriptions for the register bits are given in Table 7.3-9.

Table 7.3-9 Bank Output Enable Assertion Delay Control Registers (SMBWSTOENR0—SMBWSTOENR7), Addresses (SMC_BANK_ADDR + 0x0C)

Bit	31—4	3—0
Name	RSVD	WSTOEN

Bit	Name	Type	Description
31—4	RSVD	RAZ	Reserved. Do not modify and read as zero.
3—0	WSTOEN	Read/Write	Output enable assertion delay from chip select assertion. Defaults to 0000 at reset.

7.3.3.5 Bank Write Enable Assertion Delay Control Registers (SMBWSTWENR0—SMBWSTWENR7)

SMBWSTWENR0 to SMBWSTWENR7 are the SMC bank write enable assertion delay control registers, which need to be programmed for the configuration of the SMC memory banks 0 to 7. Each register is identical in structure. The descriptions for the register bits are given in Table 7.3-10.

Table 7.3-10 Bank Write Enable Assertion Delay Control Registers (SMBWSTWENR0—SMBWSTWENR7), Addresses (SMC_BANK_ADDR + 0x10)

Bit	31—4	3—0
Name	RSVD	WSTWEN

Bit	Name	Type	Description
31—4	RSVD	RAZ	Reserved. Do not modify and read as zero.
3—0	WSTWEN	Read/Write	Write enable assertion delay from chip select assertion. Defaults to 0000 at reset.

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (continued)

7.3.3.6 Bank Control Registers (SMBCR0—SMBCR7)

SMBCR0 to SMBCR7 are the SMC bank control registers, which need to be programmed for the configuration of the SMC memory banks 0 to 7. Each register is identical in structure. At reset, the memory bank default external memory width is as shown in Table 7.3-11.

Table 7.3-11 SMC Reset Default Memory Width

MC Memory Bank	Default Memory Width
Bank 0	16-bit
Bank 1	16-bit
Bank 2	16-bit
Bank 3	16-bit
Bank 4	16-bit
Bank 5	16-bit
Bank 6	16-bit
Bank 7	16-bit

The descriptions for the register bits are given in Table 7.3-12.

Table 7.3-12 Bank Control Registers (SMBCR0—SMBCR7), Addresses (SMC_BANK_ADDR + 0x14)

Bit	31—8	7—6	5	4	3	2	1	0
Name	RSVD	MW	BM	WP	CSPol	WaitEn	WaitPol	RBLE

Bit	Name	Type	Description
31—8	RSVD	RAZ	Reserved. Do not modify and read as zero.
7—6	MW	Read/Write	Memory width. Defaults to 01 at reset for each bank (see Table 7.3-11). 00—8-bit. 01—16-bit. 10 or 11—Reserved.
5	BM	Read/Write	Burst mode. 0—Nonburst memory devices (default at reset). 1—Burst ROM memory.
4	WP	Read/Write	Write protect. 0—No write protection, e.g., SRAM, write enabled flash (default at reset). 1—Device is write protected, e.g., ROM, burst ROM, or read only flash or SRAM.
3	CSPol	Read/Write	The chip select polarity bit indication for each bank. 0—Active-low SMCS (default at reset). 1—Active-high SMCS. **Note:** Care must be taken to set this bit to the correct value if the device has the output enable lines tied off so that the outputs will always be driven, since if the incorrect polarity chip select is used, the device will continuously drive data out.
2	WaitEn	Read/Write	External memory controller wait signal enable. 0—The SMC will not be controlled by the external wait signal (default at reset). 1—The SMC will look for the external wait input signal, SMWAIT.
1	WaitPol	Read/Write	Polarity of the external wait input for activation. 0—The SMWAIT signal is active-low (default at reset). 1—The SMWAIT signal is active-low.
0	RBLE	Read/Write	Read byte lane enable. 0—BE[1:0]N all deasserted high during system reads from external memory (default at reset). 1—BE[1:0]N all asserted low during system reads from external memory. For 16-bit devices, RBLE should always be set to 1 so that the BE[1:0]N outputs low during a read.

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

7.3.3.7 Bank Status Registers (SMBSR0—SMBSR7)

SMBSR0 to SMBSR7 are the SMC bank status registers. These registers indicate the state of various conditions such as errors on write protected regions, time-outs due to external waits and any bus transfer errors. The software can clear each of the error conditions by writing a 1 to the appropriate bit position. The descriptions for the register bits are given in Table 7.3-13.

Table 7.3-13 Bank Status Registers (SMBSR0—SMBSR7), Addresses (SMC_BANK_ADDR + 0x18)

Bit	31—3	2	1	0
Name	RSVD	WaitToutErr	WriteProtErr	BusErr

Bit	Name	Type	Description
31—3	RSVD	RAZ	Reserved. Do not modify and read as zero.
2	WaitToutErr	Read/Write	External wait time-out error flag, read: 0—No error (default at reset). 1—External wait time-out error. Writing a 1 to this bit will clear the external wait time-out error flag. Writing a 0 to this bit will have no effect.
1	WriteProtErr	Read/Write	Write protect error status flag, read: 0—No error (default at reset). 1—Write protect error. Writing a 1 to this bit will clear the write protect error status flag. Writing a 0 to this bit will have no effect.
0	BusErr	Read/Write	Bus transfer error status flag, read: 0—No error (default at reset). 1—Bus transfer error. Writing a 1 to this bit will clear the bus transfer error status flag. Writing a 0 to this bit will have no effect.

7 Call Processor (CP) Block (continued)

7.3 Static Memory Controller (SMC) (continued)

7.3.3.8 Bank Output Enable Deassertion to Chip Select Deassertion Hold Delay Control Registers (SMBWST2OENR0—SMBWST2OENR7)

SMBWST2OENR0 to SMBWST2OENR7 are the SMC bank output enable deassertion to chip select deassertion hold delay control registers, which need to be programmed for the configuration of the SMC memory banks 0 to 7. Each register is identical in structure. The descriptions for the register bits are given in Table 7.3-14.

Table 7.3-14 Bank Output Enable Deassertion to Chip Select Deassertion Hold Delay Control Registers (SMBWST2OENR0—SMBWST2OENR7), Addresses (0x700000E4 + 8 *n*)

Bit	31—4	3—0
Name	RSVD	WST2OEN

Bit	Name	Type	Description
31—4	RSVD	RAZ	Reserved. Do not modify and read as zero.
3—0	WST2OEN	Read/Write	Chip select deassertion delay from output enable deassertion. Defaults to 0000 at reset.

7.3.3.9 Bank Write Enable Deassertion to Chip Select Deassertion Hold Delay Control Registers (SMBWST2WENR0—SMBWST2WENR7)

SMBWST2WENR0 to SMBWST2WENR7 are the SMC bank write enable deassertion to chip select deassertion hold delay control registers, which need to be programmed for the configuration of the SMC memory banks 0 to 7. Each register is identical in structure. The descriptions for the register bits are given in Table 7.3-15.

Table 7.3-15 Bank Write Enable Deassertion to Chip Select Deassertion Hold Delay Control Registers (SMBWST2WENR0—SMBWST2WENR7), Addresses (0x700000E8 + 8 *n*)

Bit	31—4	3—0
Name	RSVD	WST2WEN

Bit	Name	Type	Description
31—4	RSVD	RAZ	Reserved. Do not modify and read as zero.
3—0	WST2WEN	Read/Write	Chip select deassertion delay from write enable deassertion. Defaults to 0000 at reset.

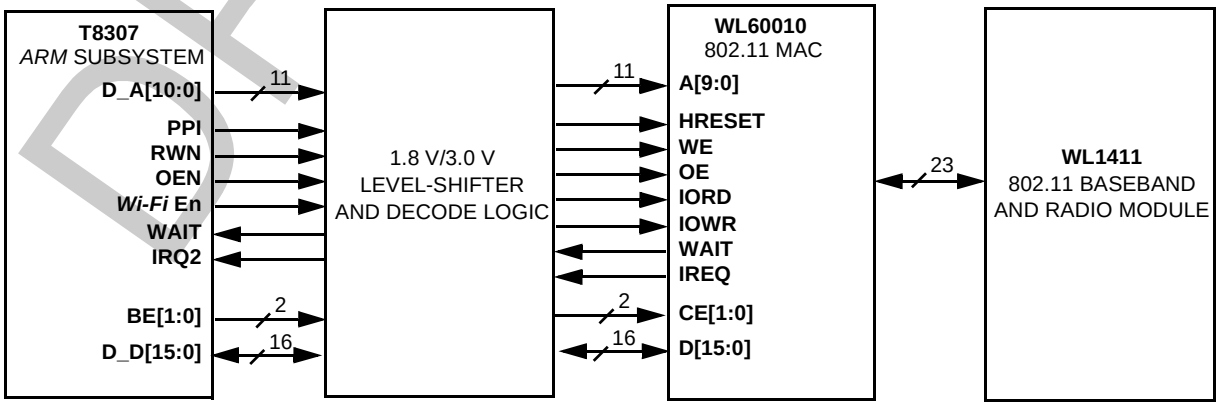


Figure 7.3-25 T8307 *Wi-Fi* Interface

7 Call Processor (CP) Block (continued)

7.4 DMA Controller (DMAC)

The DMA controller allows peripheral-to-memory, memory-to-peripheral, peripheral-to-peripheral, and memory-to-memory transactions.

T8307 CP block DMAC has 4 channels. Each DMA channel can provide unidirectional DMA transfers for a single source and destination. For example, a bidirectional serial port requires one channel for transmit and one for receive. The source and destination areas can each be either a memory region or a peripheral. The source and destination areas will be accessible through the single AHB master within the DMAC.

7.4.1 Operation

7.4.1.1 Enabling the DMA Controller

To enable the DMA controller set the DMA enable bit in the DMACConfiguration register.

7.4.1.2 Disabling the DMA Controller

To disable the DMA controller, complete the following steps:

1. Read the DMACEnbldChns register and ensure that all the DMA channels have been disabled. If any channels are active see Section 7.4.1.4.
2. Disable the DMA controller by writing 0 to the DMA enable bit in the DMACConfiguration register.

7.4.1.3 Enabling a DMA Channel

To enable the DMA channel set the channel enable bit in the relevant DMA channel configuration register.

Note: The channel must be fully initialized before it is enabled. Additionally, the enable bit of the DMA controller must be set before any channels are enabled.

7.4.1.4 Disabling a DMA Channel

A DMA channel can be disabled in the following ways:

- Write directly to the channel enable bit. Any outstanding data in the FIFOs is lost if this method is used.
- Use the active and halt bits in conjunction with the channel enable bit.
- Wait until the transfer completes. The channel is then automatically disabled.

7.4.1.5 Disabling a DMA Channel and Losing Data in the FIFO

Clear the relevant channel enable bit in the relevant channel configuration register. The current AHB transfer (if one is in progress) completes and the channel is disabled. Any data in the FIFO is lost.

7.4.1.6 Disabling a DMA Channel Without Losing Data in the FIFO

To disable a DMA channel without losing data in the FIFO:

1. Set the halt bit in the relevant channel configuration register. This causes any further DMA requests to be ignored.
2. Poll the active bit in the relevant channel configuration register until it reaches 0. This bit indicates whether there is any data in the channel that has to be transferred.
3. Clear the channel enable bit in the relevant channel configuration register.

7.4.1.7 Set Up a New DMA Transfer

To set up a new DMA transfer, complete the following steps:

1. If the channel is not set aside for the DMA transaction, then complete the following steps:
   a. Read the DMACEnbldChns controller register and find out which channels are inactive.
   b. Choose an inactive channel that has the required priority.
3. Program the DMA controller.

7 Call Processor (CP) Block (continued)

7.4 DMA Controller (DMAC) (continued)

7.4.1.8 Halting a DMA Channel

Set the halt bit in the relevant DMA channel configuration register. The current source request is serviced. Any further source DMA requests are ignored until the halt bit is cleared.

7.4.1.9 Programming a DMA Channel

To program a DMA channel, complete the following steps:

1. Choose a free DMA channel with the priority needed. Where DMA channel 0 has the highest priority and DMA channel 3 the lowest priority.
2. Clear any pending interrupts on the channel to be used by writing to the DMACIntTCClr and DMACIntErrClr registers. The previous channel operation might have left interrupts active.
3. Write the source address into the DMACCxSrcAddr register.
4. Write the destination address into the DMACCxDestAddr register.
5. Write the address of the next LLI into the DMACCxLLI register. If the transfer comprises of a single packet of data then 0 must be written into this register.
6. Write the control information into the DMACCxControl register.
7. Write the channel configuration information into the DMACCxConfiguration register. If the enable bit is set then the DMA channel is automatically enabled.

7.4.2 Registers

The following sections list the DMA controller registers. For T8307, DMA_CH_ADDR = 0x70003100, 0x70003120, 0x70003140 and 0x70003160 for DMA channels 0, 1, 2, and 3, respectively.

7 Call Processor (CP) Block (continued)

7.4 DMA Controller (DMAC) (continued)

7.4.2.1 Interrupt Status Register (DMACIntStatus)

The DMACIntStatus register is read-only and shows the status of the interrupts after masking. A high bit indicates that a specific DMA channel interrupt request is active. The request can be generated from either the error or terminal count interrupt requests. Table 7.4-1 shows the bit assignment of the DMACIntStatus register.

Table 7.4-1 Interrupt Status Register (DMACIntStatus), Address (0x70003000)

Bit	3—0
Name	IntStatus

Bit	Name	Type	Function
3—0	IntStatus	Read	Status of the DMA interrupts after masking.

7.4.2.2 Interrupt Terminal Count Status Register (DMACIntTCStatus)

The DMACIntTCStatus register is read-only and indicates the status of the terminal count after masking. This register must be used in conjunction with the DMACIntStatus register if the combined interrupt request, DMACINTCOMBINE, is used to request interrupts.

If the DMACINTTC interrupt request is used, only the DMACIntTCStatus register is read to ascertain the source of the interrupt request. Table 7.4-2 shows the bit assignment of the DMACIntTCStatus register.

Table 7.4-2 Interrupt Terminal Count Status Register (DMACIntTCStatus), Address (0x70003004)

Bit	3—0
Name	IntTCStatus

Bit	Name	Type	Function
3—0	IntTCStatus	Read	Interrupt terminal count request status.

7.4.2.3 Interrupt Terminal Count Clear Register (DMACIntTCClear)

The DMACIntTCClear register is write-only and is used to clear a terminal count interrupt request. When writing to this register, each data bit that is set high causes the corresponding bit in the status register to be cleared. Data bits that are low have no effect on the corresponding bit in the register. Table 7.4-3 shows the bit assignment of the DMACIntTCClear register.

Table 7.4-3 Interrupt Terminal Count Clear Register (DMACIntTCClear), Address (0x70003008)

Bit	3—0
Name	IntTCClear

Bit	Name	Type	Function
3—0	IntTCClear	Write	Terminal count request clear.

7 Call Processor (CP) Block (continued)

7.4 DMA Controller (DMAC) (continued)

7.4.2.4 Interrupt Error Status Register (DMACIntErrorStatus)

The DMACIntErrorStatus register is read-only register and indicates the status of the error request after masking. This register must be used in conjunction with the DMACIntStatus register if the combined interrupt request, DMACINTCOMBINE, is used to request interrupts.

If the DMACINTERROR interrupt request is used only the DMACIntErrorStatus register needs to be read. Table 7.4-4 shows the bit assignment of the DMACIntErrorStatus register.

Table 7.4-4 Interrupt Error Status Register (DMACIntErrorStatus), Address (0x7000300C)

Bit	3—0
Name	IntErrorStatus

Bit	Name	Type	Function
3—0	IntErrorStatus	Read	Interrupt error status.

7.4.2.5 Interrupt Error Clear Register (DMACIntErrClr)

The DMACIntErrClr register is a write-only register and is used to clear the error interrupt requests. When writing to this register, each data bit that is high causes the corresponding bit in the status register to be cleared. Data bits that are low have no effect on the corresponding bit in the register. Table 7.4-5 shows the bit assignment of the DMACIntErrClr register.

Table 7.4-5 Interrupt Error Clear Register (DMACIntErrClr), Address (0x70003010)

Bit	3—0
Name	IntErrClr

Bit	Name	Type	Function
3—0	IntErrClr	Write	Interrupt error clear.

7.4.2.6 Raw Interrupt Terminal Count Status Register (DMACRawIntTCStatus)

The DMACRawIntTCStatus register is read-only. It indicates which DMA channels are requesting a transfer complete (terminal count interrupt) prior to masking. A high bit indicates that the terminal count interrupt request is active prior to masking. Table 7.4-6 shows the bit assignment of the DMACRawIntTCStatus register.

Table 7.4-6 Raw Interrupt Terminal Count Status Register (DMACRawIntTCStatus), Address (0x70003014)

Bit	3—0
Name	RawIntTCStatus

Bit	Name	Type	Function
3—0	RawIntTCStatus	Read	Status of the terminal count interrupt prior to masking.

7 Call Processor (CP) Block (continued)

7.4 DMA Controller (DMAC) (continued)

7.4.2.7 Raw Error Interrupt Status Register (DMACRawIntErrorStatus)

The DMACRawIntErrorStatus register is read-only. It indicates which DMA channels are requesting an error interrupt prior to masking. A high bit indicates that the error interrupt request is active prior to masking. Table 7.4-7 shows the bit assignment of register of the DMACRawIntErrorStatus register.

Table 7.4-7 Raw Error Interrupt Status Register (DMACRawIntErrorStatus), Address (0x70003018)

Bit	3—0
Name	RawIntErrorStatus

Bit	Name	Type	Function
3—0	RawIntErrorStatus	Read	Status of the error interrupt prior to masking.

7.4.2.8 Enabled Channel Register (DMACEnbldChns)

The DMACEnbldChns register is read-only and indicates which DMA channels are enabled, as indicated by the Enable bit in the DMACCxConfiguration register. A high bit indicates that a DMA channel is enabled. A bit is cleared on completion of the DMA transfer. Table 7.4-8 shows the bit assignment of the DMACEnbldChns register.

Table 7.4-8 Enabled Channel Register (DMACEnbldChns), Address (0x7000301C)

Bit	3—0
Name	EnabledChannels

Bit	Name	Type	Function
3—0	EnabledChannels	Read	Channel enable status.

7.4.2.9 Software Burst Request Register (DMACSoftBReq)

The DMACSoftBReq register is read/write and it allows DMA burst requests to be generated by software. A DMA request can be generated for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Writing 0 to this register has no effect.

Reading the register indicates which sources are requesting DMA burst transfers. A request can be generated from either a peripheral or the software request register. Table 7.4-9 shows the bit assignment of the DMACSoftBReq register.

Table 7.4-9 Software Burst Request Register (DMACSoftBReq), Address (0x70003020)

Bit	15—0
Name	SoftBReq

Bit	Name	Type	Function
15—0	SoftBReq	Read/write	Software burst request.

Note: It is recommended that software and hardware peripheral requests are not used at the same time.

7 Call Processor (CP) Block (continued)

7.4 DMA Controller (DMAC) (continued)

7.4.2.10 Software Single Request Register (DMACSoftSReq)

The DMACSoftSReq read/write register allows DMA single requests to be generated by software. A DMA request can be generated for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Writing 0 to this register has no effect.

Reading the register indicates which sources are requesting single DMA transfers. A request can be generated from either a peripheral or the software request register. Table 7.4-10 shows the bit assignment of the DMACSoftSReq register.

Table 7.4-10 Software Single Request Register (DMACSoftSReq), Address (0x70003024)

Bit	15—0
Name	SoftSReq

Bit	Name	Type	Function
15—0	SoftSReq	Read/write	Software single request.

Note: It is recommended that software and hardware peripheral requests are not used at the same time.

7.4.2.11 Software Last Burst Request Register (DMACSoftLBReq)

The DMACSoftLBReq read/write register allows DMA last burst requests to be generated by software. A DMA request can be generated for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Writing 0 to this register has no effect.

Reading the register indicates which sources are requesting last burst DMA transfers. A request can be generated from either a peripheral or the software request register. Table 7.4-11 shows the bit assignment of the DMACSoftLBReq register.

Table 7.4-11 Software Last Burst Request Register (DMACSoftLBReq), Address (0x70003028)

Bit	15—0
Name	SoftLBReq

Bit	Name	Type	Function
15—0	SoftLBReq	Read/write	Software last burst request.

7 Call Processor (CP) Block (continued)

7.4 DMA Controller (DMAC) (continued)

7.4.2.12 Software Last Single Request Register (DMACSoftLSReq)

The DMACSoftLSReq read/write register allows DMA last single requests to be generated by software. A DMA request can be generated for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Writing 0 to this register has no effect.

Reading the register indicates which sources are requesting last single DMA transfers. A request can be generated from either a peripheral or the software request register. Table 7.4-12 shows the bit assignment of the DMACSoftLSReq register.

Table 7.4-12 Software Last Single Request Register (DMACSoftLSReq), Address (0x7000302C)

Bit	15—0
Name	SoftLSReq

Bit	Name	Type	Function
15—0	SoftLSReq	Read/write	Software last single request.

7.4.2.13 Configuration Register (DMACConfiguration)

The DMACConfiguration read/write register is used to configure the operation of the DMA controller. The endianness of the DMAC AHB master interface can be altered by writing to the M1 bit of this register. The AHB master interface is set to little-endian mode on reset. Table 7.4-13 shows the bit assignment of the DMACConfiguration register.

Table 7.4-13 Configuration Register (DMACConfiguration), Address (0x70003030)

Bit	31—2	1	0
Name	RSVD	M1	E

Bit	Name	Type	Function
31—2	RSVD	—	Reserved. Read as zero, do not modify.
1	M1	Read/write	Endianness configuration for the AHB master within the DMAC: 0—Little-endian mode. 1—Big-endian mode. This bit is reset to 0.
0	E	Read/write	DMA controller enable: 0—Disabled. 1—Enabled. This bit is reset to 0.

7 Call Processor (CP) Block (continued)

7.4 DMA Controller (DMAC) (continued)

7.4.2.14 Synchronization Register (DMACSync)

The DMACSync read/write register is used to enable or disable synchronization logic for the DMA request signals. The DMA request signals consist of the DMACBREQ[15:0], DMACSREQ[15:0], DMACLBREQ[15:0], and DMACLSREQ[15:0] signals. A bit set to 0 enables the synchronization logic for a particular group of DMA requests. A bit set to 1 disables the synchronization logic for a particular group of DMA requests.

This register is reset to 0, synchronization logic enabled.

Note: Synchronization logic must be used when the peripheral generating the DMA request runs on a different clock to the DMA controller. For peripherals running on the same clock as the DMA controller disabling the synchronization logic improves the DMA request response time. If necessary, the DMA response signals, DMACCLR and DMACTC, must be synchronized in the peripheral. Table 7.4-14 shows the bit assignment of the DMACSync register.

Table 7.4-14 Synchronization Register (DMACSync), Address (0x70003034)

Bit	15—0
Name	DMACSync

Bit	Name	Type	Function
15—0	DMACSync	Read/write	DMA synchronization logic for DMA request signals enabled or disabled. A low bit indicates that the synchronization logic for the DMACBREQ[15:0], DMACSREQ[15:0], DMACLBREQ[15:0], and DMACLSREQ[15:0] request signals is enabled. A high bit indicates that the synchronization logic is disabled.

7 Call Processor (CP) Block (continued)

7.4 DMA Controller (DMAC) (continued)

7.4.2.15 Channel Registers

The channel registers are used to program a DMA channel. These registers consist of the following:

- Four DMACCxSrcAddr registers.
- Four DMACCxDestAddr registers.
- Four DMACCxLLI registers.
- Four DMACCxControl registers.
- Four DMACCxConfiguration registers.

When performing scatter/gather DMA the first four registers are automatically updated.

7.4.2.16 Channel Source Address Registers (DMACCxSrcAddr)

The four read/write DMACCxSrcAddr registers contain the current source address (byte-aligned) of the data to be transferred. Each register is programmed directly by software before the appropriate channel is enabled. When the DMA channel is enabled this register is updated:

- As the source address is incremented.
- By following the linked list when a complete packet of data has been transferred.

Reading the register when the channel is active does not provide useful information. This is because by the time that software has processed the value read, the channel might have progressed. It is intended to be read only when the channel has stopped, in which case it shows the source address of the last item read. Table 7.4-15 shows the bit assignment of the DMACCxSrcAddr registers.

Note: The source and destination addresses must be aligned to the source and destination widths.

Table 7.4-15 Channel Source Address Register (DMACCxSrcAddr), Address (DMA_CH_ADDR + 0x00)

Bit	31—0
Name	SrcAddr

Bit	Name	Type	Function
31—0	SrcAddr	Read/write	DMA source address.

7 Call Processor (CP) Block (continued)

7.4 DMA Controller (DMAC) (continued)

7.4.2.17 Channel Destination Address Registers (DMACCxDestAddr)

The four read/write DMACCxDestAddr registers contain the current destination address (byte-aligned) of the data to be transferred.

Each register is programmed directly by software before the channel is enabled. When the DMA channel is enabled the register is updated as the destination address is incremented and by following the linked list when a complete packet of data has been transferred.

Reading the register when the channel is active does not provide useful information. This is because by the time that software has processed the value read, the channel might have progressed. It is intended to be read only when a channel has stopped, in which case it shows the destination address of the last item read. Table 7.4-16 shows the bit assignment of a DMACCxDestAddr register.

Table 7.4-16 Channel Destination Address Register (DMACCxDestAddr), Address (DMA_CH_ADDR + 0x04)

Bit	31—0
Name	DestAddr

Bit	Name	Type	Function
31—0	DestAddr	Read/write	DMA destination address.

7.4.2.18 Channel Linked List Item Register (DMACCxLLI)

The four read/write DMACCxLLI registers contain a word aligned address of the next *Linked List Item* (LLI). If the LLI is 0, then the current LLI is the last in the chain, and the DMA channel is disabled once all DMA transfers associated with it are completed.

Note: Programming this register when the DMA channel is enabled has unpredictable side effects. Table 7.4-17 shows the bit assignment of a DMACCxLLI register.

Table 7.4-17 Channel Linked List Item Register (DMACCxLLI), Address (DMA_CH_ADDR + 0x08)

Bit	31—2	1	0
Name	LLI	RSVD	RSVD

Bit	Name	Type	Function
31—2	LLI	Read/write	Linked list item. Bits[31:2] of the address for the next LLI. Address bits[1:0] are 0.
1	RSVD	Read/write	Reserved. Must be written as 0, masked on read.
0	RSVD	Read/write	Reserved. Read as 0. Must be written as 0.

Note: To make loading the LLIs more efficient for some systems, the LLI data structures can be made 4-word aligned.

7 Call Processor (CP) Block (continued)

7.4 DMA Controller (DMAC) (continued)

7.4.2.19 Channel Control Registers (DMACCxControl)

The four read/write DMACCxControl registers contain DMA channel control information such as the transfer size, burst size, and transfer width. Each register is programmed directly by software before the DMA channel is enabled. When the channel is enabled the register is updated by following the linked list when a complete packet of data has been transferred.

Reading the register whilst the channel is active does not give useful information. This is because by the time that software has processed the value read, the channel might have progressed. It is intended to be read only when a channel has stopped. Table 7.4-18 shows the bit assignment of a DMACCxControl register.

Table 7.4-18 Channel Control Register (DMACCxControl), Address (DMA_CH_ADDR + 0x0C)

Bit	31	30—28	27	26	25—24	23—21	20—18	17—15	14—12	11—0
Name	I	Prot	DI	SI	RSVD	DWidth	SWidth	DBSize	SBSize	TransferSize

Bit	Name	Type	Function
31	I	Read/write	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30—28	Prot	Read/write	Protection.
27	DI	Read/write	Destination increment. When set the destination address is incremented after each transfer.
26	SI	Read/write	Source increment. When set the source address is incremented after each transfer.
25—24	RSVD	Read/write	Reserved. Read as 0. Must be written with 0.
23—21	DWidth	Read/write	Destination transfer width. Transfers wider than the AHB master bus width are illegal. The source and destination widths can be different from each other. The hardware automatically packs and unpacks the data as required.
20—18	SWidth	Read/write	Source transfer width. Transfers wider than the AHB master bus width are illegal. The source and destination widths can be different from each other. The hardware automatically packs and unpacks the data as required.
17—15	DBSize	Read/write	Destination burst size. Indicates the number of transfers that make up a destination burst transfer request. This value must be set to the burst size of the destination peripheral, or if the destination is memory, to the memory boundary size.
14—12	SBSize	Read/write	Source burst size. Indicates the number of transfers that make up a source burst. This value must be set to the burst size of the source peripheral, or if the source is memory, to the memory boundary size.
11—0	TransferSize	Read/write	Transfer size. Indicates the number of (source width) transfers to perform when the DMA controller is the flow controller. The transfer size value must be set before the channel is enabled. Transfer size is updated as data transfers are completed on the destination bus. Reading the register when the channel is active does not give useful information. This is because by the time that software has processed the value read, the channel might have progressed. It is intended to be used only when a channel has stopped. If the DMA controller is not the flow controller the transfer size value is not used. This register is decremented after each destination transfer.

7 Call Processor (CP) Block (continued)

7.4 DMA Controller (DMAC) (continued)

Table 7.4-19 shows the value of the DBSize or SBSsize bits and the corresponding burst sizes.

Table 7.4-19 Source or Destination Burst Size

Bit value of DBSize or SB Size	Source or Destination Burst Transfer Request Size
000	1
001	4
010	8
011	16
100	32
101	64
110	128
111	256

Table 7.4-20 shows the value of the SWidth or DWidth bits and the corresponding width.

Table 7.4-20 Source or Destination Burst Width

Bit value of SWidth or DWidth	Source or Destination Width
000	Byte (8-bit)
001	Halfword (16-bit)
010	Word (32-bit)
011	Reserved
100	Reserved
101	Reserved
110	Reserved
111	Reserved

7 Call Processor (CP) Block (continued)

7.4 DMA Controller (DMAC) (continued)

AHB access information is provided to the source and destination peripherals when a transfer occurs. The transfer information is provided by programming the DMA channel (the Prot bit of the DMACCxControl register, and the Lock bit of the DMACCxConfiguration register). These bits are programmed by software and peripherals can use this information if necessary. Three bits of information are provided, and Table 7.4-21 describes the purpose of the three protection bits.

Table 7.4-21 Protection Bits

Bit	2	1	0
Name	Cachable or not cachable	Bufferable or not bufferable	Privileged or user

Bit	Description	Purpose
2	Cachable or not cachable	Indicates that the access is cachable or not cachable: 0—Not cachable. 1—Cachable. This indicates that the access is cachable. This bit can, for example, be used to indicate to an AMBA bridge that when it saw the first read of a burst of eight it can transfer the whole burst of eight reads on the destination bus, rather than pass the transactions through one at a time. This bit controls the AHB HPROT[3] signal.
1	Bufferable or not bufferable	Indicates that the access is bufferable, or not bufferable: 0—Not bufferable. 1—Bufferable. This bit indicates that the access is bufferable. This bit can, for example, be used to indicate to an AMBA bridge that the read can complete in zero wait-states on the source bus without waiting for it to arbitrate for the destination bus and for the slave to accept the data. This bit controls the AHB HPROT[2] signal.
0	Privileged or user	Indicates that the access is in user, or privileged mode: 0—User mode. 1—Privileged mode. This bit controls the AHB HPROT[1] signal.

7 Call Processor (CP) Block (continued)

7.4 DMA Controller (DMAC) (continued)

7.4.2.20 Channel Configuration Registers (DMACCxConfiguration)

The four DMACCxConfiguration registers are read/write and are used to configure the DMA channel. The registers are not updated when a new LLI is requested. Table 7.4-22 shows the bit assignment of a DMACCxConfiguration register.

Table 7.4-22 Channel Configuration Register (DMACCxConfiguration), Address (DMA_CH_ADDR + 0x10)

Bit	31—19	18	17	16	15	14	13—11	10—6	5—1	0
Name	RSVD	H	A	L	ITC	IE	FlowCntrl	DestPeripheral	SrcPeripheral	E

Bit	Name	Type	Function
31—19	RSVD	—	Reserved. Must be written as zero, masked on read.
18	H	Read/write	Halt: 0—Allow DMA requests. 1—Ignore further source DMA requests. The contents of the channels FIFO are drained. This value can be used with the active and channel enable bits to cleanly disable a DMA channel.
17	A	Read	Active: 0—There is no data in the FIFO of the channel. 1—The FIFO of the channel has data. This value can be used with the halt and channel enable bits to cleanly disable a DMA channel.
16	L	Read/write	Lock. When set this bit enables locked transfers.
15	ITC	Read/write	Terminal count interrupt mask. When cleared this bit masks out the terminal count interrupt of the relevant channel.
14	IE	Read/write	Interrupt error mask. When cleared this bit masks out the error interrupt of the relevant channel.
13—11	FlowCntrl	Read/write	Flow control and transfer type. This value is used to indicate the flow controller and transfer type. The flow controller can be the DMA controller, the source peripheral, or the destination peripheral. The transfer type can be either memory-to-memory, memory-to-peripheral, peripheral-to-memory, or peripheral-to-peripheral.
10—6	DestPeripheral	Read/write	Destination peripheral. This value selects the DMA destination request peripheral. This field is ignored if the destination of the transfer is to memory.
5—1	SrcPeripheral	Read/write	Source peripheral. This value selects the DMA source request peripheral. This field is ignored if the source of the transfer is from memory.

7 Call Processor (CP) Block (continued)

7.4 DMA Controller (DMAC) (continued)

Table 7.4-22 Channel Configuration Registers (DMACCxConfiguration) (continued)

Bit	Name	Type	Function
0	E	Read/write	Channel enable. Reading this bit indicates whether a channel is currently enabled or disabled: 0—Channel disabled. 1—Channel enabled. The channel enable bit status can also be found by reading the DMACEnbldChns register. A channel is enabled by setting this bit. A channel can be disabled by clearing the enable bit. This causes the current AHB transfer (if one is in progress) to complete and the channel is then disabled. Any data in the channels FIFO is lost. Restarting the channel by simply setting the channel enable bit has unpredictable effects and the channel must be fully reinitialized. The channel is also disabled, and channel enable bit cleared, when the last LLI is reached or if a channel error is encountered. If a channel has to be disabled without losing data in a channels FIFO the Halt bit must be set so that further DMA requests are ignored. The active bit must then be polled until it reaches 0, indicating that there is no data left in the channels FIFO. Finally, the channel enable bit can be cleared.

Table 7.4-23 describes the bit values of the three flow control and transfer type bits.

Table 7.4-23 Flow Control and Transfer Type Bits

Bit Value	Transfer Type	Controller
000	Memory to memory	DMA
001	Memory to peripheral	DMA
010	Peripheral to memory	DMA
011	Source peripheral to destination peripheral	DMA
100	Source peripheral to destination peripheral	Destination peripheral
101	Memory to peripheral	Peripheral
110	Peripheral to memory	Peripheral
111	Source peripheral to destination peripheral	Source peripheral

7 Call Processor (CP) Block (continued)

7.4 DMA Controller (DMAC) (continued)

The DMA controller can support up to 16 sources of peripheral DMA requests. Table 7.4-24 lists the mapping from the DestPeripheral/SrcPeripheral bits to the individual T8307 peripherals.

Table 7.4-24 DMA Mapping to T8307 Peripherals

Destination/Source Peripheral	Peripheral DMA Request
0	UART0 Tx
1	UART0 Rx
2	UART1 Tx
3	UART1 Rx
4	SD/MMC
5	Reserved
6	SIM Tx
7	SIM Rx
8	SSP0 Tx
9	SSP0 Rx
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved

7 Call Processor (CP) Block (continued)

Section 7.5 through Section 7.16 detail the CP-peripheral functions.

7.5 Programmable Interrupt Controller (PIC)

The PIC receives signals from 31 interrupt sources, groups and prioritizes them, and drives the two interrupt signals at the interface to the core. A list of features of the PIC follows:

- Thirty-one maskable interrupt inputs.
- Two programmable priority groups (IRQ, FIQ).
- Thirty-one programmable priority levels.

7.5.1 Operation

Figure 7.5-1 shows the block diagram of the interrupt controller.

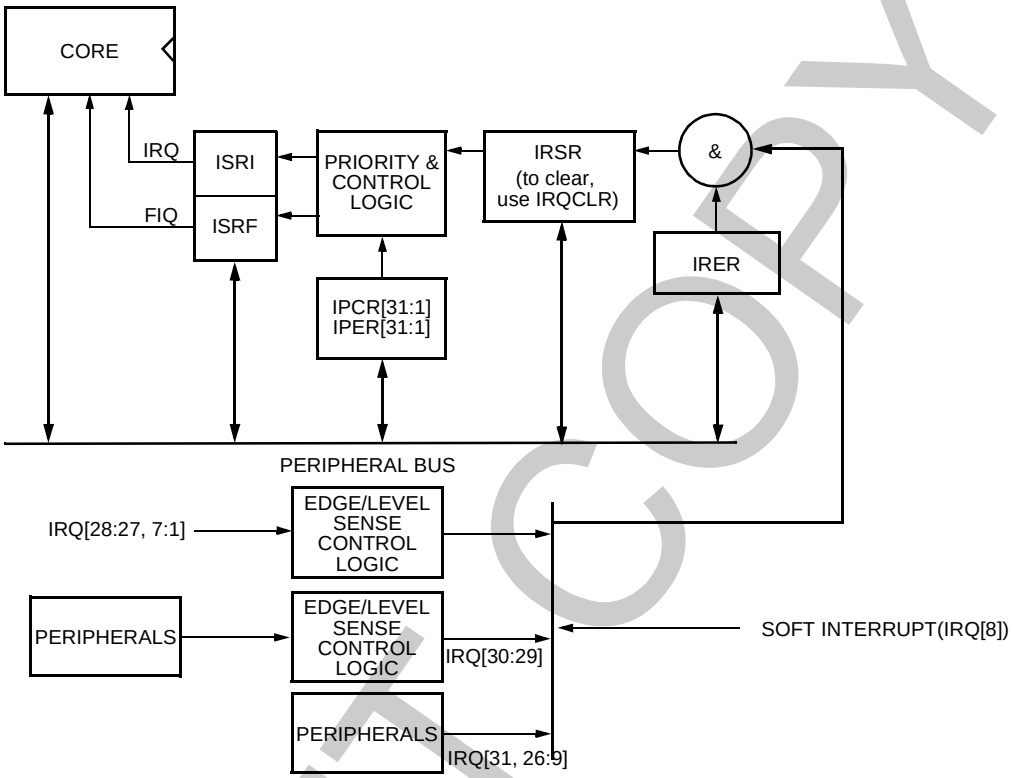
As shown in Figure 7.5-1, the interrupt controller receives as input 31 interrupt request signals: IRQ[31, 26:8], and fully programmable IRQ[30:27, 7:1]. The ordering of the IRQ signals is purely arbitrary and does not imply any relative priority. The interrupt request enable register (IRER) (see Table 7.5-7) provides a central point where the interrupts are enabled or disabled for the interrupt request status path. In particular, the interrupt signals on input lines IRQ[31:1] are logically ANDed with IRER[31:1], and the results are transferred to the interrupt request status register (IRSR) (see Table 7.5-8). At any time, the core reads the IRSR in order to check for pending interrupts.

The interrupt priority control registers (IPCRs) provide a means by which the relative priority of the interrupts is assigned programmatically. Each IPCR has an index field that contains the number of the interrupt assigned to that particular priority level. The IPCRs have an implicit priority ordering, where IPCR1 has the highest priority, and IPCR31 has the lowest priority. At RESET, all of the IPCRs are disabled.

The *ARM* core interface includes two maskable interrupt request inputs, IRQ and FIQ, where an active FIQ request preempts an active IRQ request. Each interrupt is assigned to either the IRQ group or the FIQ group by assigning a 1 (FIQ) or 0 (IRQ) to bit 5 of the corresponding IPCR register (see Table 7.5-3). Each group is handled independently. These inputs are referred to as core IRQ and core FIQ.

7 Call Processor (CP) Block (continued)

7.5 Programmable Interrupt Controller (PIC) (continued)



Key:
ISRI—Interrupt in-service register for core IRQ.
ISRF—Interrupt in-service register for core FIQ.
IPCR—Interrupt priority control register.
IPER—Interrupt priority enable register.
IRSR—Interrupt request status register.
IRQCLR—Interrupt request source clear register.
IRER—Interrupt request enable register.
IRQ—Interrupt request source.

Figure 7.5-1 Block Diagram of the Interrupt Controller

For FIQ and IRQ, the interrupt control logic determines the interrupt source to service next and sets the value for that interrupt in the interrupt in-service register (ISRI or ISRF) (see Table 7.5-1). The interrupt controller issues core IRQ or core FIQ, respectively, to the core. If an interrupt of higher priority is latched in the IRSR before the in-service register is read, the in-service register is updated with the value of the higher-priority interrupt. However, if the in-service register is read, the current register value is frozen until the corresponding bit in the IRSR register is reset to zero.

Prior to returning from the interrupt service routine, the interrupt is cleared from the interrupt in-service register by writing a 1 into the appropriate bit of the interrupt request source clear register (IRQCLR) (see Table 7.5-7).

The interrupt service routine also checks the IRSR for other pending interrupt requests and handles these interrupts before returning.

The IRQ request signals are mapped as shown in Table 6.3-1.

7 Call Processor (CP) Block (continued)

7.5 Programmable Interrupt Controller (PIC) (continued)

The 11 fully programmable interrupts are normally synchronized prior to processing. The interrupts are programmed to be edge-detect or level-sensitive and active-high or active-low. During the special CLOCKOFF power-down mode, they are set to be asynchronous and turn the clocks back on if they are asserted. On RESET, all interrupts are disabled, and all priority enable bits in the interrupt priority enable registers are on.

7.5.2 Registers

7.5.2.1 Interrupt In-Service Registers (ISRI and ISRF)

The interrupt in-service register (ISRI and ISRF) (see Table 7.5-1) contains the encoded value of the current highest-priority interrupt. Writes to the ISR are ignored. If reading the ISR, the current value is frozen until the corresponding interrupt is cleared in the IRSR. This register is set to all 0s on all reset conditions.

Table 7.5-1 Interrupt In-Service Registers, Addresses (ISRI 0x700C1094, ISRF 0x700C1098)

Bit	31—7	6—0
Name	RSVD	IIS

Bit	Name	Description
31—7	RSVD	Reserved.
6—0	IIS	Interrupt source. The encoded value of the interrupt source. Table 7.5-2 shows bit 6:0 encoding.

7 Call Processor (CP) Block (continued)

7.5 Programmable Interrupt Controller (PIC) (continued)

Table 7.5-2 Bit 6—0 Encoding

Bit 6—0	Interrupt Source	Interrupt Type	Comment
0000000	No Interrupt	—	—
0000100	IRQ1	IRQ1 Pin Interrupt.	Fully Programmable
0001000	IRQ2	IRQ2 Pin Interrupt.	Fully Programmable
0001100	IRQ3	IRQ3 Pin Interrupt.	Fully Programmable
0010000	IRQ4	IRQ4 Pin Interrupt.	Fully Programmable
0010100	IRQ5	IRQ5 Pin Interrupt.	Fully Programmable
0011000	IRQ6	IRQ6 Pin Interrupt.	Fully Programmable
0011100	IRQ7	Keyboard Interrupt.	Fully Programmable
0100000	IRQ8	Software Interrupt.	—
0100100	IRQ9	Reserved.	—
0101000	IRQ10	DMA Error Interrupt.	—
0101100	IRQ11	DMA Terminal Count Interrupt.	—
0110000	IRQ12	Programmable Timer Interrupt.	—
0110100	IRQ13	RTC Interrupt.	—
0111000	IRQ14	CP-Side SSP/I^2S (SSP0) Interrupt.	—
0111100	IRQ15	UART0 Interrupt.	—
1000000	IRQ16	UART1 Interrupt.	—
1000100	IRQ17	Reserved.	—
1001000	IRQ18	Reserved.	—
1001100	IRQ19	SIM Interrupt.	—
1010000	IRQ20	PIO Pin[7:0] I/O Interrupt.	—
1010100	IRQ21	PIO Pin[15:8] I/O Interrupt.	—
1011000	IRQ22	PIO Pin[23:16] I/O Interrupt.	—
1011100	IRQ23	PIO Pin[31:24] I/O Interrupt.	—
1100000	IRQ24	SD/MMC Interrupt 0.	—
1100100	IRQ25	PIO Pin[39:32] I/O Interrupt.	—
1101000	IRQ26	PIO Pin[47:40] I/O Interrupt.	—
1101100	IRQ27	ICP Interrupt.	Fully Programmable
1110000	IRQ28	UART Rx0 or Rx1 Pin Interrupt for Line Wake-up.	Fully Programmable
1110100	IRQ29	USB Core Interrupt.	Fully Programmable
1111000	IRQ30	USB Suspend Interrupt.	Fully Programmable
1111100	IRQ31	SD/MMC Interrupt 1.	—

7 Call Processor (CP) Block (continued)

7.5 Programmable Interrupt Controller (PIC) (continued)

7.5.2.2 Interrupt Priority Control Registers (IPCR1—IPCR31)

The interrupt priority control registers (IPCR) define the relative priority of each interrupt (see Table 7.5-3). The interrupt assigned to IPCR1 has the highest priority, and the interrupt assigned to IPCR31 has the lowest priority. Only interrupts that are assigned to IPCRs generate interrupts to the core. These registers are set to zero on RESET conditions.

Table 7.5-3 Interrupt Priority Control Registers (IPCR1—IPCR31), Addresses (0x700C1018—0x700C1090)

Bit	31—6	5	4—0
Name	RSVD	TYP	IS

Bit	Name	Description
31—6	RSVD	Reserved.
5	TYP	Interrupt type. Indicates which interrupt is driven to the core if this interrupt is active. If 1, the interrupt will be mapped to FIQ. If 0, the interrupt will be mapped to IRQ.
4—0	IS	Interrupt source. Assigns an interrupt to the priority control register. If 00000, there is no interrupt assigned to this priority level. If 00001, it corresponds to IRQ1. If 11111, it corresponds to IRQ31.

7.5.2.3 Interrupt Request Status Register (IRSR)

The interrupt request status register (IRSR) (see Table 7.5-4) indicates the status of the latched IRQ request inputs. The IRSR bits are set based on the value in the IRER register (see Table 7.5-5).

Table 7.5-4 Interrupt Request Status Register (IRSR), Address (0x700C1000)

Bit	31—1	0
Name	I*n**	RSVD

Bit	Name	Description
n	I*n*	IRQ*n* status. Indicates that an interrupt is active from interrupt request *n*. If 1, there is an active interrupt from interrupt source *n*. If 0, there is no interrupt from interrupt source *n*. This interrupt is cleared by writing a 1 to bit *n* of the IRQCLR.
0	RSVD	Reserved.

* Replace *n* with any bit from 1—31.

7 Call Processor (CP) Block (continued)

7.5 Programmable Interrupt Controller (PIC) (continued)

7.5.2.4 Interrupt Request Enable Registers (IRER)

The interrupt request enable registers (IRER) (see Table 7.5-5) enable or disable an interrupt request signal. Upon disabling an IRER bit, the corresponding bit in the interrupt request status register (see Table 7.5-4) is cleared. The enable register has a dual mechanism for setting and clearing the enable bits. This allows enable bits to be set or cleared independently, with no knowledge of the other bits in the enable register. To set the enable bits, a write is performed to the interrupt request enable set address. Each data bit that is set to one enables the corresponding interrupt. to clear the enable bits, a write is performed to the interrupt request enable clear address. Each data bit that is set to one disables the corresponding interrupt. IRER registers are set to all zero on all RESET conditions.

Table 7.5-5 Interrupt Request Enable Registers (IRER), Addresses (Clear 0x700C100C/Set 0x700C1008)

Bit	31—1	0
Name	E*n**	RSVD

Bit	**Name**	**Description**
n	E*n*	Interrupt *n* enable. Indicates if interrupt *n* is enabled or disabled. If 1, interrupt *n* is enabled. If 0, interrupt *n* is disabled.
0	RSVD	Reserved.

* Replace *n* with any bit from 1—31.

7.5.2.5 Interrupt Priority Enable Registers (IPER)

The interrupt priority enable registers (IPER) (see Table 7.5-6) enables or disables an interrupt source based on its priority level, as encoded in the IPCR registers. This simplifies the management of nested interrupt service routines by disabling lower-priority interrupts while enabling higher-priority interrupts relative to the current interrupt.

The IPER has a dual mechanism for setting and clearing the enable bits. This sets or clears enable bits independently, with no knowledge of the other bits in the IPER.

To set the enable bits, a write is performed to the IPER. Each data bit that is set to one enables the corresponding interrupt. To clear the enable bits, a write is performed to the IPCR. Each data bit that is set to one disables the corresponding interrupt. IPER register bits are set to all ones on all RESET conditions.

Table 7.5-6 Interrupt Priority Enable Registers (IPER), Addresses (Clear 0x700C10A4/Set 0x700C10A0)

Bit	31—1	0
Name	E*n**	FRZ

Bit	**Name**	**Description**
n	E*n*	Interrupt *n* enable. If 1, enables the interrupt request that has priority level *n*. If 0, disables the interrupt request that has priority level *n*.
0	FRZ	If bit 0 is 1, reading an in-service register causes the current value to be frozen until the corresponding interrupt is cleared in the IRQCLR register. If bit 0 is 0, the in-service register value is not frozen and may change if a higher-priority IRQ is asserted.

* Replace *n* with any bit from 1—31.

7 Call Processor (CP) Block (continued)

7.5 Programmable Interrupt Controller (PIC) (continued) (continued)

7.5.2.6 Interrupt Request Source Clear Register (IRQCLR)

The interrupt request source clear register (see Table 7.5-7) clears the service interrupt. Write a 1 to the corresponding bit to clear the source.

Note: This register reverts back to 0 upon the completion of the write.

Table 7.5-7 Interrupt Request Source Clear Register (IRQCLR), Address (0x700C109C)

Bit	31—1	0
Name	C*n* †	RSVD

Bit	Name	Description
n	C*n*	Clear interrupt *n*. A write of 1 to this bit clears interrupt *n*.
0	RSVD	Reserved.

† Replace *n* with any bit from 1—31.

7.5.2.7 Soft Interrupt Request Register (SOFTIRQ)

The soft interrupt request register (see Table 7.5-8) is used for programmed interrupts. A write to bit 0 of this register sets or clears a programmed interrupt. This register can also be cleared by writing a one to bit 8 of the IRQCLR register. Writing 1 to bit 8 of IRQCLR is the recommended way of clearing SOFTIRQ.

Table 7.5-8 Soft Interrupt Request Register (SOFTIRQ), Address (0x700C1010)

Bit	31—1	0
Name	RSVD	Soft Interrupt

Bit	Name	Description
31—1	RSVD	Reserved.
0	Soft interrupt	Soft interrupt. If 1, a soft interrupt is active. If 0, a soft interrupt is not active.

7 Call Processor (CP) Block (continued)

7.5 Programmable Interrupt Controller (PIC) (continued)

7.5.2.8 Fully Programmable Interrupt Control Registers (FPIRQC1—FPIRQC7, FPIRQC27—FPIRQC30)

The fully programmable interrupt control registers configure the IRQ1—IRQ7 and IRQ27—IRQ30 interrupt requests. Bit 0 to bit 3 are writable and readable; however, bit 4 is read only. Table 7.5-9 shows the format of fully programmable interrupt control registers.

Note: To avoid bogus interrupt being latched into IRSR register, the software should follow these steps when setting FPIRQCx register: disable the interrupt using IRER; program FPIRQCx; clear the interrupt using IRQCLR; enable the interrupt using IRER.

Table 7.5-9 Fully Programmable Interrupt Control Registers (FPIRQC1—FPIRQC7, FPIRQC27—FPIRQC30), Addresses (0x700C10A8—0x700C10C8, 0x700C10D8—0x700C10DC)

Bit	31—5	4	3	2	1	0
Name	RSVD	DAT	ASY	POL	SEN	ENA

Bit	Name	Description
31—5	RSVD	Reserved.
4	DAT	Interrupt data. A read-only copy of the data on the interrupt pin delayed by three clock cycles.
3	ASY	Asynchronous interrupt. Determines if the pin can cause an interrupt asynchronously. This functionality is only used in the CLKOFF powerdown mode. The fully programmable interrupts are always synchronized when not in this mode: If 1, the fully programmable interrupt is asynchronous. If 0, the fully programmable interrupt is synchronous.
2	POL	Interrupt polarity. Determines the polarity of the fully programmable interrupt. If 1, the fully programmable interrupt detects a low-to-high transition or high level. If 0, the fully programmable interrupt detects a high-to-low transition or low level.
1	SEN	Interrupt sense. Determines the sense of the interrupt. If 1, the fully programmable interrupt is transition-detect. If 0, the fully programmable interrupt is level-sensitive.
0	ENA	Interrupt enable. Determines if the fully programmable interrupt is enabled, and it disables the programmable I/O functionality on the pin if it is MUXed. If 1, the fully programmable interrupt is enabled. If 0, the fully programmable interrupt is disabled.

7 Call Processor (CP) Block (continued)

7.5 Programmable Interrupt Controller (PIC) (continued)

7.5.2.9 Slow-to-Fast Clock Select Register (SFCSEL)

This register selects interrupts that will cause the system clock to switch from slow to fast clock automatically. The slow to fast switching occurs when the interrupt is activated.

Table 7.5-10 Slow-to-Fast Clock Select Register (SFCSEL), Address (0x700C10CC)

Bit	31—1	0
Name	SFS	RSVD

Bit	Name	Description
31—1	SFS	Slow to fast interrupt selector. If 1, enables an interrupt to cause the system clock to switch from slow clock to fast clock automatically. If 0, does not allow an interrupt to cause the system clock to switch from slow clock to fast clock automatically Resets to all high.
0	RSVD	Reserved.

7.5.2.10 Bypass the Wait for Clock Counter Register (BPWFCC)

This register causes the wait for clock counter to be bypassed automatically when the corresponding interrupt is activated.

Table 7.5-11 Bypass the Wait for Clock Counter Register (BPWFCC), Address (0x700C10D4)

Bit	31—1	0
Name	BWC	RSVD

Bit	Name	Description
31—1	BWC	Bypass the wait for clock counter. This register enables or disables an interrupt to cause the wait for clock counter to be bypassed automatically. If the interrupt's corresponding bit equals 1, the wait for clock counter is bypassed when the interrupt is activated. If the interrupt's corresponding bit equals 0, the wait for clock counter counts down when the interrupt is activated.
0	RSVD	Reserved.

7 Call Processor (CP) Block (continued)

7.6 Parallel Peripheral Interface (PPI)

The PPI consists of six 8-bit ports of programmable I/O pins. A list of features of the PPI follows:

- Each bit is programmed as either an input or an output.
- Inputs are programmed to be level-sensitive or transition-detect.
- Outputs are programmed to be open-drain or direct-drive.
- Programmable polarity (inverted or not) for inputs and outputs.
- Edges (transitions) on any one of the inputs in the port cause a port-specific interrupt request to be asserted.
- Each I/O can be programmed to have an internal pull-up connected.

7.6.1 Operation

Figure 7.6-1 shows a block diagram of a single PPI port. Each PPI port controls eight I/O pins. Up to four PPI ports are connected together to share a 32-bit peripheral address range. Multiple address ranges are used if more than 32 pins are required. If multiple ports are connected, the data bits in each register become a byte of a 32-bit address. For example, bits[7:0] of byte 0 of each register control bits[7:0] of the 32-bit register; bits[7:0] of byte 1 control bits[15:8] of the 32-bit register. The functionality of each pin is programmed independently through the data direction, port sense, port polarity, port interrupt enable, and port pull-up enable registers. The port data set and clear addresses (see Table 7.6-3) are used to read input pins and to write output pins.

On T8307, there are six 8-bit ports of PPI. Ports 0—3 are controlled together in a 32-bit group with the base address 0x700C6000, called Group 1. Ports 4—5 are controlled by a 16-bit group with the base address 0x700D3000, called Group 2.

The data direction register (see Table 7.6-1) controls whether a corresponding bit is an input or an output. The port sense register (see Table 7.6-5) configures inputs as either level-sensitive or transition-detect, and outputs as open-drain or direct-drive. The port polarity register (see Table 7.6-7) allows both inputs and outputs to be inverted at the I/O pin. The port interrupt enable register (see Table 7.6-9) controls whether individual bits in a port can generate interrupts.

7 Call Processor (CP) Block (continued)

7.6 Parallel Peripheral Interface (PPI) (continued)

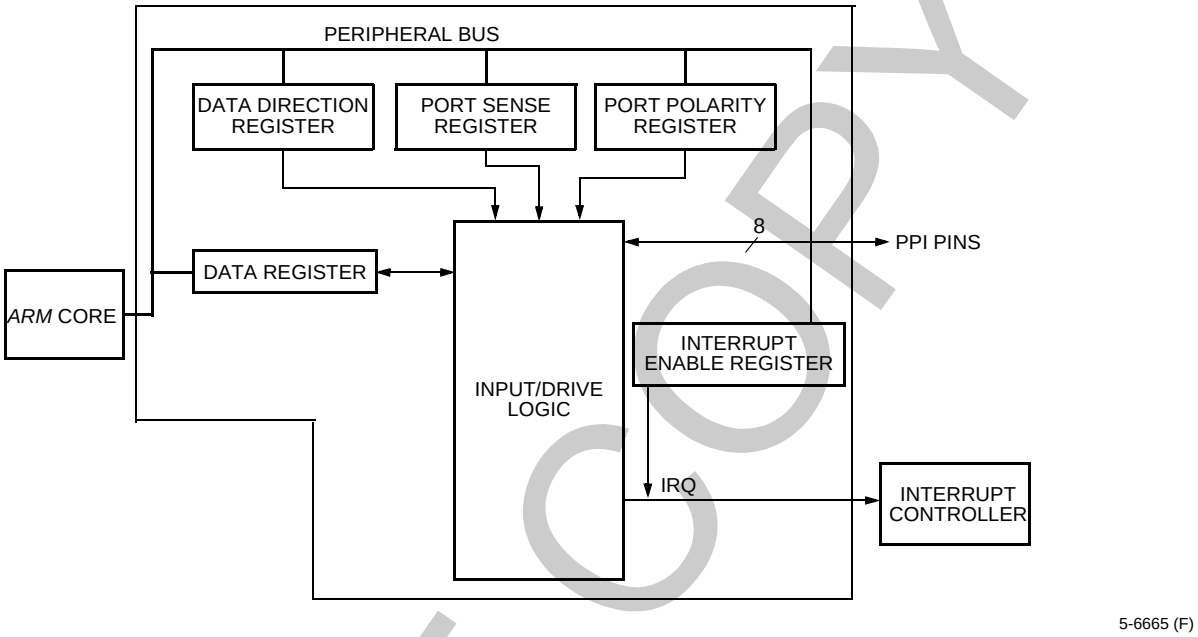


Figure 7.6-1 Block Diagram of One Byte of the Programmable Peripheral Interface

7.6.2 Pin Configuration on Reset

After reset, all PPI pins are configured as inverting level-sensitive inputs without pull-ups and enabled to cause interrupts.

7.6.3 Procedure for Writing to an Output Pin

1. Program the data direction register for the pin as an output.
2. Program the port sense register for the output as open-drain or direct-drive.
3. Program the port polarity register for the output as inverted or noninverted (relative to the port data register).
4. Write a value in the port data register using the port data clear address or the port data set address to specify the output level. If the corresponding port polarity register bit is 1, a 1 in the data register causes the output pin to drive high if it is programmed as a direct-drive output or causes the output pin to go to high impedance if it is programmed as an open-drain output. Conversely, if the corresponding port polarity register bit is 0, a 1 in the data register causes both direct-drive and open-drain output pins to drive low.

7 Call Processor (CP) Block (continued)

7.6 Parallel Peripheral Interface (PPI) (continued)

Regarding writes to the port data register for **input** pins:

- A write to a level-sensitive input has no effect.
- A write of 0 to a transition-detect input has no effect.
- A write of 1 to a transition-detect input (using the port data set address) clears the bit in the port data register to 0 with one exception. If the input is transition-detect and if the selected edge (selected in the port polarity register) occurs at the same time that a 1 is written to the bit in the port data register, the write is ignored and the register bit does not clear but is set or remains set.

7.6.4 Procedure for Reading from an Input Pin

1. Program the port data direction register for the pin as an input.
2. Program the port sense register for the input as level-sensitive or transition-detect.
3. Program the pull-up enable register if a pull-up resistor is desired on the I/O.
4. Program the port polarity register to indicate whether the level on the pin is inverted before going to the port data register (for level-sensitive inputs), or to indicate which edge results in a 1 appearing in the port data register (for transition-detect inputs).
5. If the input is changed to a transition-detect input, if the configuration of a transition-detect input is changed, or if the pin multiplexing control has changed, write a 1 to the bit in the data register to clear the bit. This clears the 1 in the bit that was left over from when the input was programmed as level-sensitive or that can result from transients during the configuration/multiplexing change.
6. Read the port data register by reading the port data clear address or the port data set address. It has the same effect as reading the port data register.

If the input is configured as level-sensitive, a high value on the pin is read as 1 in the port data register if the corresponding bit of the port polarity register is 1. Conversely, a low value on the input is read as 1 if the corresponding bit of the port polarity register is 0.

If the input is programmed to be transition-detect and the corresponding bit of the port polarity register is 1, a low-to-high transition on the pin registers a value of 1 in the corresponding bit in the data register. This value is changed to 0 by writing a 1 to that same bit in the port data register (using the port data set address), although if another low-to-high transition occurs at the same time that the 1 is being written, the register bit is not cleared but remains set.

If the input is programmed to be transition-detect and the corresponding bit of the port polarity register is 0, a high-to-low transition on the pin registers a value of 1 in the corresponding bit in the data register. This value is changed to 0 by writing a 1 to that same bit in the port data register (using the port data set address), although if another high-to-low transition occurs at the same time that the 1 is being written, the register bit is not cleared but remains set.

When the port data register is written using the port data set or clear address, only the chip pins configured as outputs are modified; those configured as inputs are unaffected. (However, note that writing the port data register for a transition-detect input clears the bits in that register even though the chip pin does not change.)

7 Call Processor (CP) Block (continued)

7.6 Parallel Peripheral Interface (PPI) (continued)

Input pins are asynchronous and are sampled at the system clock rate. In order for an input signal to be registered, it must have a minimum pulse-width of two system clock periods (see Figure 7.6-2). The CLK in this figure is the system clock as defined by the clock selected in the reset/power/clock management block.

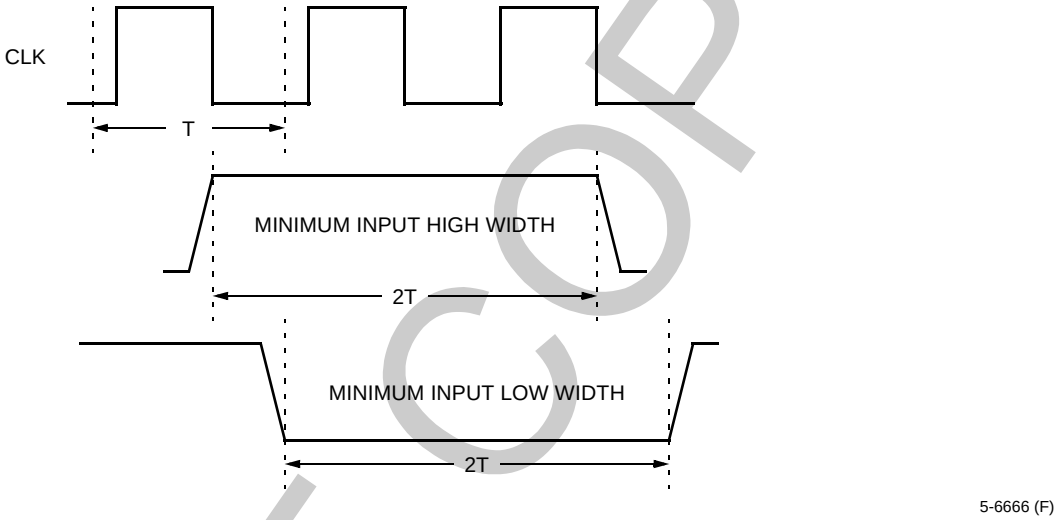


Figure 7.6-2 Minimum Input Pulse-Width Requirement for an Input Pin

7.6.5 Port Interrupts

Each PPI port contains logic to generate a port interrupt request when edges (transitions) occur on general-purpose input pins associated with the port. Port bits that are configured as general-purpose outputs or pins that are not enabled in the port interrupt enable register do not result in PPI interrupts.

The interrupt-activating edges on general-purpose input pins are described below.

If the pin is configured as a general-purpose level-sensitive input, the corresponding bit in the port interrupt enable register is set, and if the port's interrupt request signal is enabled in the interrupt controller, then either a low-to-high or high-to-low transition on the level-sensitive pin generates a port interrupt request. Note that even though the pin is level-sensitive, it is pin edges that generate the interrupt requests.

Notes: The pin's bit in the port's data register always reflects the level on the pin.

If the pin is configured as a general-purpose, rising-transition-detect input, the corresponding bit in the port interrupt enable register is set, and the port's interrupt request signal is enabled in the interrupt controller, then a low-to-high transition on the pin generates a port interrupt request. In addition, the pin's bit in the port's data register is set to 1.

If the pin is configured as a general-purpose, falling-transition-detect input, the corresponding bit in the port interrupt enable register is set, and the port's request signal is enabled in the interrupt controller, then a high-to-low transition on the pin generates a port interrupt request. In addition, the pin's bit in the port's data register is set to 1.

An interrupt request from a PPI port is cleared by writing a 1 to the port's bit in the interrupt controller's IRQ source clear register. However, if another interrupt-activating transition occurs on the pin simultaneously with the write to the IRQ source clear register, the write to the register is ignored and the port's interrupt request remains active.

7 Call Processor (CP) Block (continued)

7.6 Parallel Peripheral Interface (PPI) (continued)

Interrupts related to transition-detect inputs require two operations to clear evidence of the interrupt from the PPI:

- A write of 1 to the bit in the port data register (using the port data set address) is required in order to clear the register bit to 0. This is done first.
- A write of 1 to the port's bit in the IRQ source clear register is required in order to clear the port's interrupt request from the PPI.

If an interrupt is active for a PPI port, no other interrupt activity will be detected for that port until the interrupt is cleared. However, the port data register will continue to reflect activity on all port pins.

Note that the port data register reflects the state of level-sensitive PPI pins at the time the register is read, which may not be the state of the pins at the time an interrupt request was generated. For edge-sensitive inputs, the port data register does reflect past pin activity, or activity described above.

7.6.6 Registers

On T8307, there are six 8-bit ports of PPI. Ports 0—3 are controlled together in a 32-bit group with the base address 0x700C6000, called Group 1. Ports 4—5 are controlled by a 16-bit group with the base address 0x700D3000, called Group 2.

7.6.6.1 Port Data Direction Register (PPI1DIR, PPI2DIR)

The port data direction register (see Table 7.6-1) contains 1 bit for each of the general-purpose I/O pins. If a bit in the port data direction register is a one, the corresponding pin is an output; otherwise, it is an input.

Table 7.6-1 Port Data Direction Register (PPI1DIR), Address (0x700C6000)—Group 1

Bit	31—0
Name	PDDR[31:0]

Bit	Name	Description
31—0	PDDR[31:0]	Direction bits for PIO[31:0]. 1 = Output. 0 = Input.

Table 7.6-2 Port Data Direction Register (PPI2DIR), Address (0x700D3000)—Group 2

Bit	31—16	15—0
Name	RSVD	PDDR[47:32]

Bit	Name	Description
31—16	RSVD	Reserved.
15—0	PDDR[47:32]	Direction bits for PIO[47:32]. 1 = Output. 0 = Input.

7 Call Processor (CP) Block (continued)

7.6 Parallel Peripheral Interface (PPI) (continued)

7.6.6.2 Port Data Register (PPI1DATA, PPI2DATA)

The port data register (see Table 7.6-3) reads general-purpose input pins and writes general-purpose output pins. When a port data register is read, the bits configured as outputs reflect the value previously written to the register. The bits configured as inputs reflect the (possibly inverted) level on the input pin for level-sensitive inputs, or they reflect prior edge activity for transition-detect inputs.

When a new value is written to a port data register, the corresponding pins that are programmed as general-purpose outputs change to or stay at this value. Register bits configured as transition-detect inputs are set to zero if a 1 is written to the register bit. Register bits configured as level-sensitive inputs do not respond to writes to the register. Table 7.6-3 shows the format of the port data register.

The port data register is written by writing to either the port data clear address (0x700C601C for Group 1, 0x700D301C for Group 2) or the port data set address (0x700C6020 for Group 1, 0x700D3020 for Group 2). A write to the port data set address writes a 1 to selected bits of the port data register (those bits with a value of 1 during the write). The other bits of the port data register remain unchanged. A write to the port data clear address writes a 0 to selected bits of the port data register (those bits with a value of 1 during the write). The other bits of the port data register remain unchanged.

Note that use of the port data set address and port data clear address allows writing selected bits of the port data register using only one operation: a write to one of these two registers. No read-modify-write operations are necessary. The port data register can be read by reading either the port data set address or the port data clear address.

Table 7.6-3 Port Data Register (PPI1DATA), Addresses (Clear 0x700C601C/Set 0x700C6020)—Group 1

Bit	31—0
Name	PDAT[31:0]

Bit	Name	Description
31—0	PDAT[31:0]	Port data bits for PIO[31:0].

Table 7.6-4 Port Data Register (PPI2DATA), Addresses (Clear 0x700D301C/Set 0x700D3020)—Group 2

Bit	31—16	15—0
Name	RSVD	PDAT[47:32]

Bit	Name	Description
31—16	RSVD	Reserved.
15—0	PDAT[47:32]	Port data bits for PIO[47:32].

7 Call Processor (CP) Block (continued)

7.6 Parallel Peripheral Interface (PPI) (continued)

7.6.6.3 Port Sense Register (PPI1SEN, PPI2SEN)

The port sense register (see Table 7.6-5) configures general-purpose inputs as either level-sensitive or transition-detect, and outputs as open-drain or direct-drive. If a bit in the register is 0, the corresponding input pin is level-sensitive or the corresponding output pin is direct-drive if a 3-state I/O buffer is used. If a bit in the register is 1, the corresponding input pin is transition-detect or the output pin is open-drain when a 3-state I/O buffer is used. If an open-drain I/O buffer is used for a pin, that pin will be open-drain when it is an output, regardless of the setting of the port sense register bit.

Table 7.6-5 Port Sense Register (PPI1SEN), Address (0x700C600C)—Group 1

Bit	31—0
Name	PSEN[31:0]

Bit	Name	Description
31—0	PSEN[31:0]	Sense bits for PIO[47:32].

Table 7.6-6 Port Sense Register (PPI2SEN), Address (0x700D300C)—Group 2

Bit	31—16	15—0
Name	RSVD	PSEN[47:32]

Bit	Name	Description
31—16	RSVD	Reserved.
15—0	PSEN[47:32]	Sense bits for PIO[47:32].

7 Call Processor (CP) Block (continued)

7.6 Parallel Peripheral Interface (PPI) (continued)

7.6.6.4 Port Polarity Registers (PPI1POL, PPI2POL)

The port polarity register specifies inversion of both input and output signals at general-purpose pins. As a reference, logic signals in the port data registers are considered to be positive, or active-high. A value of 0 in a port polarity register causes a signal entering or leaving the device on the pin to be inverted, thereby conforming to a negative, or active-low signal convention outside the device. Conversely, a value of 1 in the register causes a signal entering or leaving the device on the pin to be simply buffered, thereby conforming to a positive, or active-high signal convention. The interpretation of the register bits differs somewhat for transition-detect inputs, as described in the following paragraphs.

For a level-sensitive input, a value of 1 in the port polarity register results in the value on the input pin being placed in the corresponding port data register (noninverted, level-sensitive input), while a value of 0 in the port polarity register results in the value on the pin being inverted before being placed in the corresponding port data register (inverted, level-sensitive input). The value in the port polarity register does not affect interrupt generation by level-sensitive inputs. Both edges of such inputs can generate an interrupt.

For a transition-detect input, a value of 1 in a port polarity register selects detection of a low-to-high transition at the pin (rising transition-detect input). Conversely, a value of 0 selects detection of a high-to-low transition at the pin (falling transition-detect input). The selected transition results in a 1 in the corresponding port data register and also triggers an interrupt if interrupts are enabled for the port in the interrupt controller.

For a direct-drive output, a 1 in the appropriate bit of the port polarity register results in the value in the port data register being driven to the chip pin (noninverted, direct-drive output), while a 0 in the appropriate bit of the port polarity register results in the inverse of the register value being driven to the pin (inverted, direct-drive output).

For an open-drain output, a 1 in the appropriate bit of the port polarity register results in the chip pin being driven to a 0 if there is a 0 in the corresponding port data register, and results in the chip pin going to high impedance if there is a 1 in the port data register (noninverted, open-drain output). For an open-drain output, a 0 in the appropriate bit of the port polarity register results in the chip pin being driven to high impedance if there is a 0 in the corresponding port data register and results in the chip pin being driven to 0 if there is a 1 in the port data register (inverted, open-drain output). Table 7.6-7 shows the format of the port polarity register. On reset, all bits of the port polarity register are cleared to 0, indicating inversion.

Table 7.6-7 Port Polarity Register (PPI1POL), Address (0x700C6010)—Group 1

Bit	31—0
Name	PPOL[31:0]

Bit	Name	Description
31—0	PPOL[31:0]	Polarity bits for PIO[31:0].

Table 7.6-8 Port Polarity Register (PPI2POL), Address (0x700D3010)—Group 2

Bit	31—16	15—0
Name	RSVD	PPOL[47:32]

Bit	Name	Description
31—16	RSVD	Reserved.
15—0	PPOL[47:32]	Polarity bits for PIO[47:32].

7 Call Processor (CP) Block (continued)

7.6 Parallel Peripheral Interface (PPI) (continued)

7.6.6.5 Port Interrupt Enable Register (PPI1IE, PPI2IE)

The port interrupt enable register (see Table 7.6-9) selects which bits of a port cause the port interrupt to be generated. If a bit in the register is 1, the bit is configured as an input and PPI interrupts are enabled for the appropriate bit in the interrupt controller, the pin generates interrupts based on how it is configured in the port sense and port polarity registers. On reset, bits of this register are set to 1, indicating interrupts are enabled for all bits.

Table 7.6-9 Port Interrupt Enable Register (PPI1IE), Address (0x700C6008)—Group 1

Bit	31—0
Name	PPIE[31:0]

Bit	Name	Description
31—0	PPIE[31:0]	Interrupt enable bits for PIO[31:0].

Table 7.6-10 Port Interrupt Enable Register (PPI2IE), Address (0x700D3008)—Group 2

Bit	31—16	15—0
Name	RSVD	PPIE[47:32]

Bit	Name	Description
31—16	RSVD	Reserved.
15—0	PPIE[47:32]	Interrupt enable bits for PIO[47:32].

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART)

The universal asynchronous serial communications controller (UART) provides an independent serial channel, which can operate in full-duplex mode. The following are the features of the UART:

- Full-duplex asynchronous communication.
- 32 bytes of FIFO for both receive and transmit.
- FIFO threshold interrupts.
- 1 start bit, 7 or 8 data bits, 1 optional parity bit, 1 or 2 stop bits.
- Programmable baud rate (17-bit system clock divider).
- Complete status reporting capabilities.
- Single interrupt routed to the PIC.
- Support for DMA transfers.
- Autoconfiguration mode with autobaud and autoformat operation.
- Hardware loopback for autoconfiguration mode.
- Character matching interrupts (up to three different characters).
- Modem support (RTS, CTS, DSR, DTR, DCD, RI) for DTE or DCE applications.
- Software flow control.

7.7.1 Operation

As shown in Figure 7.7-1, the function of the UART is to convert incoming serial data on the Rx line to parallel data for the CPU, and convert parallel data from the CPU to serial data on the Tx line. The baud rate and byte format are selected by direct programming of the baud rate and control register and/or by using the autoconfiguration mode. The autoconfiguration mode does not completely depend on hardware to pick the baud rate. There is a set of programmable registers that assists the hardware in selecting optimal values for higher baud rates. The status of the transmitter and receiver FIFOs and autoconfiguration operation may be used to generate interrupts. The controller uses two main modes of operation: normal operation or autoconfiguration. The normal mode operation is used after the software configures the format or after the autoconfiguration mode terminates.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

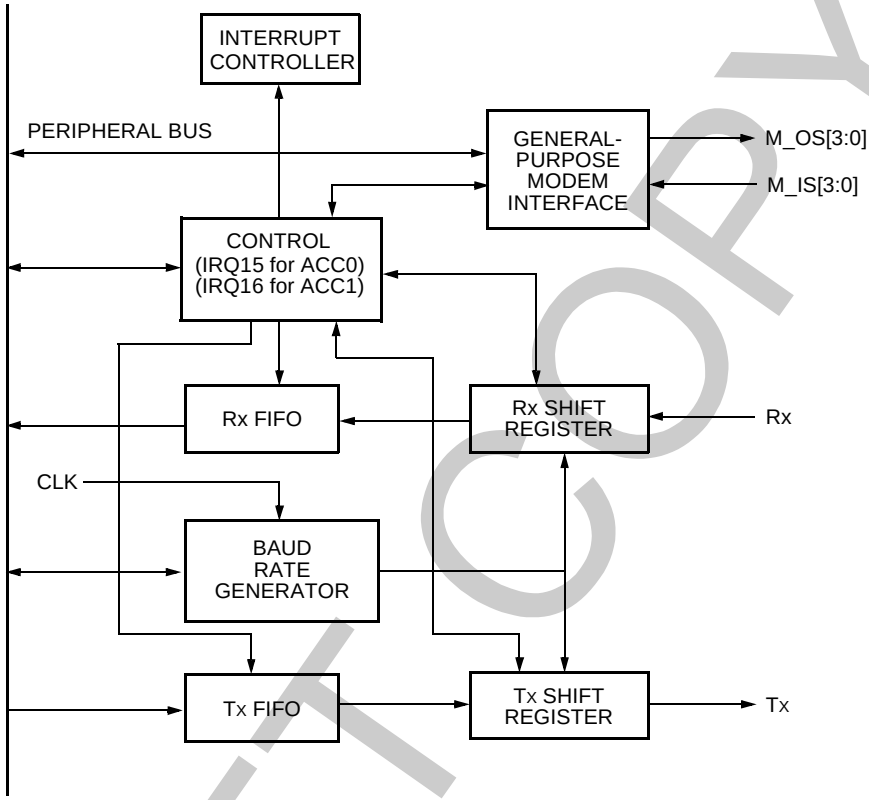


Figure 7.7-1 Block Diagram of the Asynchronous Serial Communications Controller

7.7.1.1 Normal Operation After Configuration

In order to transmit data on the transmit line Tx, the baud divisor register is set, the transmitter control register is set, and then data is written into the transmitter FIFO. The data from the transmitter FIFO is transferred to the transmitter shift register. A start bit is generated, and then the data is shifted to the output one bit at a time at the rate determined by the UART clock rate and the value programmed in the baud divisor register. The number of data bits can be programmed between 7 bits—8 bits. The optional parity bit is then generated, followed by 1 or 2 stop bits. When the number of bytes remaining in the transmitter FIFO is less than or equal to the transmitter FIFO threshold programmed in the transmitter control register, the transmitter FIFO threshold bit is set in the UART status register. An interrupt can also be generated on this condition if the interrupt is enabled in the transmitter control register.

To receive serial data from the Rx input pin, set up the baud rate, and set the receiver control register. At the appropriate time after a start bit is detected, the data on the Rx line is shifted into the receiver shift register. This is done by delaying one-half baud period from the beginning edge of the start bit, waiting for 1 baud period, and then sampling each data bit in the center of its ideal bit time. The number of data bits expected can be programmed for 7 bits—8 bits. After shifting one character and the optional parity bit into the receiver shift register, the data is tested for a parity error and the data plus the error flag are transferred to the receiver FIFO. When the number of bytes in the receiver FIFO exceeds the receiver FIFO threshold programmed in the receiver control register, the receiver FIFO threshold bit is set in the UART status register. An interrupt can also be generated on this condition if the interrupt is enabled in the receiver control register. If the controller detects receive errors, it sets appropriate error bits in the status register and will generate an interrupt if the receiver control register enables the corresponding interrupt.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

Additional status information associated with character recognition and conditional idle timing is also available and can be used as an interrupt source. A single interrupt line is used to generate interrupt to the CPU. The interrupt type can be read from the status register.

7.7.1.2 Modem Interface

Support for standard 6-pin modem interfaces is available through the general-purpose modem interface registers. The six modem pins (data terminal ready, request to send, ring indicator, data carrier detect, data set ready, and clear to send) can be implemented for DTE or DCE applications using these general-purpose registers.

This general-purpose register can support up to six pins with either four input (two output) pins or four output (two input) pins, depending on the number of pins bonded out. The output register will drive output pins with the inverted value from the register. The input data register, when read, provides the inverted value of the associated input pin and also detects logic changes on the input pin. The changes on the pin must be longer than the clock period supplied to the UART to guarantee detection. Any of the input ports can be assigned to any of the standard modem input pins, and any of the output pins can be assigned as a standard modem pin output.

7.7.1.3 Autoconfiguration Mode

This UART also supports an autoconfiguration mode that performs autobaud and autoformat operation. Autobaud mode is always used in conjunction with the autoformat modes, although under some conditions, the configuration will terminate with only the autobaud operation complete. The autobaud operation allows automatic setting of the baud rate and the autoformat operation supports automatically setting the baud rate, character size (7 bits or 8 bits), parity configuration, and the number of stop bits. The autobaud and autoformat operation starts in the configuration mode, and after detecting the configuration input sequence, switches to a normal mode. These modes allow the hardware to automatically set some of the hardware configuration values given certain restrictions. The autobaud and autoformat operations require that special character sequences must appear on the Rx ports in order to properly detect the correct configuration. Autoformat mode requires receiving either the AT or the aT two-character sequence. The special sequence required for autobaud must come from the following set of two character sequences: AT, at, A/, or a/. The automatic configuration may complete with only the autobaud operation complete and no automatic format operation, depending upon the input sequence. The automatic configuration will complete upon receipt of any of the four sequences used in autobaud operation; therefore, only the baud rate measurement will complete if the A/ or a/ sequence is received during configuration.

The automatic configuration hardware will reject improper sequences if it can detect that the input sequence is improper and does not match the correct set of configuration characters. An invalid initialization sequence can cause the configuration logic to make an incorrect baud rate measurement, and the detected input sequence may not match what was actually sent. The probability that an improper sequence is misidentified as a valid sequence is minimal.

It is also possible that a valid sequence could be missed if it follows too soon after an invalid sequence. For example, a character of value of 0x0 with even parity and 8-bit character length would yield a temporary baud period measurement of 10 times the correct period and therefore, the actual character would end long before the receive logic finished sampling the receive input. This sequence would be rejected, but until the receive logic finished sampling at the wrong rate, any new data would be read incorrectly.

The automatic configuration will respond to the input sequences in the following ways: autobaud operation and autoformat operation.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

7.7.1.4 Autobaud Operation

The following list shows the basic steps used in automatically selecting the baud rate:

- The measured baud period is continually updated to the width of the first logic 0 pulse that it detects (assuming that the pulse-width is greater than a minimum threshold and less than a maximum) until a valid sequence is detected. A continual update of the pulse-width of the start bit of the first character is used to detect valid configuration sequences after having received and rejected an invalid one. However, the logic will wait until it has finished the current character before it can look for another start bit to set the baud rate.
- The configuration logic will continue to hunt for the correct baud rate until the proper input sequence, AT, at, A/, or a/, is detected. When the proper input sequence is detected, the baud rate measurement will stop. While the autoconfiguration logic is running, it uses the baud rate derived from the first start bit to detect the configuration sequence.
- The measured clock divisors value may not be accurate enough to use directly for high baud rates. For higher baud rates; however, there is a set of programmable registers that allows the hardware to convert the measured values into prespecified values that should be closer to the desired baud rate divisors.
- The hardware will directly use the measured baud rate divisor or it will select the converted baud rate divisor from the programmable registers. Also, software can always write the baud rate divisor if so desired; however, it would not normally write the divisor register during autobaud operation.
- The autobaud measurement completion depends only on the character sequence received and not the extra bits used for autoformatting. In other words, it is possible to receive a valid input sequence that provides ambiguous format information while still giving a valid baud rate. However, in some cases, the logic will detect that the received pattern cannot be a valid set of characters and will reject the sequence in those cases.
- The measured value of the baud rate must also fall between a minimum and a maximum threshold. If not, the baud measurement is considered invalid and the measurement is rejected.
- If the logic detects a bad character sequence or baud error, status bits are set in the autoconfiguration, and if enabled, an interrupt will be set.

7.7.1.5 Autoformat Operation

The autoformat operation includes the autobaud operation described in Section 7.7.1.4, along with the following list of operations:

- The logic value in the eighth—twelfth bit positions (bits[11:7]) after the first start bit, and the eighth—tenth bit positions (bits[9:7]) after the second start bit are stored and used for calculating the data format settings.
- If one of the strings A/ or a/ is received, an interrupt will be generated if the interrupt enable bit in the autoconfiguration register is set, but no format changes will be made. The Rx control will be updated to set the Rx enable to active and the autoconfiguration loopback will be disabled.
- If either of the strings AT or at is received and the parity and stop bits patterns fit into a recognizable category, an interrupt will be generated if the interrupt enable bit in the autoconfiguration register is set, and the format will set according to the measurement. The Tx and Rx control register will automatically be updated to autoconfiguration loopback stop. Also, force loopback in the Rx control register must be cleared by software if it was already set.
- If the logic detects a bad character sequence or baud error, status bits are set in the autoconfiguration, and if enabled, an interrupt will be set. The autoconfiguration will continue to look for a valid sequence.

Figure 7.7-2 shows an example of the possible receive input sequences that the hardware can use for autoformatting. Twenty-four sequences are shown; the first 12 are 7-bit formats and the second group of 12 show 8-bit data. Each of the two main groups is further grouped into cases as follows:

- Groups of three: no parity, odd parity, and even parity.
- Four cases of stop bits[4:1]. In addition to checking for the correct character string, several of the bit positions at the end of the byte are examined.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

Figure 7.7-2 highlights bits[11:7] after the start bit of the first character. Bits[9:7] of the second character are also stored to determine the byte format. Figure 7.7-2 shows the bits in the input sequence that are stored in a register (format data register) and used for determining the format. The register can be read by software though the software can override any of the format settings. Table 7.7-1 lists the formats associated with the valid values of the format data register.

7.7.1.6 Special Considerations for the Loopback Features

The autoconfiguration loopback, the receive force loopback, the transmit control logic, and the receiver control logic all interact when the loopback features are used. The following list provides a general description of the effects of the enabling the loopback feature on the transmit and receive operations:

- The autoconfiguration loopback takes the highest precedence, and its effect is immediate and causes the current character transmission to abort.
- The transmit state machine will not attempt to start new data while the autoconfiguration loopback is active, even if the transmit disable control bit is not set in the transmit control register and data is available in the Tx FIFO.
- The receive state machine will not attempt to receive new data while the autoconfiguration loopback is enabled only because the autoconfiguration mode does not allow writes to the Rx FIFO, and the autoconfiguration loopback clears automatically when the autoconfiguration mode completes.
- The force loopback enable in the receiver control register does not prevent the Rx FIFO from receiving data.
- The force loopback will not loop back data onto the Tx port from the Rx port until the Rx state machine is idle.
- The force loopback will cause the transmitter to abort any current transmissions.
- The force loopback will prevent the transmit control logic from starting a new character transmission.
- Ending the force loopback takes effect immediately regardless if a character is being received.
- The transmitter state machine cannot unload a new character from the Tx FIFO and start a new transmission until the transmit disable control bit is not set, the Tx FIFO is not empty, and the autoconfiguration loopback and force loopback are both inactive.
- Setting the transmission disable bit will not interrupt a character that is in transmission.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

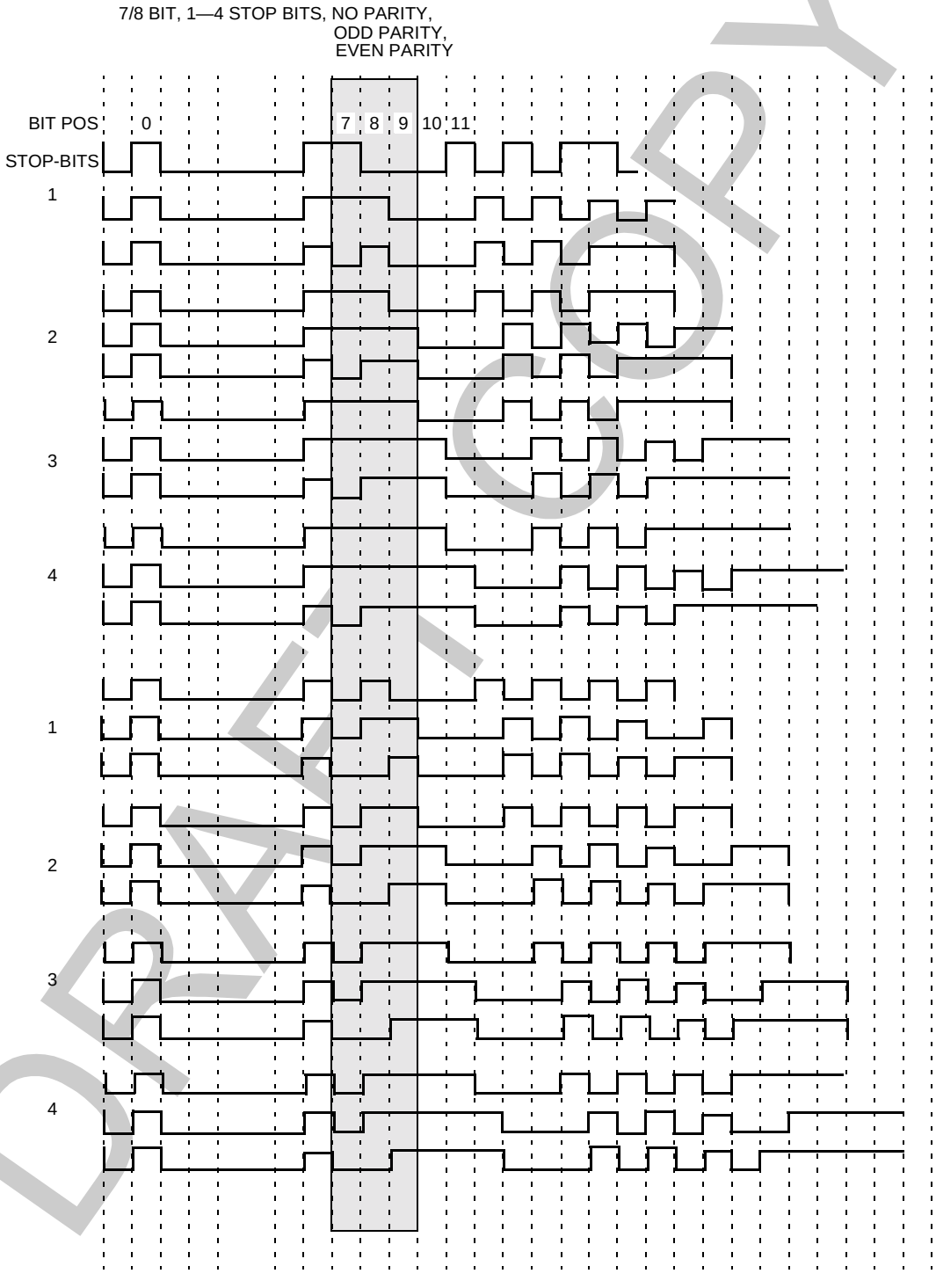


Figure 7.7-2 Possible AT Sequences with Groups, Three Parity Cases: None, Odd, and Even

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

7.7.1.7 Break Characters

The UART supports the generation and detection of break characters. The break character is generated by setting a specific bit in the transmit control register; see Table 7.7-27 for a detailed description on how to generate a break character. When the receiver circuit detects break characters, a special pattern is written into the Rx FIFO (Table 7.7-29 and Table 7.7-30), the UART status register (Table 7.7-19) is updated, and an interrupt is generated if the appropriate enable bit in the receiver control register (Table 7.7-21) is set.

7.7.1.8 Interrupt Support

The UART provides a single interrupt source to the device's interrupt controller; however, within the UART are several interrupt sources. The sources of the UART interrupt can be put into two categories: event sources and condition sources. The main difference in the sources is the manner in which the interrupt source is cleared. Besides clearing the associated interrupt enable, the interrupt source can be cleared by either reading the appropriate register or by changing the condition that is causing the interrupt. The event-sourced interrupts are based upon one-time events, and the interrupt source is cleared when the UART status register is read or the autoconfiguration register is read. The condition sourced interrupts are associated with the current state of the UART and the interrupt source cannot be cleared by reading any status register, but by changing the condition associated with the interrupts.

7.7.1.9 DMA Support

The UART provides two ready signals to the DMA controller, one for transmit and the other for receive. The transmit ready signal is asserted when the transmit FIFO is empty. The receive ready signal is asserted when the receive FIFO has at least one valid character in it (it is not empty). The DMA controller must be programmed to use the desired ready signals when it is set up.

7.7.1.10 Operation On Reset

Upon any reset, the UART performs the following:

1. All ongoing transfers are aborted.
2. Both transmitter and receiver FIFOs are reset.
3. The autoconfiguration control register is reset to all 0s.
4. The transmitter control register is reset to all 0s to disable transmitter FIFO interrupts and disable transmitter parity generation.
5. The receiver control register is reset to all 0s to disable receiver FIFO interrupts, disable receiver parity checking, and disable receiver error interrupts.
6. The UART status register is reset to all 0s.
7. The FIFO status register is set to reflect the current status of both transmitter and receiver FIFOs (empty).
8. The baud divisor register is reset to all 0s.

7.7.1.11 External Interface

The external interface of the UART may be connected to standard serial interface drivers/receivers (e.g., RS-232, RS-422). A start bit is transmitted using a low output signal, while a stop bit is transmitted using a high signal. Figure 7.7-3 shows a timing diagram for a single character with a parity bit.

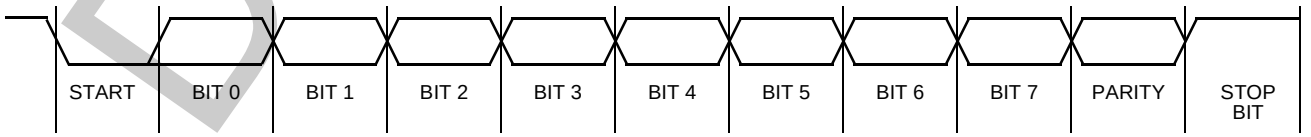


Figure 7.7-3 UART Transmit Timing Diagram

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

7.7.1.12 Rx Line Interrupt

The Rx input pins of ACC0 and ACC1 have the capability of generating line wake-up interrupt (see Figure 7.7-4). When enabled, a low on either ACC Rx port will drive the IRQ28 interrupt request low. During the CP block wait-for-interrupt period, an external device connected to ACC0 or ACC1 can use this feature to wake up the system by simply starting the serial transmission.

Note that IRQ28 must be programmed as either active-low or falling edge sensitive in programmable interrupt controller (PIC) module.

IRQ28 is designed for UART line wake-up interrupt purpose only. It should not be used as a general-purpose external interrupt request input.

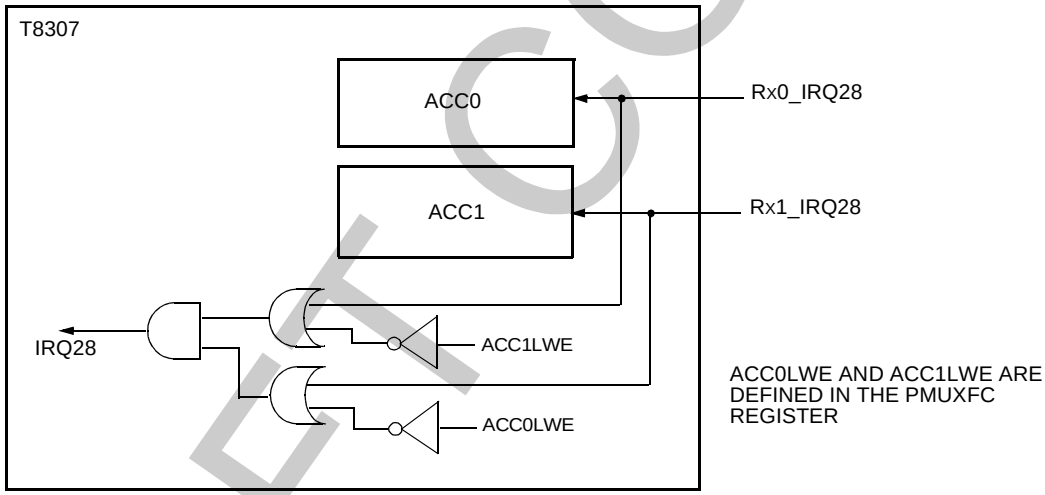


Figure 7.7-4 ACC0 and ACC1 Rx Line Interrupt

7.7.2 Registers

Table 6.2-1 lists the registers and their addresses of the UART. UART_BASE_ADDR is 0x700C8000 for ACC0; UART_BASE_ADDR is 0x700C9000 for ACC1.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

7.7.2.1 Autoconfiguration Control Register (ACCAC)

The autoconfiguration register is used by software to start the autoconfiguration mode and to provide status bits showing the input sequence status.

Table 7.7-1 Autoconfiguration Control Register (ACCAC), Address (UART_BASE_ADDR + 0x024)

Bit	31—17	16	15	14	13	12	11	10—9	8	7—3	2—0
Name	RSVD	AC_En	AC_IE	AC_LBE	AC_Err_IE	AC_Done	Baud_Err_St	Ac_Cmd_Err	Vld_F	F_Stat	S_Stat

Bit	Name	Description
31—17	RSVD	Reserved.
16	AC_En	Autoconfiguration enable control. When this bit is set to a logic 1, the UART enters into the autoconfiguration mode. This bit can be set by writing a 1 to this bit and it can be cleared by writing a 0 to it.
15	AC_IE	Autoconfiguration interrupt enable. When this bit is set to a logic 1, and the configuration done bit in the status register is set, the UART will assert an interrupt. When this bit is set to a 0, no interrupt will be generated when the configuration done bit in the status register is set.
14	AC_LBE	Autoconfiguration loopback enable. When this bit is set to a logic 1, the signal on the Rx input pin is looped back to the Tx output pin after certain conditions are met. The transmitter logic immediately aborts any current transmission (and does not reload the Tx FIFO); however, the actual loopback will not occur until the receiver state machine returns to the idle state if it is not already there. When this control bit is a logic 0, the Tx pin is controlled by transmitter control logic and the receiver force loopback logic. This mode will self-clear when the autoconfiguration operation completes with the autobaud operation complete, the autobaud and autoformat operation complete, or the AC_En bit is cleared by software.
13	AC_Err_IE	Autoconfiguration error interrupt enable. When this bit is set to a logic 1, the UART is allowed to generate an interrupt if the Baud_Err_St flag is active or the AC_Cmd_Err flags indicate a command error was detected. When set to a logic 0, these error status bits can still be set; however, they cannot cause an interrupt.
12	AC_Done	The autoconfiguration done flag. This bit sets to a 1 after autoconfiguration is complete. When the appropriate bit in the interrupt enable register is set, this bit will cause an interrupt when it becomes active (logic 1). This bit cannot be set by software, but is cleared automatically when the autoconfiguration register is read by software.
11	Baud_Err_St	Baud error state. This bit is set by the hardware when the autobaud logic measures a baud period that is out of range (either too high or too low). This bit cannot be written by software, but it will automatically clear itself to a logic 0 when this register is read by the CPU.
10—9	AC_Cmd_Err	Autoconfiguration command error. These bits indicate that the autoformat logic detected a bad format or bad character. When these bits are set to 01 or 10, the autoconfiguration logic recognizes the first 7 bits of two characters as a command sequence; however, the logic does not recognize the trailing bit patterns as consistent with any of the supported character formats. When these bits are 11, the logic has detected an invalid character format or has detected an unexpected character without seeing a valid command pattern within the first 7 bits of two characters. These bits cannot be set by software, but are cleared automatically when the autoconfiguration register is read by software.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

Table 7.7-1 Autoconfiguration Control Register (ACCAC) (continued)

Bit	Name	Description
8	Vld_F	Valid format. This bit tells whether the automatic configuration mode was able to determine the data format. A logic 1 indicates a successful autoformat. If an A\ or a\ sequence is received, this bit will not set. This bit is not valid until the autoconfiguration done flag is set. This bit cannot be set by software, but is cleared automatically when the autoconfiguration register is read by software.
7—3	F_Stat	First byte status bits. These are the indicating bits shifted in (bits[11:7] with bit 0 the first bit after the start bit) with the first byte of the automatic configuration sequence. Table 7.7-2 gives valid combinations of this bit and the other bits used to determine the correct data format. These bits are not valid unless the autoconfiguration done flag is set, and also, these bits cannot be cleared by software.
2—0	S_Stat	Second byte status bits. These are the indicating bits shifted in (bits[9:7] with bit 0 the first bit after the start bit) with the second byte of the automatic configuration sequence. Table 7.7-2 gives valid combinations of this bit and the other bits used to determine the correct data format. These bits are not valid unless the autoconfiguration done flag is set, and also, these bits cannot be cleared by software.

Table 7.7-2 shows the configuration taken for the given values in bits[7:0] of the autoconfiguration register and when a valid autoformat sequence is received. Table 7.7-2 shows information from the received sequence that is sufficient to uniquely identify the format, assuming that the sequence itself is valid. The dashes in the first column represent states that do not have to be identified to make the sequence unique for a given format; however, if the state represented by the dash has a required value, the hardware will check it when it checks the sequence for validity. The x in the second column represents true don't care values.

Table 7.7-2 Unique Autoformat Responses That Identify Format

Bits[7:3] (Bits[11:7] of First Character)	Bits[2:0] (Bits[9:7] of Second Character)	Char Size (7/8 Bits)	Stop Bits 1, 2, + (More Than 2)	Parity: N—None O—Odd, E—Even	
				at	AT
100--	1xx	7	1	N	N
1100-	01x	7	1	E	O
010--	11x	7	1	O	E
1100-	11x	7	2	N	N
1110-	011	7	2	E	O
0110-	111	7	2	O	E
111--	111	7	+	N	N
1111-	011	7	+	E	O
0111-	111	7	+	O	E
010--	01x	8	1	N	N
0110-	001	8	1	E	O
0010-	011	8	1	O	E
0110-	011	8	2	N	N
01110	001	8	2	E	O
00110	011	8	2	O	E
0111-	011	8	+	N	N
01111	001	8	+	E	O
00111	011	8	+	O	E

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

7.7.2.2 Baud Divisor Register (ACCBDR)

The baud divisor register is used to divide the system clock to generate different baud rates. The baud rate generator is 17 bits wide; hence, division factors of 1—131,072 can be programmed.

Table 7.7-3 Baud Divisor Register (ACCBDR), Address (UART_BASE_ADDR + 0x058)

Bit	31—17	16—0
Name	RSVD	Baud_Div

Bit	Name	Description
31—17	RSVD	Reserved.
16—0	Baud_Div	Baud divisor. Bits[16:0] specify the baud rate divisor. The divisor is 1 for a value of all 0s in bits[16:0], and 131,072 for a value of all 1s in bits[16:0]. These bits can be automatically set in the automatic configuration mode, or written by software. This value resets to all 0s. The actual minimal useful value in this register depends upon the UART internal clock frequency and the exact value of the possible baud rate.

The following equation gives the baud clock divisor value for a given baud rate:

CLK: ACC clock in MHz.

BR: baud rate in bits/s.

BDF: baud clock division factor (1—131,072).

BDF = ((CLK x 10^6)/BR).

The value written to the baud rate generator should be ((BDF rounded to the nearest integer) – 1).

The maximum baud rate depends on the clock frequency and the accuracy of the programmed rate compared with the desired baud rate. It is possible that the clock rate will not be an integer multiple of the ideal baud rate and, therefore, there will be a difference between the desired rate and the programmed rate.

The following equation gives the error in sampling the input data. There are two equations and the equation that gives the worst error is the one that should be chosen.

| 100% * 12 * (BDF (actual)/BDF (ideal) – 1) + 2/(BDF (ideal) | or

| 100% * 12 * (BDF (actual)/BDF (ideal) – 1) |

Example:

Desired baud rate = 721,000

System clock rate = 26 MHz, desired BDF = 36.061, actual BDF = 36

Total error ~ 2.0%.

This particular case comes out particularly well because of the small difference in the actual baud divisor and the desired baud divisor. Table 7.7-4 shows the cumulative error in the sample location due to the difference in baud rate between the ideal BDF derived value and actual BDF value. The total error column shows the total error by the time the second stop bit is sampled.

The last term accounts for the one clock uncertainty caused by sampling an asynchronous signal. The fourth column uses the same equation except that the ideal BDF is off by ±1. This case represents the potential error that could occur during the autobaud operation, where the measured baud rate deviates by ±1 clock cycle. As shown in Table 7.7-4 and Table 7.7-6, the error rate for high baud rates is unacceptable without some adjustments.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

Table 7.7-4 shows various baud rates, the UART clock division associated with the baud rate and the percentage difference between the desired and actual baud rate. The actual baud rate will deviate from the desired baud rate in some cases because the UART clock is not an integer multiple of all of the example baud rates.

Table 7.7-4 Sample Baud Rates for 60 MHz UART Clock

Desired Baud Rate	Desired BDF	Actual BDF	Baud Rate Deviation (%)
721000	83.218	83	0.262
460800	130.208	130	0.160
230400	260.417	260	0.160
153600	390.625	391	–0.096
115200	520.833	521	—0.032
76800	781.250	781	0.032
57600	1041.667	1042	–0.032
38400	1562.500	1562	0.032
19200	3125.000	3125	0.000
14400	4166.667	4167	–0.008
9600	6250.000	6250	0.000
7200	8333.333	8333	0.004
4800	12500.000	12500	0.000
3600	16666.666	16667	–0.002
2400	25000.000	25000	0.000
1800	33333.332	33333	0.001
1200	50000.000	50000	0.000
600	100000.000	100000	0.000
300	200000.000	200000	0.000

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

Table 7.7-5 Sample Location Error Various Baud Rates for 26 MHz System Clock

Desired BR	Desired BDF	Actual BDF	Total Error Actual	Total Error for Actual BDF ± 1
721000	36.1111	36	2.03	34
460800	56.4236	56	9	29.7
230400	112.8472	113	1.6	12.4
153600	169.2708	169	1.9	9
115200	225.6944	226	1.6	7
76800	338.5417	339	1.6	5.2
57600	451.3889	451	1	3.7
38400	677.0833	677	0.15	1.9
19200	1354.1667	1354	0.15	1
14400	1805.5556	1806	0.30	0.96
9600	2708.3333	2708	0.15	0.30
7200	3611.1111	3611	0.04	0.30
4800	5416.6667	5417	0.07	0.30
3600	7222.2222	7222	0.04	0.20
2400	10833.3333	10833	0.04	0.14
1800	14444.4444	14444	0.04	0.12
1200	21666.6667	21667	0.02	0.07
900	28888.8889	28889	0.005	0.05
600	43333.3333	43333	0.01	0.02
300	86666.6667	86667	0.005	0.02

Also, the sample clock may be delayed by 1 UART clock period because of the uncertainty in detecting the start bit. The actual sample will start early by 1/2 of a UART clock for odd values of the programmed baud rate. This occurs because the sampling logic counters must first divide the programmed value by 2 in order to locate the center of the start bit. The difference in the programmed baud period and the desired baud period, ERROR, must be added for each bit sampled after the start bit. Thus, the total error in locating the sample point of the second stop bit would be approximately (for 1 start bit, 8 bits of data, 1 parity and 2 stop bits):

Total error = 12 x ERROR + 1 clock cycle.

The following equation gives the error in sampling the input data. There are two equations and the equation that gives the worst error is the one that should be chosen.

| 100% * 12 * (BDF (actual)/BDF (ideal) – 1) + 2/(BDF (ideal) | or

| 100% * 12 * (BDF (actual)/BDF (ideal) – 1) |

Example:

Desired baud rate = 721,000

System clock rate = 26 MHz, desired BDF = 36.061, actual BDF = 36

Total error ~ 2.0%.

This particular case comes out particularly well because of the small difference in the actual baud divisor and the desired baud divisor. Table 7.7-4 shows the cumulative error in the sample location due to the difference in baud rate between the ideal BDF derived value and actual BDF value. The total error column shows the total error by the time the second stop bit is sampled.

The last term accounts for the one clock uncertainty caused by sampling an asynchronous signal. The fourth column uses the same equation except that the ideal BDF is off by ±1. This case represents the potential error that could occur during the autobaud operation, where the measured baud rate deviates by ±1 clock cycle. As shown in Table 7.7-4 and Table 7.7-6, the error rate for high baud rates is unacceptable without some adjustments.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

Table 7.7-6 Sample Location Error Various Baud Rates for 13 MHz System Clock

Desired BR	Desired BDF	Actual BDF	Total Error Actual	Total Actual Error with BDF Value ±1
721000	18.0305	18	2.03	68
460800	28.2118	28	9	49.78
230400	56.4236	56	9	29.75
153600	84.6354	85	5.19	19.60
115200	112.847	113	1.64	12.39
76800	169.271	169	1.92	8.96
57600	225.694	226	1.64	6.98
38400	338.542	339	1.63	5.19
19200	677.083	677	0.15	1.92
14400	902.778	903	0.30	1.63
9600	1354.17	1354	0.15	1.04
7200	1805.56	1806	0.29	0.96
4800	2708.33	2708	0.156	0.59
3600	3611.11	3611	0.04	0.37
2400	5416.67	5416	0.15	0.37
1800	7222.22	7222	0.04	0.20
1200	10833.3	10833	0.03	0.14
900	14444.4	14444	0.03	0.12
600	21666.7	21667	0.02	0.07
300	43333.3	43333	0.01	0.04

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

7.7.2.3 Range Registers and Desired Baud Divisor Registers

There is a set of programmable registers that helps the hardware select optimal values of clock divisors for higher baud rates and detect when the measured values are too high or too low. Note that the baud divisor referred to in this discussion is equal to the BDF – 1. In other words, the actual amount the system clock is divided is equal to the baud divisor + 1. The autobaud block provides the baud rate divisor, which can be generated from the baud measurement or directly from a CPU write. The measured baud rate is obtained in two different ways. For low baud rates, the baud rate divisor will be large enough so that small errors in the measurement are tolerable. For high baud rates, differences between the desired baud divisor and the actual baud divisor cannot be tolerated, and, therefore, the baud divisor is obtained indirectly for higher baud rates.

Five range registers (A—E) and five desired divider registers are used to select the exact desired baud rate divisor for higher rates. Also, a baud overflow and underflow register is used to detect measurements that are out of a useful range. The A—E range registers must be arranged in numerical order with each register representing an unsigned integer number = BDF – 1. The value from register A, value A, should be greater than value B; value B should be greater than value C; value C should be greater than D; and so on. Value E should be greater than the underflow register. The autobaud measurement count increases with the width of the baud period. The measurement must first be greater than the underflow value to be valid. After that, and while it is less than or equal to E, it is in the lowest divisor range (highest baud rate), and if the count stops before it is greater than E, then the E to underflow range is selected, and the preprogrammed divisor for that range is selected. If the measured divisor exceeds the E value and is less than or equal to D, the divisor selected should be in the D—E range. The other ranges work in the same way. When the value of divisor exceeds the A value, then it has exceeded any of the preset values, and the measured value is used unless that value exceeds the maximum acceptable value. Regardless of what value is measured, software can write the divisor register with whatever it chooses.

Table 7.7-7 Baud Divisor Register Overflow Value (ACCBDO), Address (UART_BASE_ADDR + 0x028)

Bit	31—17	16—0
Name	RSVD	Baud_Ovrf

Bit	Name	Description
31—17	RSVD	Reserved.
16—0	Baud_Ovrf	Baud overflow. Bits[16:0] specify the overflow value of the baud rate divisor. When the measured baud rate divisor is greater than this value, the baud rate measurement is considered invalid and the autobaud operation will not complete until a smaller baud rate is measured. This value resets to all 1s (0x1FFFF).

Table 7.7-8 Baud Divisor Register Underflow Value (ACCBDU), Address (UART_BASE_ADDR + 0x02C)

Bit	31—10	9—0
Name	RSVD	Baud_Undrf

Bit	Name	Description
31—10	RSVD	Reserved.
9—0	Baud_Undrf	Baud underflow. Bits[9:0] specify the underflow value of the baud rate divisor. When the measured baud rate division is less than or equal to this value, the baud rate measurement is considered invalid and the autobaud operation will not complete until a valid baud measurement can be made. This value resets to 0x0. This register can be written and read by software.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

Table 7.7-9 Range Registers A—E (ACCBRA—ACCBRE), Address (UART_BASE_ADDR + 0x030—0x40)

Bit	31—10	9—0
Name	RSVD	B_RANGE[A—E]

Bit	Name	Description
31—10	RSVD	Reserved.
9—0	B_RANGE[A—E]	Baud divisor range A—E. Unsigned integer value used to compare against the measured baud divisor. This comparison is used to determine what range of values the measured baud divisor falls into, and from this determination, the desired baud divisor can be selected.

Table 7.7-10 Baud Divisor A (ACCBDA), Address (UART_BASE_ADDR + 0x044)

Bit	31—10	9—0
Name	RSVD	Dvsr_A2B

Bit	Name	Description
31—10	RSVD	Reserved.
9—0	Dvsr_A2B	Divisor for range A—B. The unsigned integer value of the desired baud divisor = BDF – 1 if the measured divisor is less than or equal to B_RANGEA and greater than B_RANGEB.

Table 7.7-11 Baud Divisor B (ACCBDB), Address (UART_BASE_ADDR + 0x048)

Bit	31—10	9—0
Name	RSVD	Dvsr_B2C

Bit	Name	Description
31—10	RSVD	Reserved.
9—0	Dvsr_B2C	Divisor for range B—C. The unsigned integer value of the desired baud divisor = BDF – 1 if the measured divisor is less than or equal to B_RANGEB and greater than B_RANGEC.

Table 7.7-12 Baud Divisor C (ACCBDC), Address (UART_BASE_ADDR + 0x04C)

Bit	31—10	9—0
Name	RSVD	Dvsr_C2B

Bit	Name	Description
31—10	RSVD	Reserved.
9—0	Dvsr_C2D	Divisor for range C—D. The unsigned integer value of the desired baud divisor = BDF – 1 if the measured divisor is less than or equal to B_RANGEC and greater than B_RANGED.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

Table 7.7-13 Baud Divisor D (ACCBDD), Address (UART_BASE_ADDR + 0x050)

Bit	31—10	9—0
Name	RSVD	Dvsr_D2E

Bit	Name	Description
31—10	RSVD	Reserved.
9—0	Dvsr_D2E	Divisor for range D—E. The unsigned integer value of the desired baud divisor = BDF – 1 if the measured divisor is less than or equal to B_RANGED and greater than B_RANGEE.

Table 7.7-14 Baud Divisor E (ACCBDE), Address (UART_BASE_ADDR + 0x054)

Bit	31—10	9—0
Name	RSVD	Dvsr_E2U

Bit	Name	Description
31—10	RSVD	Reserved.
9—0	Dvsr_E2U	Divisor for range E to underflow. The unsigned integer value of the desired baud divisor = BDF – 1 if the measured divisor is less than or equal to B_RANGEE and greater than Baud_Undrf.

7.7.2.4 Baud Measurement Register (ACCBM)

It is very important for software to pick the appropriate values to write into the range registers and the divisor registers associated with the ranges. The hardware does not include the overhead of special hardware that would check to make sure the range and divisor registers are programmed with sensible values. The values of the range are programmed to allow flexibility in ability to work with different UART clock ranges and baud ranges.

It may happen that the baud divisors programmed for the different baud ranges may not be appropriate for the actual data rate of the received sequence. The received sequence baud rate and the preprogrammed rates must be compatible for the autobaud operation to work. However, some hardware is available that may help in diagnosing a condition where the baud rate programming is incorrect. The actual measured baud divisor value is saved in the baud measurement register that can be read by software.

Table 7.7-15 Baud Measurement Register (ACCBM), Address (UART_BASE_ADDR + 0x078)

Bit	31—17	16—0
Name	RSVD	Baud_M

Bit	Name	Description
31—17	RSVD	Reserved.
16—0	Baud_M	Baud measurement. Bits[16:0] represent the actual baud division measurement. This information may be useful in diagnosing situations where the selected baud divisor is one of the preprogrammed values, but is does not sufficiently match the actual baud rate of the received data.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

7.7.2.5 Tx/Rx Baud Rate Counters (ACCTXBC and ACCRXBC)

The baud rate counter is a 17-bit down counter that decrements by 1 every clock cycle (the system clock supplied to the UART). This counter is initialized with the value in the baud divisor register (or the baud divisor divided by 2) after the counter counts down to 0, or if the baud divisor register is written. These registers are read-only and return the counter value when read. The divide by two operation is used to locate the middle of the baud interval when the logic starts to sample the incoming character.

Table 7.7-16 Rx Baud Counter (ACCRXBC), Address (UART_BASE_ADDR + 0x05C)

Bit	31—17	16—0
Name	RSVD	RxBaud_Cntr

Bit	Name	Description
31—17	RSVD	Reserved.
16—0	RxBaud_Cntr	Receive baud counter. Bits[16:0] are the current value of the Rx baud rate counter.

Table 7.7-17 Tx Baud Counter (ACCTXBC), Address (UART_BASE_ADDR + 0x060)

Bit	31—17	16—0
Name	RSVD	TxBaud_Cntr

Bit	Name	Description
31—17	RSVD	Reserved.
16—0	TxBaud_Cntr	Transmitter baud counter. Bits[16:0] are the current value of the Tx baud rate counter.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

7.7.2.6 FIFO Status Register (ACCFIFOS)

The FIFO status register is used to inform the CPU of the status of the transmitter and receiver FIFOs. The FIFO status register is a read-only register. Writes to its address will be ignored. Table 7.7-18 shows the format of the FIFO status register.

Table 7.7-18 FIFO Status Register (ACCFIFOS), Address (UART_BASE_ADDR + 0x008)

Bit	31—8	7	6	5	4	3	2	1	0
Name	RSVD	RID	TSE	TFF	TFTH	TFE	RFF	RFTH	RFE

Bit	Name	Description
31—8	RSVD	Reserved.
7	RID	Receiver idle. If 1, the receiver is idle. If 0, the receiver is not idle.
6	TSE	TSR empty. If 1, the transmitter shift register is empty. If 0, the transmitter shift register is not empty.
5	TFF	Transmitter FIFO full. If 1, the transmitter FIFO is full. If 0, the transmitter FIFO is not full.
4	TFTH	Transmitter FIFO threshold met. If 1, the transmitter byte count is less than or equal to the Tx_Thld programmed in the Tx control register. If 0, the transmitter byte count is greater than Tx_Thld.
3	TFE	Transmitter FIFO empty. If 1, the transmitter FIFO is empty. If 0, the transmitter FIFO is not empty.
2	RFF	Receiver FIFO full. If 1, the receiver FIFO is full. If 0, the receiver FIFO is not full.
1	RFTH	Receiver FIFO threshold met. If 1, the receiver byte count is greater than or equal to the Rx_Thld programmed in the Rx control register. If 0, the receiver byte count is less than Rx_Thld.
0	RFE	Receiver FIFO empty. If 1, the receiver FIFO is empty. If 0, the receiver FIFO is not empty.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

7.7.2.7 UART Status Register (ACCS)

The UART status register is used to inform the CPU of the status of the UART. The UART status register is a read-only register. Writes to its address will be ignored.

Table 7.7-19 UART Status Register (ACCS), Address (UART_BASE_ADDR + 0x00C)

Bit	31—16	15	14	13—10	9
Name	RSVD	BdEr	BrkD	MISSC	CCE0

Bit	8—6	5	4	3	2	1	0
Name	CMF	ACD	TFT	RFEr	ROE	RPE	RFT

Bit	Name	Description
31—16	RSVD	Reserved.
15	BdEr	Baud detection error. This bit is set to a logic 1 when the automatic baud detection measures a baud period that is not within a valid range. This bit will become only one time for each baud measurement. This bit is reset to a logic 0 when this status register is read.
14	BrkD	Break detected. The receiver logic sets this bit to a logic 1 when a break character is received. If the receiver port is continuously held to a logic 0 for a period longer than the break character, this bit will only be set once for that period. This bit is reset to a logic 0 when the UART status register is read. The length of the period that is detected as a break character is determined by the number of data bits + a start bit + a parity bit (if used) + the number of stop bits. For example, if the format uses 8 bits of data, 1 parity bit, and 1 stop bit, the break character must be at least 11 baud periods long to guarantee that the receiver logic will detect the break.
13—10	MISSC	Modem input state change 3—0. These bits detect state changes on the four general-purpose modem function input pins. The general-purpose modem input pins are sampled at the UART clock rate and the corresponding bits in this register are set to a logic 1 whenever a change in state is detected. These bits are cleared when read by the CPU. The nominal minimum pulse-width of signals on the modem input state pins must be sufficiently greater than 1 UART clock period in order to guarantee detection of state changes.
9	CCE0	Character count equal 0. Character counter set to 0. This bit gets set whenever the character counter counts down to 0. This bit is reset when the UART status register is read.
8—6	CMF	Character match flags. These bits get set when the receiver character matches a valid pattern in the corresponding receive match register. The character match occurs when a byte is received; it matches a valid character in Table 7.7-26, and the character match enable bit of the receive control register is set. Bit 8 corresponds to match character register 2, bit 7 corresponds to match character register 1, and bit 6 corresponds to match character register 0. These bits are reset logic 0s when the UART status register is read.
5	ACD	Autoconfiguration done. The autoconfiguration status register is set to 1 when the autoconfiguration completes. This bit is set only once for each completion and if this bit is cleared, it will not set again unless the autoconfiguration goes to the completion state again. This bit is reset to a logic 0 when the autoconfiguration register is read.
4	TFT	Transmitter FIFO threshold. Bit 4 is the transmitter FIFO threshold event indicator. Bit 4 is set to 1 if the transmitter FIFO threshold condition is met. Bit 4 changes to 0 when the transmitter FIFO condition is not met, due to a FIFO write or control change.
3	RFEr	Receiver frame error. If bit 3 is 1, a framing error has occurred (i.e., a received character did not have a valid stop bit). This bit is reset when the UART status register is read.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

Table 7.7-19 UART Status Register (ACCS), Address (UART_BASE_ADDR + 0x00C) (continued)

Bit	Name	Description
2	ROE	Receiver overrun error. Bit 2 is the overrun indicator. If bit 2 is 1, a character was received at a time when the receive FIFO was full. This bit is reset when the UART status register is read.
1	RPE	Receiver parity error. If bit 1 is 1, a parity error has occurred in a received character. This bit is reset when the UART status register is read.
0	RFT	Receiver FIFO threshold. Bit 0 is the receiver FIFO threshold event indicator. Bit 0 is set to 1 if the receiver FIFO threshold condition is met. Bit 0 changes to 0 when the FIFO condition is not met, due to a FIFO read or control change.

Table 7.7-20 lists the status register events, type (event or condition), and the method of clearing the associated interrupt source. Table 7.7-20 refers to clearing the source of the UART interrupt. To completely understand the device's interrupt handling, the interrupt controller description must be referenced. Also, the UART has an interrupt reset input from the interrupt controller, but this pin will not clear the UART interrupt except for a single cycle. After the interrupt reset is asserted and then deasserted, the UART will continue to drive the interrupt signal to interrupt controller (assuming the interrupt remains enabled) until the interrupt source within the UART is cleared.

Table 7.7-20 Interrupt Handling

Bit	Name	Interrupt Type	UART Interrupt Source Clearing Method
15	BdEr	Event	Status register read.
14	BrkD	Event	Status register read.
13—10	MISSC	Event	Status register read.
9	CCE0	Event	Status register read.
8—6	CMF	Event	Status register read.
5	ACD	Event	Autoconfiguration register read.
4	TFT	Condition/ Event	This interrupt is an event or condition interrupt source, depending upon the TxFIFO interrupt enable bits of the transmitter control register (Table 7.7-27). The condition interrupt source is cleared by changing the TxFIFO condition. The event interrupt source is cleared upon status register read.
3	RFE	Event	Status register read.
2	ROE	Event	Status register read.
1	RPE	Event	Status register read.
0	RFT	Condition/ Event	This interrupt is an event or condition interrupt source, depending upon the RxFIFO interrupt enable bits of the receiver control register (Table 7.7-21). The condition interrupt source is cleared by changing the RxFIFO condition. The event interrupt source is cleared upon status register read.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

7.7.2.8 Receiver Control Register (ACCRXC)

The receiver control register is used to control the receiver FIFO, interrupts, and parity generation. On any reset, the receiver control register will be set to all 0s.

Table 7.7-21 Receiver Control Register (ACCRXC), Address (UART_BASE_ADDR + 0x010)

Bit	31—19	18	17	16	15—11
Name	RSVD	Brk_IE	BrkD_En	FLBk	Rx_Thld

Bit	10	9—8	7	6	5	4—3	2—1	0
Name	RxStop2	RxChSize	RID	RxNI_IEn	RxE_IEn	RxP	RxF_IEn	RxFRST

Bit	Name	Description
31—19	RSVD	Reserved.
18	Brk_IE	Break detection interrupt enable. This bit, when set to a logic 1, causes the UART to generate an interrupt if a break character is detected. Note that a break character cannot be detected if the BrD_En bit of this register is not a logic 1. When this bit is a logic 0, break characters cannot generate interrupts.
17	BrkD_En	Break detection enable. This bit, when set to a logic 1, enables the Rx state machine to detect the break character. A break character is defined as a character that is all 0s including the start bit, data, parity (if any), and all of the stop bits (1 or 2). For example, when the receiver is configured for 7-bit data with parity and 2 stop bits, the break character would be 11 baud periods long. When this bit is set to a logic 0, break characters are not detected, and thus, the UART status register or interrupts would not show the detection of any break character. Also, when a break character is received and the break detection is enabled, a special character is written into the RxFIFO. Only one break character will be written into the RxFIFO for any given continuous break period, regardless of how much longer the duration of the received break character is than the minimum period required for a break character.
16	FLBk	Force loopback. This bit forces the Rx data to be looped back onto the Tx port when this value is set a logic 1. When this bit is set, it will not take effect while a character is being received, but it will cause any current transmissions to complete. However, because the receive signal is asynchronous, the first start bit may be distorted by 1 UART clock cycle when this signal is first asserted. When this signal is deasserted, it may be possible to stop the loopback in the middle of a character.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

Table 7.7-21 Receiver Control Register (ACCRXC), Address (UART_BASE_ADDR + 0x010) (continued)

Bit	Name	Description
15—11	Rx_Thld	Rx FIFO threshold. Unsigned integer value that the number of bytes in the RxFIFO must exceed or equal to set the Rx FIFO threshold in the status register. For example if bits[15:11] are programmed to 10000 (bit 15 = 1), then if the number of characters in the RxFIFO is 16 or more, the Rx threshold flag will be set.
10	RxStop2	Receive select two stop bits. When this bit is a logic 1, the receiver expects 2 stop bits.
9—8	RxChSize	Receive character size. Selects the character size, 8 bits or 7 bits. 00—7-bit characters. 01—8-bit characters. 11—Reserved. 10—Reserved.
7	RID	Receiver interface disable. Bit 7 is used to disable the receiver. Writing a 1 to this bit will disable the receiver.
6	RxNI_IEn	Receiver not idle interrupt. Bit 6 is used to enable the Rx not idle interrupt.
5	RxE_IEn	Receiver error interrupt enable. Bit 5 is used to enable receiver error interrupts (parity, frame, and overrun). A value of 0 disables them, a value of 1 enables them. This bit is set to a 0 upon any reset.
4—3	RxP	Receiver parity. Bits[4:3] are used to control receiver parity checking. Table 7.7-23 shows the encoding for this field. Parity checking is disabled upon any reset.
2—1	RxF_IEn	Receiver FIFO interrupt enable. Bits[2:1] are used to control the receiver FIFO interrupt. Table 7.7-22 shows the encoding for this field. Receiver FIFO interrupts are disabled upon any reset.
0	RFRst	Receive FIFO reset. Bit 0 is used to reset the receiver FIFO. Writing a 1 to this bit will reset the receiver FIFO, discarding any data still there and marking it empty. Bit 0 must be written to 0 before the FIFO can accept new data. The receiver FIFO is reset upon any reset to the UART.

Table 7.7-22 Encoding for Bits[2:1]

Bits[2:1]	FIFO Threshold Interrupt Control
0 0	FIFO threshold interrupts disabled.
0 1	Generates an interrupt when the receiver FIFO is not empty.
1 0	Generates an interrupt when the receiver FIFO programmable threshold sets. This interrupt sets only once when the byte count crosses the threshold.
1 1	Generates an interrupt when the receiver FIFO is full.

Table 7.7-23 Encoding for Bits[4:3]

Bits[4:3]	Parity
0 0	No parity.
0 1	Mark parity (always send a 1).
1 0	Even parity.
1 1	Odd parity.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

7.7.2.9 Character Interval Count Registers (ACCCIC and ACCCICCR)

There are two registers that provide the ability to count Rx idle intervals. The purpose of these registers is to count the number of characters while the Rx input is idle and the RxFIFO is not empty, and to generate an interrupt if the Rx idle period exceeds a programmable threshold. The two registers include a control register and the actual character interval counter. The control register enables the counter register and provides the maximum character count that is loaded into the interval counter. Whenever the interval counter counts down from the maximum value to 0, an interrupt is generated if the interrupts for this function are enabled. This character counter can be programmed to free run or count down to 0 once and then stop. Every time a character is received or the RxFIFO becomes empty, this counter is reset to the maximum value if enabled and the countdown starts again while the RxFIFO is not empty and the receiver state machine is idle. The character frame length is derived from the character format settings in the receiver control register.

Table 7.7-24 Character Interval Counter Control Register (ACCCICCR), Address (UART_BASE_ADDR + 0x064)

Bit	31—12	11	10—9	8—0
Name	RSVD	CIC_IEn	CICntEn	MAXCICnt

Bit	Name	Description
31—12	RSVD	Reserved.
11	CIC_IEn	Character count interrupt enable. If this bit is a logic 1, then an interrupt will be generated every time the character count reaches 0.
10—9	CICntEn	Character interval count enable bits. The operation for these bits is as follows: 00—Character counter is disabled. 01—Character counter will count down to 0 and stop. 11—Character counter will free run (wrap at zero). 10—Reserved.
8—0	MAXCICnt	Maximum character count interval. This is the value loaded into the character count interval counter register when it is started, or when it is in continuous mode and wraps around from 0.

Table 7.7-25 Character Interval Counter Register (ACCCIC), Address (UART_BASE_ADDR + 0x068)

Bit	31—9	8—0
Name	RSVD	CICnt

Bit	Name	Description
31—9	RSVD	Reserved.
8—0	CICnt	Character interval count. This is the counter value. The counter counts down from the maximum character count interval at the frame rate until a character is received. The frame rate is equal to the baud rate divided by the frame length. The frame length includes the number of bits used for a start bit, data, parity, and stop bits. The maximum character count is reloaded into the counter and the countdown continues if a character is received, or the counter is in continuous mode and it reaches 0.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

7.7.2.10 Flow Control Registers

Flow control support is available via the set of three match registers and the general-purpose modem I/O function blocks. The normal high-speed clock must be provided for these registers to operate during system powerdown.

The match registers allow generating interrupts upon detection of the receipt of special characters. The modem interface provides support for six standard modem control signals, including the detection of changes of state on the modem control input signals. The modem input signals are sampled at the UART block clock rate, and, therefore, modem signals should remain at a given logic level for a period sufficiently longer than one clock cycle to guarantee that state changes can be detected.

7.7.2.11 Character Match Registers (ACCCMC0—ACCCMC3)

There are three match registers that allow detection of up to three matching characters.

Table 7.7-26 Character Match Control Register (ACCCMC0—ACCCMC3), Addresses (UART_BASE_ADDR + 0x06C, 0x070, 0x074)

Bit	31—10	9	8	7—0
Name	RSVD	Vld_Pat	CM_IEn	M_Char

Bit	Name	Description
31—10	RSVD	Reserved.
9	Vld_Pat	Valid pattern. This bit must be set to a logic 1 to indicate whether the pattern in bits[7:0] may be used for matching. If this bit is a logic 1, then the corresponding bit in the status register will be set when the UART receives a character that matches the pattern in M_Char (7 or 8 bits). Additionally, if this bit is set, and the CM_IEn is set, receipt of a matching character will cause an interrupt.
8	CM_IEn	Match interrupt enable. When this bit is a logic 1 and Vld_Pat of this register is a logic 1, an interrupt will be generated when the matching character is received. If either Vld_Pat or CM_IEn are a logic 0, the logic will not generate an interrupt.
7—0	M_Char	Match character. The receive character is compared to the pattern in these bits and, depending upon the state of Vld_Pat and CM_IEn, the corresponding status bits and interrupt may be set when the receive character matches this pattern. Bit 8 of this pattern is ignored if the receive character size is programmed for 7 bits.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

7.7.2.12 Transmitter Control Register (ACCTXC)

The transmitter control register is used to control the transmitter FIFO, interrupts, and parity generation. On any reset, the transmitter control register will be set to all 0s.

Table 7.7-27 Transmitter Control Register (ACCTXC), Address (UART_BASE_ADDR + 0x014)

Bit	31—17	16	15—11	10	9—8	7	6	5	4—3	2—1	0
Name	RSVD	SetBrk	Tx_Thld	TxStop2	TxChSize	TxD	TxSI	TOD	PC	FICE	TxFR

Bit	Name	Description
31—17	RSVD	Reserved.
16	SetBrk	Set break. This bit will cause the transmitter logic to generate a break character when it is set to a logic 1. This bit does not self-clear and must be written to a logic 0 to clear. The transmitter logic will finish the current transmission when it is set and then generate the break command. The software does not have to wait for the break to become asserted on the transmitter before clearing this bit; once this bit is asserted, the transmitter will always generate a break character, assuming that none of the loopback modes are enabled. The software can extend the break period because the transmitter logic will generate another break character immediately after the previous one finishes if this bit is still active at the end of the break character. Each break character is 12 baud periods long (or a multiple of 12 baud periods if the break setbrk command is not cleared by the CPU before the end of the break character).
15—11	Tx_Thld	TxFIFO threshold. Unsigned integer value that the number of bytes in the TxFIFO must be less than or equal to in order to set the Tx FIFO threshold in the status register. For example, if bits[15:11] are programmed to 10000 (bit 15 = 0), then if the number of characters in the TxFIFO is 16 or less, the Tx threshold flag will be set.
10	TxStop2	Select two stop bits. When this bit is a logic 1, the transmitter will send 2 stop bits; otherwise, 1 stop bit will be used.
9—8	TxChSize	Transistor character size. Selects the character size, 8 bits or 7 bits. 00—7-bit characters. 01—8-bit characters. 11—Reserved. 10—Reserved.
7	TxD	Transmitter disable. When this bit is set to a logic 1, it will disable the transmitter logic, and when this bit is set to a logic 0, the transmitter can send data if both the autoconfiguration and the force loopback modes are not active and if the autoconfiguration mode is not active. If already set before autoconfiguration mode, the Tx disable bit will not be automatically cleared when the autoconfiguration operation finishes, but must be cleared by software. Also, when this bit is set to a logic 1, the transmit state machine will finish any character transmission in progress, but will not unload new data from the Tx FIFO into the transmit shift register once any current transmission finishes.
6	TxSI	Transmit shift register empty interrupt. Bit 6 is used to enable the transmitter shift register empty interrupt.
5	TOD	Transmit open drain. When set to a logic 1, this bit will cause the transmitter output buffer to operate as an open-drain device. When this is a logic 0, the transmitter output buffer can drive a logic 0 or a logic 1.
4—3	TxP	Transmit parity. Bits[4:3] are used to control transmitter parity generation. Table 7.7-23 shows the encoding for this field. Parity generation is disabled upon any reset.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

Table 7.7-27 Transmitter Control Register (ACCTXC), Address (UART_BASE_ADDR + 0x014) (continued)

Bit	Name	Description
2—1	FICE	Transmit FIFO interrupt control enable. Bits[2:1] are used to control the transmitter FIFO interrupt. Table 7.7-28 shows the encoding for this field. Transmitter FIFO interrupts are disabled upon any reset.
0	TxFR	Transmit FIFO reset. Bit 0 is used to reset the transmitter FIFO. Writing a 1 to this bit will reset the transmitter FIFO, discarding any data still there and marking it empty. Bit 0 must be written to 0 before the FIFO can accept new data. The transmitter FIFO is reset upon any reset to the UART.

Table 7.7-28 FIFO Threshold Interrupt Control

Bits[2:1]	FIFO Threshold Interrupt Control
0 0	FIFO threshold interrupts disabled.
0 1	Generate an interrupt when the transmitter FIFO is not full.
1 0	Generate an interrupt when the transmitter FIFO count is less than or equal to the TxFIFO programmable threshold set in the Tx control register. This interrupt is set only once for each time the count reaches the threshold.
1 1	Generate an interrupt when the transmitter FIFO is empty.

7.7.2.13 Tx/Rx FIFO Register (ACCFIFO)

The Tx/Rx FIFO register provides access to the transmitter and receiver FIFOs. A write to this register writes a character to the transmitter FIFO. A read from this register reads a character from the receiver FIFO. Both FIFOs are reset upon any reset to the UART.

Both FIFOs provide status information for the FIFO status register and the UART status register. This information is also used to generate the transmitter and receiver FIFO threshold interrupts.

The FIFOs do not store the characters currently being transmitted from the transmitter shift register or received in the receiver shift register.

A read from an empty Rx FIFO will return the byte from the FIFO position just after the last Rx FIFO read, but it will not change the status of the Rx FIFO. A write to a full Tx FIFO will be ignored. Table 7.7-29 and Table 7.7-30 show the receiver and transmitter FIFO data format.

Table 7.7-29 Tx/Rx FIFO Register (ACCFIFO), Address (UART_BASE_ADDR + 0x01C)

Bit	31—11	9	8	7—0
Name	RSVD	RxErr	SpecChar	Rx_FIFO_OUT/Tx_FIFO_IN

Bit	Name	Description
31—11	RSVD	Reserved.
9	RxErr	Rx error (parity or frame error).
8	SpecChar	Special character flag.
7—0	Rx_FIFO_OUT/ Tx_FIFO_IN	Receive FIFO data out/transmit FIFO data in. Character to transmit when written to. Character received when read from. Bit 7 is always read as a logic 0 when receiver is in 7-bit mode.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

Table 7.7-30 FIFO Data Format

Bit Pattern	Definition	
	Rx	Tx
0x000—0x0FF	Normal 7-bit or 8-bit data (no errors). Bit 7 is always 0 for 7-bit mode.	7-bit or 8-bit data. Bit 7 should always be written as 0 for 7-bit mode.
0x100	Break character received.	Reserved.
0x300—3FF	Reserved.	Reserved.
0x200—2FF	7-bit or 8-bit data with errors. Bit 7 will always be 0 for 7-bit mode.	Reserved.

7.7.2.14 General-Purpose Modem Interface Registers (ACCMIRA and ACCMIRB)

There are two general-purpose registers, along with UART status bits, that are available for modem interface logic. These registers support up to six modem ports: two as inputs, two as outputs, and two bidirectional ports. Not all of the modem pins are available in each instantiated block. In some cases, the pins are not available at all, and in some other cases, the modem pins are shared with other functions. The PMUX module (see Section 7.15) provides the altpin control registers (see Table 7.15-4) with the ability to select the function selected for shared pins.

The feature control register also helps control the modem pins. The feature control register sets the direction of the bidirectional ports, enables the output-only ports, and enables the input ports. The alternate pin function registers also must be programmed for modem pins that are shared with other functions. The feature control register cannot control the modem pin output data or direction if the modem function is not selected by the alternate function register.

These general-purpose registers allow support of standard 6-port modem interfaces (as DTE or DCE). The output control register allows direct control of active-low output pins and enables interrupts that are triggered by state changes on the four input pins. The minimum pulse-width that the logic can detect is determined by the system clock supplied to the UART. The input register provides direct access to the inverted state of up to four input pins, and the UART status register records changes in the state of the four input pins. The modem input register bits always read back as active (logic 1) if the associated pin is programmed as a modem output or is used as a PPI port. Table 7.7-31 and Table 7.7-32 provide the description of general-purpose modem registers A and B.

Table 7.7-31 General-Purpose Modem Register A (ACCMIRA), Address (UART_BASE_ADDR + 0x00)

Bit	31—8	7—4	3—0
Name	RSVD	MISIEn	M_OS

Bit	Name	Description
31—8	RSVD	Reserved.
7—4	MISIEn	Modem interface state interrupt enable 3:0. These register bits, when set to a logic 1, enable interrupts whenever the state changes on the corresponding modem interface input pins.
3—0	M_OS	Modem interface output states 3:0. A 1 written to these register bits will cause the corresponding active-low output pins to drive a logic 0 on the output, and a 0 written to these register bits will cause logic 1s to be asserted on the output pins. When the CPU writes a 1 to this register, the state of this register bit will be read back as a logic 1. A logic 0 written to these bits will read back as a logic 0.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

Table 7.7-32 General-Purpose Modem Register B (ACCMIRB), Address (UART_BASE_ADDR + 0x04)

Bit	31—4	3—0
Name	RSVD	M_IS

Bit	Name	Description
31—4	RSVD	Reserved.
3—0	M_IS	Modem interface input state 3:0. These register bits refer to the inverted state of the corresponding four general-purpose input pins. These bits will always be read back as 0s if the corresponding M_IE bit of the feature control register is not set to a logic 1.

7.7.2.15 Feature Control Register (ACCFC)

This section describes the feature control register, which is used for controlling the general-purpose modem interface pin multiplexing operation and enabling the IrDA mode. The IrDA mode also uses the mode control register. The modem registers are shown in Table 7.7-31 and Table 7.7-32.

The UART for some applications can be used to drive an IrDA formatter. The IrDA formatter converts the UART serial data port into a signal format that is compatible with an IrDA transceiver. The feature control register (Table 7.7-33) is used to select the IrDA mode for those applications that support this feature. The feature control register also provides parameters for controlling the IrDA logic. The IrDA mode logic uses a slightly different method to set the baud rate and thus requires the mode control register (Table 7.7-35). In the non-IrDA mode, the baud rate is determined by dividing the UART clock rate by the value in the baud divisor register. In the IrDA mode, the clock division factor that determines the baud rate is the baud divisor register (sample mode value) with an additional adjustment factor. The additional adjustment factor is used to increase the accuracy of the synthesized baud rate and is controlled by the AL/CO bit in the mode register. The sample mode value ranges between 16—31. On any reset, the mode control register is set to all 0s. More details on the IrDA mode operation are provided in the IrDA chapter.

Table 7.7-33 Feature Control Register (ACCFC), Address (UART_BASE_ADDR + 0x020)

Bit	31—18	17—14	13—10	9	8	7—0
Name	RSVD	M_IE	M_OE	MUX	IDE	PWC

Bit	Name	Description
31—18	RSVD	Reserved. Write with 0.
17—14	M_IE[3:0]	Modem interface input enable. Enables the inputs from the modem pins to set the M_IS bit of modem register B (see Table 7.7-32). The M_IE bit must be set to a logic 1 before the M_IS register can be set and input changes can be detected. If the M_IE are set to logic 0, the corresponding M_IS bits will be read back as logic 0s. If the modem pin is bidirectional and the user wants to program it as an output, M_IE should not be set. Table 7.7-34 provides further details on the association of M_IE to M_IS bits.
13—10	M_OE[3:0]	Alternative pin for modem general-purpose interface inputs. When these bits are set to a logic 1, the associated modem output register can drive the pin, providing that alternate pin function (see Table 7.15-4) has selected the modem interface for the pin. When the M_OE bits are a logic 0, the associated pin will be driven to a logic Z state, unless the pin has been programmed for an alternate pin function. Table 7.7-34 provides further details on how to select the device pins to be used for the modem interface.
9	MUX	Internal MUX. If 0, IrDA. If 1, UART.
8	IDE	IrDA enable. This pin must be set to 1 to enable IrDA function on pins PIO42_IRDARX and PIO41_IRDATX. When set to a 0, the IrDA function is not available.
7—0	PWC	Pulse-width count value. These bits are used for the IrDA mode to control the IrDA signal pulse-widths.

7 Call Processor (CP) Block (continued)

7.7 Asynchronous Serial Communications Controller (UART) (continued)

Table 7.7-34 provides a description of the mapping of the feature control register control bits to the associated modem interface register and device pin.

Table 7.7-34 Mapping for General-Purpose Modem Register to Feature Control Register I/O Control

MGPI Output Enable (ACCFC Register)	MGPI Input Enable (ACCFC Register)	General-Purpose Modem Register A	General-Purpose Modem Register B	External Pin*
M_OE[3]	M_IE[3]	M_OS[3]	M_IS[3]	CTS0, RTS1
M_OE[2]	M_IE[2]	M_OS[2]	M_IS[2]	DCD0, CTS1
M_OE[1]	—	M_OS[1]	—	RI0
M_OE[0]	—	M_OS[0]	—	DSR0
—	M_IE[1]	—	M_IS[1]	DTR0
—	M_IE[0]	—	M_IS[0]	RTS0

* ACC1 has only CTS1 and RTS1 available for modem pins.

7.7.2.16 IrDA Mode Control Register (IRDAMC)

Table 7.7-35 provides a description of the mode control register, which is used for IrDA applications.

Table 7.7-35 Mode Control Register (IRDAMC), Address (UART_BASE_ADDR + 0x018)

Bit	31—8	7—4	3	2:0
Name	RSVD	SM	AL/CO	RSVD

Bit	Name	Description
31—8	RSVD	Reserved.
7—4	SM	Sample mode. Selects the input sample clock that is equal to the decimal equivalent of bits[7:4] + 16.
3	AL/CO	Alternate/constant. Controls the special alternate mode. If 1, the least significant bit of the sample count is toggled for each new bit of a transfer. If 0, the sample count remains constant for each bit.
2—0	RSVD	Reserved.

7 Call Processor (CP) Block (continued)

7.8 IrDA

The IrDA formatting feature is supported on asynchronous serial communications controller (ACC) channel 0. It works with the ACC to provide compatibility with the IrDA infrared serial data link standard. A list of features for the IrDA formatter follows:

- Operates at speeds up to 115.2 Kbytes/s.
- Programmable pulse-width to the IrDA transceiver.

7.8.1 Operation

The IrDA formatting feature is enabled before it is used by configuring the feature register and other registers of ACC0. Table 7.7-35 shows valid combinations of the mode control and feature register required for IrDA operation.

Figure 7.8-1 shows how the output of the IrDA formatting feature follows the output of the ACC channel. When the ACC output is 0, the IrDA outputs a pulse high. When the ACC output is 1, there is no pulse during that bit time. The width of the pulse is determined by the value programmed into the PWC bits of the feature register of ACC channel 0, and by the clock period of the clock to the peripheral by the following formula:

IrDA Pulse-Width = 16 x [Clock Period x (PWC + 1)]

The feature register is set to ensure that the pulse-width meets the minimum required by the transceiver being used.

Figure 7.8-2 shows how the IrDA formatting feature converts the IrDA pulse back into data compatible with ACC channel 0. When the IrDA formatter receives a pulse low, the data is converted to a low for the ACC receive line. When a pulse low is not seen, the data is held high. A pulse is seen for a minimum of two clock cycles of the clock to the peripheral for a pulse to be seen by the IrDA formatter.

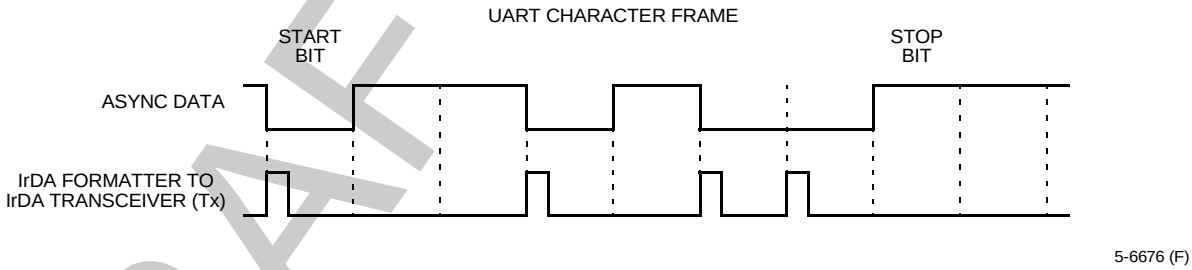


Figure 7.8-1 IrDA Formatter Transmit

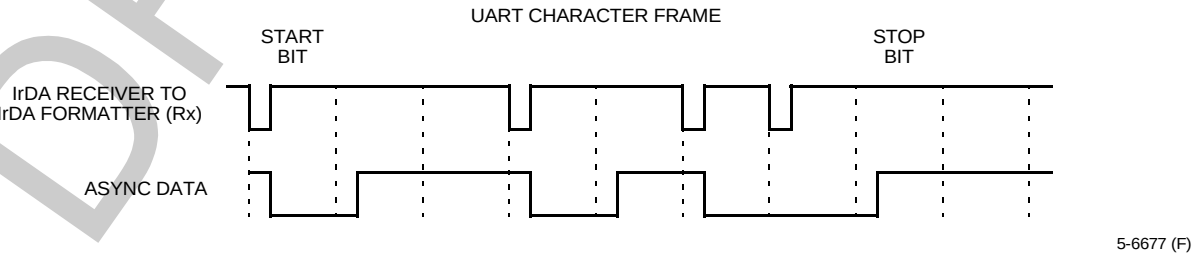


Figure 7.8-2 IrDA Formatter Receive

7 Call Processor (CP) Block (continued)

7.9 Timers

The programmable timers module supports three timer functions: interval timer, watchdog timer, and pulse-width modulator. The waveforms generated by the pulse-width modulator are present on output pins. The following is a list of features for the timer module:

- Pulse-width modulator with three output channels.
- Watchdog timer.
- Four interval timers.
- Generation of a shared interrupt request from the four interval timer channels, watchdog timer, and the three pulse-width modulators.
- Generation of a watchdog timer reset signal.

7.9.1 Operation

All of the counters in the programmable timer module operate synchronously with the peripheral clock, except the watchdog timer, which can be selected to run synchronously to the 32 kHz clock. The count rates are controlled by clock dividers that generate twelve count enable signals at intervals of 2^n of the system clock rate, where n = 1, . . . n = 12. The three timer functions independently select a count rate (see Table 7.9-3).

The interval timer function supports four independent timers running off of a common prescalar. Each timer consists of a 16-bit free-running counter, which decrements at the selected count rate, and a maximum count register, which determines the interval. When the timer is enabled, the counter begins counting down. When it reaches zero, the counter is reloaded with the value from the maximum count register and the status bit is set in the status register. If the counter starts with a value of 0, it loads the maximum count value, but it will set the status bit when it counts back to 0. The status bit will cause an interrupt if the corresponding bit in the enable register (Table 7.9-9) is set.

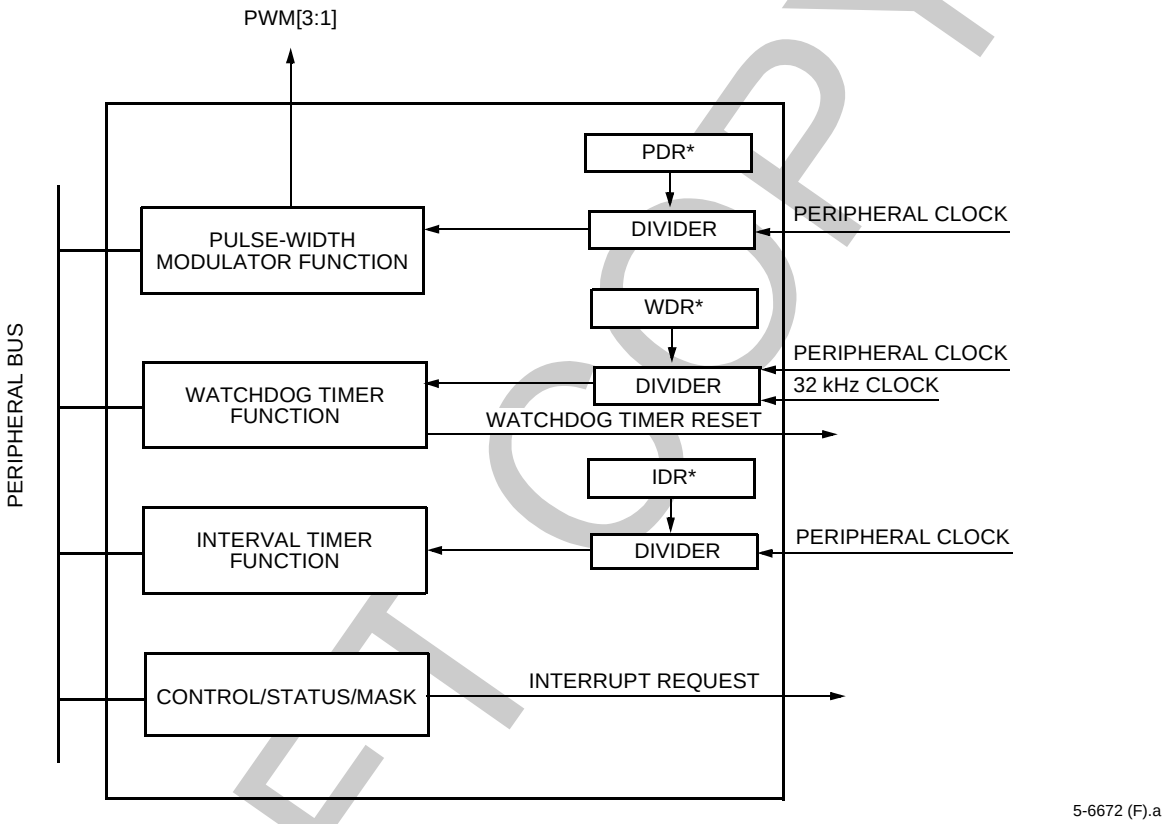
The watchdog timer function resets the device if the system software fails to restart the count sequence within a specified time interval. The watchdog timer block contains a 16-bit binary counter that increments at the selected count rate. The counter is reset to the all-0s value by writing a magic cookie value of 0xFADE to the watchdog timer count register address (Table 7.9-5). If the counter increments to the all-ones value, the watchdog timer time-out signal is asserted. The time-out signal can be configured to generate a watchdog reset or to generate an interrupt. The watchdog timer can be configured to run off of a 32 kHz clock or the peripheral clock.

The pulse-width modulator function generates output pulses with programmable period and duty cycles. These signals are used for tone generation or for low precision digital-to-analog conversion. An interrupt is asserted at the beginning of each pulse period to synchronize the loading of new register values.

7 Call Processor (CP) Block (continued)

7.9 Timers (continued)

Figure 7.9-1 shows a block diagram of the programmable timers architecture.



* PDR, WDR, and IDR are three 4-bit fields of the count rate register.

Figure 7.9-1 Block Diagram of the Programmable Timers

7.9.2 Pulse-Width Modulator

The pulse-width modulator (PWM) function is illustrated in Figure 7.9-1. The block contains three independent modulators. The count rate is selected by programming the pulse-width modulator count rate field of the count rate register with an index between 0 and 11. All three pulse-width modulators operate at the same count rate. On T8307, only two PWM channels are bonded out of the chip.

The pulse-width modulator channel operates by alternately loading the PWM maximum count register A into the PWM count register, counting down to 0x0001, then loading the PWM maximum count register B into the PWM count register, and counting down to 0x0001. At the beginning of the A count the PWM signal becomes high, and at the beginning of the B count the PWM signal becomes low.

At the beginning of the A count, the PWM status bit is set to 1 to indicate that the pulse-width modulator cycle has started. If the PWM interrupt enable bit is set to 1, the IRQ signal is asserted. Also, during the A count, the value from the PWM maximum count B register is held in a latch so that both the PWM maximum count A and PWM maximum count B registers are updated with new values to be used in the following cycle. The PWM status signal is reset by writing a 1 into that bit position in the status register.

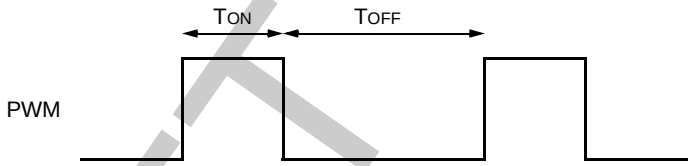
7 Call Processor (CP) Block (continued)

7.9 Timers (continued)

When the value in the PWM maximum count A register is 0, the A count sequence is suppressed and the output becomes low. The B count sequence continues and the PWM status bit is set whenever the PWM maximum count B register value is loaded into the PWM count register. Conversely, when the value in the PWM maximum count B register is zero, the B count sequence is suppressed and the output becomes high. The A count sequence continues and the PWM status bit is set whenever the PWM maximum count A register value is loaded into the PWM count register. If both of the PWM maximum count registers contain 0, the output is low and the count period is 2^{16} count cycles.

When the pulse-width modulator channel is switched from the disabled to the enabled state, the pulse-width modulator cycle restarts by loading the PWM maximum count A value and asserting the output to 1. An exception to this occurs when the PWM maximum count A register is 0, in which case, the cycle restarts by loading the PWM maximum count B value and asserting the output to 0.

The duty cycle of the PWM signal is determined by the relative magnitudes of the A and B values. The period of the PWM signal is determined by the selected PWM count rate, and the sum of the PWM maximum count A and PWM maximum count B values. If the PWM output signal is to be filtered externally to produce a dc level, then the PWM maximum count A and PWM maximum count B values should be right-normalized to minimize the period of the PWM signal. Figure 7.9-2 illustrates the relationship between the PWM maximum count register values, the PWM count rate value in the count rate register, and the PWM signal timings.



5-6673 (F)

TON (s) = PWM maximum count register A/P_CLK.
TOFF (s) = PWM maximum count register B/P_CLK.
P_CLK (Hz) = System clk/($2^{\text{PWM count rate [3:0]} + 1}$).

Figure 7.9-2 Variable Duty-Cycle Waveform Generator Output

7.9.3 Interval Timer

The interval timer function is illustrated in Figure 7.9-3. Only one of the four channels is shown. The IT count registers are free-running counters that maintain the time-base of the interval measurements. The count rate is selected by programming the interval timer count rate field of the count rate register with an index between 0 and 11.

7 Call Processor (CP) Block (continued)

7.9 Timers (continued)

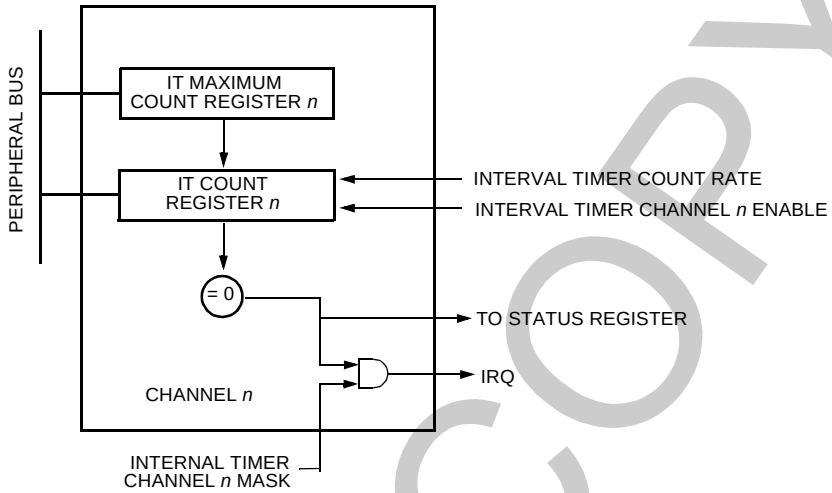


Figure 7.9-3 Block Diagram of the Interval Timer Function

The IT count register counts down until it reaches 0. When that happens, the appropriate bit is set in the status register and the count register is reloaded with the value in the maximum count register. The status bit may be reset by writing a 1 to the bit. If the appropriate bit in the interrupt enable register (Table 7.9-9) is set to 1, the status bit will cause the shared IRQ signal to be asserted.

The maximum count register may be read at any time. Writing the maximum count register will cause the count register to reset to 0.

The period of the interval timers is determined by the count rate value and the value of N in the maximum count register. The status bit will be set every N + 1 counts of the count register.

7 Call Processor (CP) Block (continued)

7.9 Timers (continued)

7.9.4 Watchdog Timer

The watchdog timer function is illustrated in Figure 7.9-4. The WT count register is a counter that maintains the time interval since the register was last reset. The count rate is selected by programming the watchdog timer count rate field of the count rate register with an index between 0 and 11.

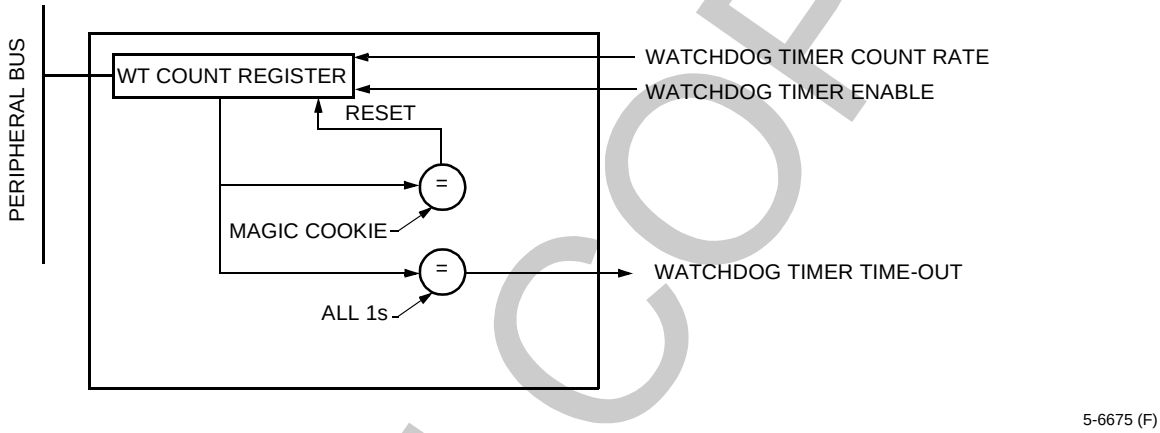


Figure 7.9-4 Block Diagram of the Watchdog Timer Function

The WT count register is read at any time. It is not written directly from the peripheral bus when the watchdog timer enable bit is set to 1 in the control register. In that case, a write access to the WT count register address with a data value equal to the (magic cookie) value causes the WT count register to be set to the all-0s value. The (magic cookie) value is 0xFADE. Writing the magic cookie value to the WT count register will also clear the WT status bit.

If the watchdog timer enable bit is set to 1, and the WT count register increments to the all-ones value, the watchdog timer time-out signal is asserted. The effect of the watchdog time-out is determined by the value of the watchdog timer interrupt (WTI) bit in the control register. If WTI is 1, a watchdog time-out will cause an interrupt. If another time-out occurs before the interrupt is cleared in the status register, a watchdog reset will occur. If WTI is 0, a time-out will always cause a watchdog reset. Once the watchdog timer function is enabled in the control register, it is not disabled and the watchdog timer count rate field of the count rate register and timer control bits (bits 8, 5—3) cannot be modified.

A status bit in the reset status register of the reset and power management function is set after the microcontroller restarts if a watchdog timer reset occurred.

Bit 4 of the control register is used to determine the effect of reset on the watchdog timer registers. If this bit is 1, the watchdog timer resets on watchdog reset but is not affected by the external RESET pin. If the bit is 0, the watchdog timer resets for all three reasons.

Bit 5 of the control register selects the clock source for the watchdog timer clock. If 1, the clock source is the 32 kHz clock. If 0, the clock source is the peripheral clock.

Note: Except for setting the WTE bit, the watchdog timer functionality should be completely set up before switching to the 32 kHz clock. After the clock servicing is set, then WTE can be set.

7.9.5 Registers

The interval timer function consists of nine registers. The watchdog timer function consists of two registers. The pulse-width modulator function consists of ten registers. All timers depend on the control, status, interrupt enable, and count rate registers.

7 Call Processor (CP) Block (continued)

7.9 Timers (continued)

7.9.5.1 PWM Maximum Count Registers (PWMMAXCA1—PWMMAXCA3, PWMMAXCB1—PWMMAXCB3)

The PWM maximum count registers A1—A3 and B1—B3 (see Table 7.9-1) determine the on and off intervals of the output PWM signals. The output is generated by alternately loading the PWM count register from the maximum count register A and the maximum count register B. The timer output becomes high at the beginning of the A count, and becomes low at the beginning of the B count. This function provides a variable duty cycle wave form on the output pins PWM1—PWM3. For T8307, only PWM1 and PWM2 are bonded out.

Table 7.9-1 PWM Maximum Count Registers (PWMMAXCA1—PWMMAXCA3, PWMMAXCB1—PWMMAXCB3), Addresses (A1—A3: 0x700C5000, 0x700C500C, 0x700C505C; B1—B3: 0x700C5004, 0x700C5010, 0x700C5060)

Bit	31—16	15—0
Name	RSVD	CV

Bit	Name	Description
31—16	RSVD	Reserved.
15—0	CV	Count value.

7.9.5.2 PWM Count Registers 1, 2, and 3 (PWMCNT1—PWMCNT3)

The pulse-width modulator alternately copies the values from the PWM maximum count A and B registers to the PWM count register and counts down to 0x0001. When a pulse-width modulator channel is enabled, it restarts by loading the PWM maximum count register A into the PWM count register. When the pulse-width modulator channel is disabled, counting stops and the last PWM output value is maintained. It is not recommended that the user read or write the PWM count registers directly. Table 7.9-2 shows the format of PWM count registers[3:1].

Table 7.9-2 PWM Count Registers (PWMCNT1—PWMCNT3), Addresses (0x700C5008, 0x700C5014, 0x700C5064)

Bit	31—16	15—0
Name	RSVD	CV

Bit	Name	Description
31—16	RSVD	Reserved.
15—0	CV	Count value.

7.9.5.3 Count Rate Register (TMRCNTRATE)

The count rate register (see Table 7.9-3) contains three 4-bit fields used to select the clock periods of the three timer functions. Table 7.9-3 shows the bit field encoding used to select the clock period.

Table 7.9-3 Count Rate Register (TMRCNTRATE), Address (0x700C5018)

Bit	31—12	11—8	7—4	3—0
Name	RSVD	IDR	WDR	PDR

Bit	Name	Description
31—12	RSVD	Reserved.
11—8	IDR	Interval timer divider rate.
7—4	WDR	Watchdog timer divider rate.
3—0	PDR	PWM divider rate.

7 Call Processor (CP) Block (continued)

7.9 Timers (continued)

Table 7.9-4 Bit Encoding for Timer Divider Rates (IDR, WDR, PDR)

Bit Field	System Clock Divisor
0 0 0 0	2
0 0 0 1	4
0 0 1 0	8
0 0 1 1	16
0 1 0 0	32
0 1 0 1	64
0 1 1 0	128
0 1 1 1	256
1 0 0 0	512
1 0 0 1	1024
1 0 1 0	2048
1 0 11	4096
1 1 0 0—1 1 1 1	Reserved

7.9.5.4 WT Count Register (WTCNT)

The WT count register (see Table 7.9-5) increments at the selected count rate when the watchdog timer enable bit is set to 1 in the control register. When the watchdog timer enable bit is set to 0, the WT count register is read and written from the peripheral bus.

When the watchdog timer enable bit is set to 1 in the control register, the WT count register is not written directly from the peripheral bus. Instead, the counter is only reset to the (all 0s) value by performing a write access with a magic cookie data value.

If the watchdog timer enable bit is set to 1 and the WT count register increments to the all-1s value, the watchdog timer reset signal is asserted.

Once the watchdog timer enable bit is set in the control register, it is not reset by software. Also, the watchdog timer count rate field of the count rate register is not modifiable.

Table 7.9-5 WT Count Register (WTCNT), Address (0x700C501C)

Bit	31—16	15—0
Name	RSVD	CV

Bit	Name	Description
31—16	RSVD	Reserved.
15—0	CV	Count value.

7 Call Processor (CP) Block (continued)

7.9 Timers (continued)

7.9.5.5 IT Maximum Count Register (ITMAXC0—ITMAXC4)

The IT maximum count register (see Table 7.9-6) determines the period of the interval timer.

Table 7.9-6 IT Maximum Count Register (ITMAXC0—ITMAXC4), Addresses (0x700C5030, 0x700C5038, 0x700C5040, 0x700C5048)

Bit	31—16	15—0
Name	RSVD	CV

Bit	Name	Description
31—16	RSVD	Reserved.
15—0	CV	Count value.

7.9.5.6 IT Count Register (ITCNT0—ITCNT4)

The IT count register (see Table 7.9-7) decrements at the selected count rate when the interval timer enable bit is set to 1 in the control register. When the value of 0 is reached, the counter value reloads with the value in the maximum count register. The IT count register can be read or written at any time. It is cleared whenever the maximum count register is written.

Table 7.9-7 IT Count Register (ITCNT0—ITCNT4), Addresses (0x700C5034, 0x700C503C, 0x700C5044, 0x700C504C)

Bit	31—16	15—0
Name	RSVD	CV

Bit	Name	Description
31—16	RSVD	Reserved.
15—0	CV	Count value.

7 Call Processor (CP) Block (continued)

7.9 Timers (continued)

7.9.5.7 Status Register (TMRSR)

The status register (see Table 7.9-8) contains eight status bits, one for each interval timer channel, one for each pulse-width modulator channel, and one for watchdog timer. An interval timer channel status bit is set to 1 whenever the value in the IT compare register of the corresponding channel matches the value in the IT count register. A pulse-width modulator status bit is set to 1 at the start of each pulse-width modulator cycle. The status bits, with the exception of WTS, are reset to 0 by writing a 1 to that bit position in the status register. The WTS bit is reset by writing the magic cookie value to the watchdog count register.

Table 7.9-8 Status Register (TMRSR), Address (0x700C5024)

Bit	31—12	11	10	9	8	7—4	3	2	1	0
Name	RSVD	WTS	P3S	P2S	P1S	RSVD	I3S	I2S	I1S	I0S

Bit	Name	Description
31—12	RSVD	Reserved.
11	WTS	Watchdog timer interrupt status. If 1, watchdog timer interrupt mode is enabled and the time-out signal has been asserted.
10	P3S	PWM channel 3 status. If 1, a new PWM cycle has begun on channel 3.
9	P2S	PWM channel 2 status. If 1, a new PWM cycle has begun on channel 2.
8	P1S	PWM channel 1 status. If 1, a new PWM cycle has begun on channel 1.
7—4	RSVD	Reserved.
3	I3S	Interval timer channel 3 status. If 1, the IT count register for channel 3 has reached 0.
2	I2S	Interval timer channel 2 status. If 1, the IT count register for channel 2 has reached 0.
1	I1S	Interval timer channel 1 status. If 1, the IT count register for channel 1 has reached 0.
0	I0S	Interval timer channel 0 status. If 1, the IT count register for channel 0 has reached 0.

7 Call Processor (CP) Block (continued)

7.9 Timers (continued)

7.9.5.8 Timer Interrupt Enable Register (TMRIE)

The enable register (see Table 7.9-9) contains eight interrupt enable bits, one for each interval timer channel, one for each pulse-width modulator channel, and one for watchdog timer. Whenever an interval timer channel status bit or pulse-width modulator status bit is asserted and the corresponding enable bit is 1, the IRQ signal is asserted.

Table 7.9-9 Timer Interrupt Enable Register (TMRIE), Address (0x700C5028)

Bit	31—12	11	10	9	8	7—4	3	2	1	0
Name	RSVD	WTE	P3E	P2E	P1E	RSVD	I3E	I2E	I1E	I0E

Bit	Name	Description
31—12	RSVD	Reserved. Always write as 0.
11	WTE	Watchdog timer interrupt enable. If 1, the timer IRQ is asserted when the watchdog timer status bit is 1. If 0, the timer IRQ is not asserted from the watchdog timer.
10	P3E	PWM channel 3 interrupt enable. If 1, the timer IRQ is asserted when PWM channel 3 starts a new PWM cycle. If 0, the timer IRQ is not asserted from PWM channel 3.
9	P2E	PWM channel 2 interrupt enable. If 1, the timer IRQ is asserted when PWM channel 2 starts a new PWM cycle. If 0, the timer IRQ is not asserted from PWM channel 2.
8	P1E	PWM channel 1 interrupt enable. If 1, the timer IRQ is asserted when PWM channel 1 starts a new PWM cycle. If 0, the timer IRQ is not asserted from PWM channel 1.
7—4	RSVD	Reserved.
3	I3E	Interval timer channel 3 interrupt enable. If 1, the timer IRQ is asserted when the interval timer channel 3 status bit is 1. If 0, the timer IRQ is not asserted from interval timer channel 3.
2	I2E	Interval timer channel 2 interrupt enable. If 1, the timer IRQ is asserted when the interval timer channel 2 status bit is 1. If 0, the timer IRQ is not asserted from interval timer channel 2.
1	I1E	Interval timer channel 1 interrupt enable. If 1, the timer IRQ is asserted when the interval timer channel 1 status bit is 1. If 0, the timer IRQ is not asserted from interval timer channel 1.
0	I0E	Interval timer channel 0 interrupt enable. If 1, the timer IRQ is asserted when the interval timer channel 0 status bit is 1. If 0, the timer IRQ is not asserted from interval timer channel 0.

7 Call Processor (CP) Block (continued)

7.9 Timers (continued)

7.9.5.9 Control Register (TMRCR)

The control register (see Table 7.9-10) enabled the timers and watchdog timer reset.

Table 7.9-10 Control Register (TMRCR), Address (0x700C502C)

Bit	31—15	14	13	12	11	10	9	8
Name	RSVD	ITE3	ITE2	ITE1	ITE0	RSVD	P3E	WTI

Bit	7	6	5	4	3	2	1	0
Name	RSVD	RSVD	WIC	WTR	WRE	RSVD	P2E	P1E

Bit	Name	Description
31—15	RSVD	Reserved.
14	ITE3	Interval timer channel 3 enable. If 1, the interval timer channel 3 is enabled to count. If 0, the interval timer channel 3 is disabled.
13	ITE2	Interval timer channel 2 enable. If 1, the interval timer channel 2 is enabled to count. If 0, the interval timer channel 2 is disabled.
12	ITE1	Interval timer channel 1 enable. If 1, the interval timer channel 1 is enabled to count. If 0, the interval timer channel 1 is disabled.
11	ITE0	Interval timer channel 0 enable. If 1, the interval timer channel 0 is enabled to count. If 0, the interval timer channel 0 is disabled.
10	RSVD	Reserved. Always write as 0.
9	P3E	PWM channel 3 enable. If 1, PWM channel 3 is enabled to count. If 0, PWM channel 3 is disabled.
8	WTI	Watchdog timer interrupt mode. If 1, a watchdog time-out will generate an interrupt by setting the watchdog timer status bit. If the status bit is already asserted from a previous interrupt, a watchdog reset will occur. If 0, a watchdog time-out will cause a watchdog reset.
7	RSVD	Reserved. Always write as 0.
6	RSVD	Reserved.
5	WIC	Watchdog timer clock. If 1, the watchdog timer runs off of the 32 kHz clock. If 0, the watchdog timer runs off of the peripheral clock. **Note:** This bit resets to 0 on external reset, but is not affected by other types of reset. This bit cannot be changed once WRE is set.

7 Call Processor (CP) Block (continued)

7.9 Timers (continued)

Table 7.9-10 Control Register (TMRCR), Address (0x700C502C) (continued)

Bit	Name	Description
4	WTR	Watchdog timer reset mode. If 1, the watchdog timer registers resets only on watchdog reset. If 0, the watchdog timer registers resets on external reset, and watchdog reset. **Note:** This bit resets to 0 on external reset, but is not affected by other types of reset. This bit cannot be changed once WRE is set.
3	WRE	Watchdog timer enable. If 1, the watchdog timer count register is enabled to count and watchdog reset is enabled. If 0, watchdog reset and the watchdog timer counter are disabled.
2	RSVD	Reserved.
1	P2E	PWM channel 2 enable. If 1, PWM channel 2 is enabled to count. If 0, PWM channel 2 is disabled.
0	P1E	PWM channel 1 enable. If 1, PWM channel 1 is enabled to count. If 0, PWM channel 1 is disabled.

7.9.5.10 IT Divider Register (ITDIV)

The interval timer divider register (see Table 7.9-11) is a read-only register used for testing purposes only. The lower 12 bits of this register contain the count value. This register can be written if all four interval timers are disabled.

Table 7.9-11 IT Divider Register (ITDIV), Address (0x700C5050)

Bit	31—12	11—0
Name	RSVD	CV

Bit	Name	Description
31—12	RSVD	Reserved.
11—0	CV	Count value.

7.9.5.11 WT Divider Register (WTDIV)

The watchdog timer divider register (see Table 7.9-12) is a read-only register used for testing purposes only. The lower 12 bits of this register contain the count value. This register can be written if the watchdog timer is disabled.

Table 7.9-12 WT Divider Register (WTDIV), Address (0x700C5054)

Bit	31—12	11—0
Name	RSVD	CV

Bit	Name	Description
31—12	RSVD	Reserved.
11—0	CV	Count value.

7 Call Processor (CP) Block (continued)

7.9 Timers (continued)

7.9.5.12 PWM Divider Register (PWMDIV)

The PWM divider register (see Table 7.9-13) is a read-only register used for testing purposes only. The lower 12 bits of this register contain the count value. This register can be written if all three waveform generators are disabled.

Table 7.9-13 PWM Divider Register (PWMDIV), Address (0x700C5058)

Bit	31—12	11—0
Name	RSVD	CV

Bit	Name	Description
31—12	RSVD	Reserved.
11—0	CV	Count value.

7 Call Processor (CP) Block (continued)

7.10 Keyboard Interface

The keyboard interface consists of 12 programmable I/O pins that are configured for use in scanning a keyboard. A list of features of the keyboard interface follows:

- Maximum 6 x 6 matrix is supported.
- Pins that are not needed for the keyboard can be used as programmable I/O.
- Keyboard inputs must be active for a selectable minimum pulse-width before interrupt generation.
- Each I/O can be programmed to have an optional internal pull-up connected.

Pins that are used as general-purpose programmable I/O have the following features:

- Each bit is programmable as either an input or an output.
- Inputs are programmable to be level-sensitive or transition-detect.
- Outputs are programmable to be open-drain or direct-drive.
- Programmable polarity (inverted or not) for inputs and output.
- Each I/O can be programmed to have an optional internal pull-up connected.

7.10.1 Operation

Figure 7.10-1 shows the block diagram of the keyboard interface. The functionality of each pin is programmed independently through the keyboard data direction, keyboard sense, keyboard polarity, keyboard interrupt enable, and keyboard pull-up enable registers. The keyboard data register reads input pins and writes output pins, and can be accessed with the keyboard data clear address and keyboard data set address.

Input pins[7:0] of the keyboard interface can be used to generate interrupts from the keyboard matrix. The keyboard interrupt enable register determines which of these bits are being used to generate interrupts. It is recommended that pins that generate interrupts be programmed as level-sensitive inputs. Bits[2:0] of the keyboard control register determine the number of 32 kHz clock cycles that the input must be continuously active before a keyboard interrupt is generated. If the corresponding bit of the keyboard polarity register is 0, the input pin is active when it is 0. If the corresponding bit of the keyboard polarity register is 1, the input pin is active when it is 1.

Any keyboard interface pin that does not have its bit set to 1 in the keyboard interrupt enable register can be used for general-purpose I/O, but cannot generate an interrupt.

Any keyboard interface pin that is not programmed as an input (see Table 7.10-1) is used as an output. Outputs are programmed in the same manner whether they are being used with the keyboard or as general-purpose I/O.

7 Call Processor (CP) Block (continued)

7.10 Keyboard Interface (continued)

The keyboard data direction register controls whether a corresponding bit is an input or an output. The keyboard port sense register configures inputs to be either level-sensitive or transition-detect, and outputs to be open-drain or direct-drive. The keyboard polarity register allows both inputs and outputs to be inverted at the I/O pin. The keyboard pull-up enable register allows an internal pull-up to be connected to the pins. See Table 11.1 for the value of the pull-up resistor.

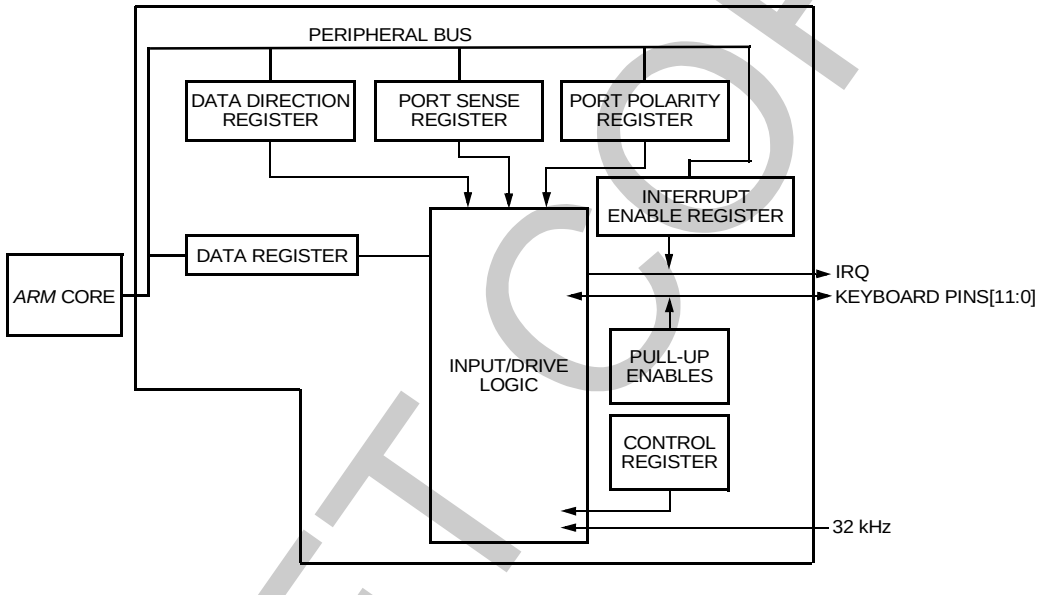


Figure 7.10-1 Block Diagram of Keyboard Interface

7.10.2 Pin Configuration on Reset

After reset, all 12 keyboard interface pins are configured as inverting level-sensitive inputs with the pull-up disabled and with interrupts disabled in the keyboard interrupt enable register.

7.10.3 Procedure for Writing to an Output Pin

1. Program the keyboard data direction register for the pin as an output.
2. Program the keyboard sense register for the output as open-drain or direct-drive.
3. Program the keyboard polarity register for the output as inverting or noninverting (relative to the keyboard data register).
4. Write a value in the keyboard data register using the keyboard data clear address or keyboard data set address to specify the output level. If the corresponding keyboard polarity register bit is 1, a 1 in the keyboard data register causes the output pin to drive high if it is programmed to be a direct-drive output, or causes the output pin to go to high impedance if it is programmed as an open-drain output. Conversely, if the corresponding keyboard polarity register bit is 0, a 1 in the keyboard data register causes both direct-drive and open-drain output pins to drive low.

Information regarding writes to the keyboard data set and clear addresses affecting input pins follows:

- A write to a level-sensitive input has no effect.
- A write of 0 to a transition-detect input has no effect.
- A write of 1 to a transition-detect input (using the keyboard data set address) clears the bit in the keyboard data register to 0 with one exception. If the input is transition-detect and if the selected edge (selected in the keyboard polarity register) occurs at the same time that a 1 is written to the bit in the keyboard data register, the write is ignored and the register bit is not cleared but is set or remains set.

7 Call Processor (CP) Block (continued)

7.10 Keyboard Interface (continued)

7.10.4 Procedure for Reading from an Input Pin

1. Program keyboard data direction register for the pin as an input.
2. Program the keyboard sense register for the input as level-sensitive or transition-detect. It is recommended that pins that generate interrupts (see Table 7.10-3) be programmed as level-sensitive.
3. Program the keyboard pull-up enable register if a pull-up is desired on the I/O pin.
4. Program the pull-up enable control 2 register to indicate whether the level on the pin is inverted before going to the keyboard data register (for level-sensitive inputs), or to indicate which edge results in a 1 appearing in the keyboard data register (for transition-detect inputs).

. If the input is changed to a transition-detect input, if the configuration of a transition-detect input is changed, or if the pin multiplexing control has changed, write a 1 to the bit in the keyboard data address (using the keyboard data set register) to clear the bit. This clears any 1 in the bit left over from when the input was programmed as level-sensitive, or that might result from transients during the configuration/multiplexing change.

5. For inputs that generate interrupts, program the keyboard polarity register to determine which level on the pin will result in an interrupt. See Section 7.10.6.5.
6. Read the keyboard data register by reading the keyboard data set address or keyboard data clear address.

If the input is configured as level-sensitive, a high value on the pin is read as 1 in the keyboard data register if the corresponding bit of the keyboard polarity register is 1. Conversely, a low value on the input is read as 1 if the corresponding bit of the keyboard polarity register is 0. Note that the keyboard data register reflects the level on the pin (possibly inverted) at the time the register is read. For inputs that generate interrupts, this may or may not be the level that caused the keyboard interrupt.

If the input is programmed to be transition-detect and the corresponding bit of the keyboard polarity register is 1, a low-to-high transition on the pin registers a value of 1 in the corresponding bit in the keyboard data register. This value is changed to 0 by writing a 1 to that same bit using the keyboard data set address, although if another low-to-high transition occurs at the same time that the 1 is being written, the data register bit is not cleared but remains set.

If the input is programmed to be transition-detect and the corresponding bit of the keyboard polarity register is 0, a high-to-low transition on the pin registers a value of 1 in the corresponding bit in the keyboard data register. This value is changed to 0 by writing a 1 to that same bit using the keyboard data set address, although if another high-to-low transition occurs at the same time that the 1 is being written, the register bit is not cleared but remains set in the data register.

When the keyboard data set address or keyboard data clear address is written, only the chip pins configured as outputs are modified in the keyboard data register; those configured as inputs are unaffected.

Input pins are asynchronous and are sampled at the system clock rate before being reflected in the keyboard data register. In order for an input signal to be registered in the keyboard data register, it must have a minimum pulse-width of two system clock periods (see Figure 7.10-2). The CLK in this figure is the system clock as defined by the clock selected in the reset/power/clock management block.

7 Call Processor (CP) Block (continued)

7.10 Keyboard Interface (continued)

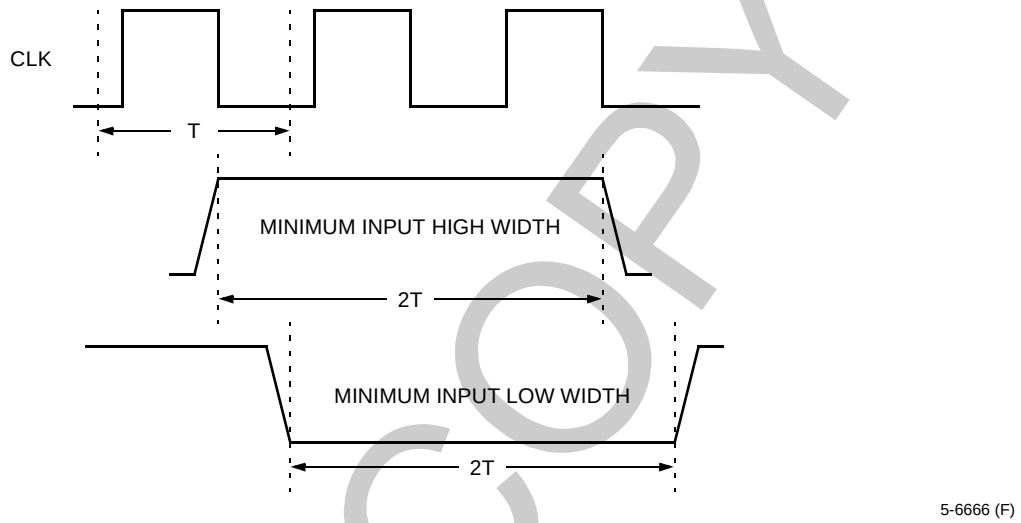


Figure 7.10-2 Minimum Input Pulse-Width Requirement for a General-Purpose Input Pin

Input pins that can generate keyboard interrupts are sampled both at the system clock rate and at the divided 32 kHz clock rate and must satisfy additional pulse-width requirements before they can generate an interrupt. See Section 7.10.5.

7.10.5 Keyboard Interrupts

The keyboard interface can generate an interrupt to the interrupt controller. Only keyboard inputs 0 through 7 can cause this interrupt. All eight inputs share a common interrupt request. The keyboard interface contains logic to generate an interrupt request when any of these inputs that are configured to generate keyboard interrupts become active. Bits[7:0] are configured to generate keyboard interrupts by using the keyboard interrupt enable register and by enabling the keyboard interrupt in the interrupt controller. Bits that are configured as general-purpose outputs or pins that are not enabled in the keyboard interrupt enable register do not affect the interrupt request signal. When a pin that is configured to cause a keyboard interrupt becomes active (the active level is the value in the keyboard polarity register), a delay counter that clocks off of the divided 32 kHz clock is started. Once the number of divided 32 kHz clock cycles specified in the keyboard control register is reached, an interrupt is generated. If the input becomes inactive at any time before the count is reached, the counter resets and an interrupt is not generated. The delay helps prevent interrupts due to noise.

If a keyboard interrupt request has been generated, no other interrupt activity will be detected on the keyboard pins until the interrupt is cleared. However, the keyboard data register will continue to reflect activity on all keyboard pins.

Since the interrupt logic operates off the divided 32 kHz clock, the keyboard can generate interrupts even if the system or peripheral bus clocks are stopped (e.g., when in WFI mode or clocks-off mode). See Section 7.2 for further information on WFI mode and clocks-off mode.

Note: Due to the fact that the keyboard input is asynchronous to the divided 32 kHz clock, the actual delay required to generate an interrupt can vary by up to one divided 32 kHz clock pulse. See Section 7.10.6.6.

There are two choices for supplying the 32 kHz clock to the keyboard interrupt logic:

1. Bring in a CMOS level 32 kHz on the X1RTC chip pin (X2RTC is grounded) and use the RTC module in bypass mode.
2. Use a crystal external to the chip, attached to the X1RTC and X2RTC pins, and use the RTC in crystal mode.

See Section 7.11 for further information on the RTC operation.

7 Call Processor (CP) Block (continued)

7.10 Keyboard Interface (continued)

To configure pins[7:0] for interrupts, it is recommended that the pin be configured as a level-sensitive input. The corresponding bit in the keyboard interrupt enable register must be set. For the interrupt to be generated, the keyboard's interrupt request signal must be enabled in the interrupt controller. Also, the keyboard control register must be programmed for the required delay before an interrupt is generated. The keyboard logic will become enabled for interrupts one divided 32 kHz clock cycle after the conditions in this paragraph are met. After this point, the input pin must be continuously active for the times indicated in the delay count field of the control register in order for an interrupt to be generated.

The keyboard interrupt request is cleared by writing a 1 to the keyboard interrupt bit in the interrupt controller's IRQ source clear register. Once an interrupt is generated, another interrupt cannot be generated until this request is cleared.

Note that the keyboard data register reflects the state of level-sensitive keyboard pins at the time the register is read, which may not be the state of the pins at the time an interrupt was generated.

See Section 7.10.8 for an example of the use of keyboard interrupts.

7.10.6 Registers

All registers except the keyboard interrupt enable register are used regardless of whether a bit is used for keyboard purposes or for general-purpose I/O. The keyboard interrupt enable register determines which of bits[7:0] can generate a keyboard interrupt.

7.10.6.1 Keyboard Data Direction Register (KBDDIR)

Table 7.10-1 Keyboard Data Direction Register (KBDDIR), Address (0x700C7000)

Bit	31—12	11—0
Name	RSVD	KDDR[11:0]

Bit	Name	Description
31—12	RSVD	Reserved.
11—0	KDDR[11:0]	Direction bits. If 1, the corresponding pin is an output; If 0, the corresponding pin is an input.

7 Call Processor (CP) Block (continued)

7.10 Keyboard Interface (continued)

7.10.6.2 Keyboard Data Register (KBDDAT)

The keyboard data register (see Table 7.10-2) reads keyboard input pins and writes keyboard output pins. When the keyboard data register is read, the bits configured as outputs reflect the value previously written to the register. The bits configured as inputs reflect the (possibly inverted) level on the input pin for level-sensitive inputs, or they reflect prior edge activity for transition-detect inputs. See Section 7.10.4.

When a new value is written to the keyboard data register, the corresponding pins that are programmed as outputs change to or stay at this value. Register bits configured as transition-detect inputs are set to zero if a 1 is written to the register bit. Register bits configured as level-sensitive inputs do not respond to writes to the register (see Section 7.10.3).

The keyboard data register is written by writing to either the keyboard data clear register (0x700C701C) or the keyboard data set register (0x700C7020). A write to the keyboard data set register writes a 1 to selected bits of the keyboard data register (those bits with a value of 1 during the write to the keyboard data set register). The other bits of the keyboard data register remain unchanged. A write to the keyboard data clear register writes a 0 to selected bits of the keyboard data register (those bits with a value of 1 during the write to the keyboard data clear register). The other bits of the keyboard data register remain unchanged.

Note that use of the keyboard data set register and keyboard data clear register allows writing selected bits of the keyboard data register using only one operation, a write to one of these two registers. No read-modify-write operations are necessary.

The keyboard data register can be read by reading either the keyboard data set register or the keyboard data clear register.

Table 7.10-2 Keyboard Data Register (KBDDAT), Addresses (Clear 0x700C701C/Set 0x700C7020)

Bit	31—12	11—0
Name	RSVD	KDAT[11:0]

Bit	Name	Description
31—12	RSVD	Reserved.
11—0	KDAT[11:0]	Data bits.

Note: Reading either address returns the value in the keyboard data register.

7.10.6.3 Keyboard Interrupt Enable Register (KBDIE)

The keyboard interrupt enable register (see Table 7.10-3) indicates which of the keyboard I/O can generate the keyboard interrupt. Only bits[7:0] can generate an interrupt. If a bit in the register is 1, is configured as an input, and if interrupts for the keyboard are enabled in the interrupt controller, then the corresponding I/O pin generates a keyboard interrupt when it is active. The keyboard polarity register value is the active level. The keyboard control register's value is the pulse-width necessary on the input to generate an interrupt.

Table 7.10-3 Keyboard Interrupt Enable Register (KBDIE), Address (0x700C7008)

Bit	31—8	7—0
Name	RSVD	KIE[7:0]

Bit	Name	Description
31—8	RSVD	Reserved.
7—0	KIE[7:0]	If 1, the corresponding keyboard interrupt is enabled. If 0, the corresponding keyboard interrupt is disabled.

7 Call Processor (CP) Block (continued)

7.10 Keyboard Interface (continued)

7.10.6.4 Keyboard Sense Register (KBDSEN)

The keyboard sense register (see Table 7.10-4) configures keyboard inputs as either level-sensitive or transition-detect, and outputs as open-drain or direct-drive. If a bit in the register is 0, the corresponding input pin is level-sensitive, or the corresponding output pin is direct-drive if a 3-state I/O buffer is used. If a bit in the register is 1, the corresponding input pin is transition-detect, or the output pin is open-drain when a 3-state I/O buffer is used. If an open-drain I/O buffer is used for a pin, that pin will be open-drain when it is an output, regardless of the setting of the keyboard sense register bit.

Table 7.10-4 Keyboard Sense Register (KBDSEN), Address (0x700C700C)

Bit	31—12	11—0
Name	RSVD	KSEN[11:0]

Bit	Name	Description
31—12	RSVD	Reserved.
11—0	KSEN[11:0]	Sense bits.

7.10.6.5 Keyboard Polarity Register (KBDPOL)

The keyboard polarity register (see Table 7.10-5) is used to specify inversion of both input and output signals and is also used to specify the active level on a keyboard input necessary to generate an interrupt. As a reference, logic signals in the keyboard data register are considered to be positive, or noninverted. A value of 0 in a keyboard polarity register causes a signal entering or leaving the device on the pin to be inverted before being reflected in the keyboard data register, thereby conforming to a negative, or inverted, signal convention outside the device. Conversely, a value of 1 in the polarity register causes a signal entering or leaving the device on the pin to be simply buffered before being reflected in the keyboard data register, thereby conforming to a positive, or noninverted, signal convention. The interpretation of the register bits differs somewhat for transition-detect inputs and for inputs used to generate interrupts, as described in the following paragraphs.

For a level-sensitive input, a value of 1 in the keyboard polarity register results in the value on the input pin being placed in the corresponding keyboard data register bit (noninverted, level-sensitive input), while a value of 0 in the keyboard polarity register results in the value on the pin being inverted before being placed in the corresponding keyboard data register bit.

For inputs that can generate an interrupt, the keyboard polarity register determines the active level on the input that will result in an interrupt, as well as whether or not the input will be inverted before appearing in the keyboard data register. Note that the inputs that generate an interrupt must be programmed to be level-sensitive. A value of 0 in the keyboard polarity register specifies that the active level on the pin (to generate an interrupt) is 0 and that the value on the pin will be inverted before being placed in the keyboard data register. A value of 0 in the keyboard polarity register is typically used to detect a key press. A value of 1 in the keyboard polarity register specifies that the active level on the pin (to generate an interrupt) is 1 and that the value on the pin will be placed in the corresponding keyboard data register bit. A value of 1 in the keyboard polarity register is typically used to detect a key release.

For a transition-detect input, a value of 1 in a keyboard polarity register selects detection of a low-to-high transition at the pin (rising transition-detect input). Conversely, a value of 0 selects detection of a high-to-low transition at the pin (falling transition-detect input). The selected transition results in a 1 in the corresponding keyboard data register.

For a direct-drive output, a 1 in the appropriate bit of the keyboard polarity register results in the value in the keyboard data register being driven to the chip pin, while a 0 in the appropriate bit of the keyboard polarity register results in the inverse of the register value being driven to the pin.

7 Call Processor (CP) Block (continued)

7.10 Keyboard Interface (continued)

For an open-drain output, a 1 in the appropriate bit of the keyboard polarity register results in the chip pin being driven to a 0 if there is a 0 in the corresponding keyboard data register, and results in the chip pin going to high impedance if there is a 1 in the keyboard data register. For an open-drain output, a 0 in the appropriate bit of the keyboard polarity register results in the chip pin being driven to high impedance if there is a 0 in the corresponding keyboard data register, and results in the chip pin being driven to 0 if there is a 1 in the keyboard data register. A summary of the use of the keyboard polarity register and other keyboard registers is shown in Table 7.10-5.

Table 7.10-5 Keyboard Polarity Register (KBDPOL), Address (0x700C7010)

Bit	31—12	11—0
Name	RSVD	KPOL[11:0]

Bit	Name	Description
31—12	RSVD	Reserved.
11—0	KPOL[11:0]	Polarity bits.

7.10.6.6 Keyboard Control Register (KBDCNTL)

Table 7.10-6 shows the format of the keyboard control register.

Table 7.10-6 Keyboard Control Register (KBDCNTL), Address (0x700C7018)

Bit	31—4	3	2—0
Name	RSVD	IRQ	DC[2:0]

Bit	Name	Description
31—4	RSVD	Reserved.
3	IRQ	Interrupt status. Indicates that a keyboard interrupt has been generated. This bit is cleared when the keyboard interrupt is cleared in the interrupt controller.
2—0	DC[2:0]	Delay count. Determines the number of divided 32 kHz cycles the input must be continuously active before an interrupt is generated. The divisor is determined by the keyboard debounce register in the RST block. The active level is the value of the input's polarity bit. Table 7.10-7 shows the decode of the delay count field. An interrupt might be generated after the input is continuously active for the minimum number of divided 32 kHz cycles. An interrupt will definitely be generated if the input is continuously active for the maximum number of divided cycles. The uncertainty results from the asynchronous nature of the input relative to the divided 32 kHz clock.

Table 7.10-7 Delay Count Field

Value	Minimum Number of Cycles	Maximum Number of Cycles
0 0 0	1	2
0 0 1	2	3
0 1 0	3	4
0 1 1	4	5
1 0 0	5	6
1 0 1	6	7
1 1 0	7	8
1 1 1	8	9

7 Call Processor (CP) Block (continued)

7.10 Keyboard Interface (continued)

7.10.7 Summary of Programming Modes

Table 7.10-8 Programming Modes Summary

Keyboard Data Direction Register	Keyboard Sense Register	Keyboard Polarity Register	Keyboard Function
0	0	0	Inverted, level-sensitive input. Active level for interrupt = 0.
0	0	1	Noninverted, level-sensitive input. Active level for interrupt = 1.
0	1	0	Falling transition-detect input.
0	1	1	Rising transition-detect input.
1	0	0	Inverting, direct-drive output.
1	0	1	Noninverting, direct-drive output.
1	1	0	Inverting, open-drain output.
1	1	1	Noninverting, open-drain output.

7.10.8 Example of Software Usage of Keyboard Interface

The following example of software usage has these characteristics:

- The keyboard is physically configured as a 6 x 6 matrix with 6 rows and 6 columns of keys.
- Each row of keys corresponds to one keyboard input pin.
- Each column of keys corresponds to one keyboard output pin.
- Each key can be identified as the intersection of a row and a column.
- There are pull-up resistors on both the input and output pins.
- If an output for a column is low when a key is pressed in that column, then the low on the output is transferred to the input associated with the key's row. Therefore, a low on an input corresponds to a key press, and a high on an input corresponds to a key release.

The polarity bit for an input determines which input level generates an input. When the polarity bit is low for the input, a low on the input pin (i.e., a key press) generates an interrupt if the low value is continuously present for the time indicated by the delay count field in the control register. When the polarity bit is high for the input, a high on the input pin (i.e., a key release) for the required time generates an interrupt.

In this example, the keyboard debounce register in the RST block and the delay count register are programmed to generate a keyboard interrupt only when a keypress has been stable for a predetermined time. The predetermined time depends on the specs of the keyboard being used.

Note: A typical key press lasts for hundreds of milliseconds.

The following is an outline of the software usage of the keyboard interface for this example.

Setup:

1. Write the appropriate value to the keybounce timer control register (found in the reset block) to enable clock and debounce.
2. Program the RTC portion of the chip appropriately to ensure that the 32 kHz clock output from the RTC oscillator is active.
3. Program the appropriate keyboard registers for keyboard bits[5:0] to be level-sensitive inputs and for keyboard bits[11:6] to be open-drain outputs.
4. Clear the keyboard polarity bits for all six inputs to 0; initially it will be looking for low voltage on inputs (key press).
5. Program the pull-up enable control 2 register to enable pull-up resistors on keyboard bits[5:0] and bits[11:6].
6. Program the keyboard control register with the desired delay count.
7. Program the interrupt controller to enable the keyboard interrupt. Also for this example, program the interrupt controller to detect a high level on the IRQ signal.

7 Call Processor (CP) Block (continued)

7.10 Keyboard Interface (continued)

8. Program the interrupt controller's priority control register 1 to make the keyboard interrupt the highest priority interrupt (for this example).
9. Enable keyboard interrupts by setting the bit in the interrupt controller's interrupt request enable register.
10. Enable interrupts for keyboard pins 0 through 5 by writing a 1 to these bits in the keyboard interrupt enable register. Note that these bits remain 1 during all subsequent operation in this example.

Operation:

11. Make all six outputs low while waiting for key press.
12. Get interrupt due to key press.
13. Disable interrupts for keyboard by writing 0 to the appropriate bit in the interrupt controller's interrupt request enable register.
14. Clear the interrupt from the interrupt controller and the keyboard logic by writing to the interrupt request source clear register in the interrupt controller.
15. To determine which key(s) resulted in the interrupt, scan the keys, which consists of making all six outputs go high and then making each output go low, one at a time. While each output is low, read the six inputs from the keyboard data register. If an input is low, the numbers of this input and of the output determine the key that was pressed. Note that because the polarity bit is 0, a low on the input will read as 1 in the data register.
16. Back to normal; all six outputs low.
17. Make the keyboard polarity bit high for the input that caused the interrupt; keyboard logic will next look for key release.
18. Enable interrupts for keyboard by writing 1 to the bit in the interrupt controller's interrupt request enable register. Note that it is important that the polarity bit had been written to 1 before this point. If not, the keyboard would immediately give another interrupt for the same key press, since the key continues to be pressed at this point.
19. Get interrupt for key released (probably several hundred milliseconds after the previous step).
20. Disable interrupts for keyboard by writing 0 to the bit in the interrupt controller's interrupt request enable register.
21. Clear the interrupt by writing to the interrupt request source clear register (clears the interrupt out of the interrupt controller and the keyboard logic).
22. Make all six keyboard polarity bits low; keyboard logic will next look for key press.
23. Enable interrupts for keyboard by writing 1 to the bit in the interrupt controller's interrupt request enable register. Note that it is important that the polarity bits had been written to 0 before this point. If not, the keyboard would immediately give another interrupt since no key has yet been pressed at this point.
24. Get interrupt due to next key press.

7 Call Processor (CP) Block (continued)

7.11 Real-Time Clock (RTC)

The real-time clock (RTC) is driven by a 32.768 kHz clock from a crystal oscillator. The input clock is divided by 32,768 to generate a clock with a 1 second period that increments a 29-bit seconds counter. In addition, it can generate interrupts at a programmed time. The RTC input pins are X1RTC and X2RTC. The following are the features of RTC:

- 17-year time interval with 1 second resolution.
- Programmed time alarm interrupt.
- Alarm output pin.

7.11.1 Operation

The RTC consists of a seconds counter. The input clock frequency to the RTC is 32.768 kHz. The seconds counter is updated using a clock generated by dividing the external input clock to the RTC by 32,768. An option to use the system clock is also provided for a manufacturing test.

To set an alarm, load an appropriate value into the seconds alarm register and enable interrupt IRQ13 in the interrupt enable register. The RTC control register bit 2 is 1 if the interrupt was due to the seconds alarm.

The RTC assumes an uninterrupted power supply, V_{RTC}. The voltage range for V_{RTC} is between 1.50 V—1.65 V whether V_{DD}_CORE is off or on. The RTC function is specially designed to handle the situation where V_{DD}_CORE to the microcontroller is between 0 V—1.5 V. Note that the RESETN and RTCALARMN pin operates from the V_{RTC} supply, and, therefore, utilize 1.5 V logic level. OSC32OUT operates from the V_{DD}_IO_1P8 power supply. It is the responsibility of the external logic to select either the normal power supply or a backup battery to provide uninterrupted V_{RTC}. Switching to backup power must be done in the following sequence:

1. Assert the RESETN signal.
2. Switch V_{RTC} to backup power.
3. Turn off 1.5 V and 1.8 V power supplies and maintain V_{RTC}.

Switching back to primary power has to be done in the following sequence:

1. Turn on 1.5 V and 1.8 V power supplies.
2. Switch V_{RTC} back to primary power.
3. Deassert the RESETN signal.

The internal logic is used by the microcontroller to force internal inputs to the RTC to appropriate levels in order to ensure correct functionality and reduce power dissipation.

If the RTC is not used in the system, the following steps should be followed to eliminate any unnecessary power dissipation:

1. Write the value 0x01 to the control register. This disables the crystal oscillator circuit, the divider, and the seconds counter.
2. Set bit 13 of the power management register in the reset, power, and clock management block to 1 to disable the system clock used in the RTC.

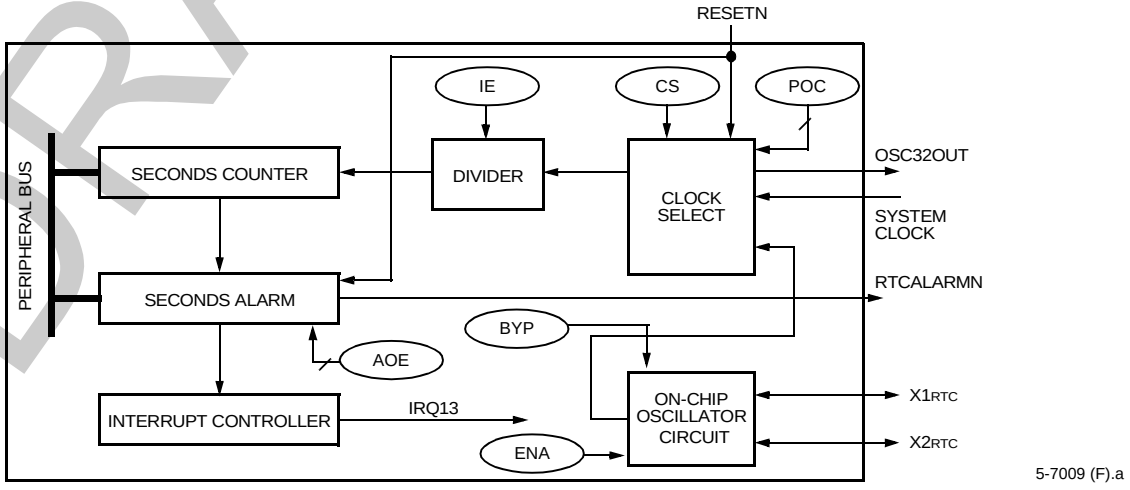


Figure 7.11-1 Functional Block Diagram of RTC

7 Call Processor (CP) Block (continued)

7.11 Real-Time Clock (RTC) (continued)

7.11.2 Registers

The RTC consists of seconds counter, seconds alarm, divider, and control registers. The user can read, but not write, the seconds counter register when it is being incremented. If a write to the seconds counter register is attempted during an update cycle, bit 31 is set and IRQ13 is asserted. The duration of the update cycle is 122 µs (i.e., four RTC clock periods). These registers are not changed by any RESET condition, and their stored values are undefined after a powerup sequence. The RTC circuitry remains enabled and active while the rest of the microcontroller is powered down.

7.11.2.1 Clock Control Register (RTCCNTL)

The control register (see Table 7.11-1) is used to select the clock source, enable the RTC counters, and provide status information on the alarm interrupt and illegal updates to the divider register or seconds counter register.

Table 7.11-1 Clock Control Register (RTCCNTL), Address (0x700CC000)

Bit	31—11	10—9	8	7	6—5	4	3	2	1	0
Name	RSVD	AOE	RSVD	ENA	POC	BYP	IWI	AI	IE	CS

Bit	Name	Description
31—11	RSVD	Reserved.
10—9	AOE	RTCALARMN 3-state enable control. Operates in conjunction with the RESETN pin (see Table 7.11-2). **Note:** RTCALARMN is powered by the VRTC supply.
8	RSVD	Reserved.
7	ENA	Enables the on-chip oscillator circuit. If 1, the on-chip oscillator circuit is powered and generating a clock signal. An external 32.7687 kHz crystal must be connected between X1RTC and X2RTC pins as shown in Figure 7.11-2 for proper operation. If 0, the on-chip oscillator circuit is powered off. **Note:** This bit should be cleared (0) when bypass mode is selected (BYP = 1) or for power savings when the RTC is not used.
6—5	POC	OSC32OUT control. In conjunction with the RESETN pin, determines the value on the OSC32OUT pin (see Table 7.11-3). **Note:** OSC32OUT is powered by the VDD_IO_1P8 supply.

7 Call Processor (CP) Block (continued)

7.11 Real-Time Clock (RTC) (continued)

Table 7.11-1 Clock Control Register (RTCCNTL), Address (0x700CC000) (continued)

Bit	Name	Description
4	BYP	Bypasses the on-chip oscillator circuit. This controls a MUX at the output of the oscillator circuit. If 1, an externally-generated clock signal input on the X1RTC pin replaces the on-chip generated clock at the oscillator circuit output. X2RTC must be grounded for proper operation. The ENA field should be cleared (0) when BYP is set. If 0, the clock generated by the on-chip oscillator circuit is selected as the output of the oscillator circuit. **Note:** Clearing BYP does not automatically power off the on-chip oscillator circuit. Use the ENA field for power control.
3	IWI	Illegal write interrupt. Indicates the illegal write status. If 1, a write operation on the seconds counter or divider register occurred during an update cycle when the divider register was enabled. If 0, an illegal write did not occur. This bit is reset by writing a 1 to it. This condition causes an interrupt if the IRQ13 interrupt is enabled in the interrupt controller.
2	AI	Alarm interrupt. Indicates the alarm interrupt status. If 1, an interrupt occurred due to the seconds alarm. If 0, the seconds alarm did not occur. This bit is reset by writing a 1 to it. This condition causes an interrupt if the IRQ13 interrupt is enabled in the interrupt controller.
1	IE	Increment enable. Enables incrementing of the divider register. If 1, the divider register is enabled to increment. If 0, the divide register is disabled.
0	CS	Clock select. Selects the source of the clock for the divider register. If 1, the clock is from the oscillator circuit or the X1RTC pin. If 0, the input clock is the system clock.

7 Call Processor (CP) Block (continued)

7.11 Real-Time Clock (RTC) (continued)

7.11.2.2 RTCALARMN Control and Encoding

The RTCALARMN pin is controlled in two different ways. When bit 10 of the control register is 0, the output is controlled by bit 9. When bit 10 is 1, the output is controlled by the RESETN pin.

Table 7.11-2 RTCALARMN Control and Encoding

Bit 10	Bit 9	RESETN	RTCALARMN Output
0	0	Don't care	3-state, high Z
0	1		Alarm signal
1	Don't care	0	3-state, high Z
1		1	Alarm signal

7.11.2.3 OSC32OUT Control and Encoding

The OSC32OUT pin is controlled in two different ways. When bit 6 of the control register is 0, the output is controlled by bit 5. When bit 6 is 1, the output is controlled by the RESETN pin.

Table 7.11-3 OSC32OUT Control and Encoding

Bit 6	Bit 5	RESETN	OSC32OUT Output
0	0	Don't care	3-state, high Z
0	1		RTC clock
1	Don't care	0	3-state, high Z
1		1	RTC clock

7.11.2.4 Seconds Alarm Register (RTCSECA)

IRQ13 is asserted when the values in the seconds alarm (see Table 7.11-4) and seconds counter (see Table 7.11-5) registers are equal and the RTC interrupt is enabled in the interrupt controller. The alarm condition is distinguished by examining bit 2 of the control register. Table 7.11-4 shows the format of the seconds alarm register.

Table 7.11-4 Seconds Alarm Register (RTCSECA), Address (0x700CC004)

Bit	31—29	28—0
Name	RSVD	SA

Bit	Name	Description
31—29	RSVD	Reserved.
28—0	SA	Represents time in seconds.

7 Call Processor (CP) Block (continued)

7.11 Real-Time Clock (RTC) (continued)

7.11.2.5 Seconds Counter Register (RTCSECC)

The seconds counter register bits[28:0] (see Table 7.11-5) shows the time in seconds. The seconds counter register is typically incremented once per second. If bit 31 of the seconds counter register is set when read, then the other bits are invalid, which means the user must retry the read or write operation after the RTC clears the UCP bit.

Table 7.11-5 Seconds Count Register (RTCSECC), Address (0x700CC008)

Bit	31	30—29	28—0
Name	UCP	RSVD	SC

Bit	Name	Description
31	UCP	Bit 31 is used to indicate that an update cycle is in progress. If 1, an update cycle was in progress when the read access occurred. If 0, the returned value was stable.
30—29	RSVD	Reserved.
28—0	SC	Seconds continued represents time in seconds.

7.11.2.6 Divider Register (RTCDIV)

The divider register (see Table 7.11-6) contains the current count of input clocks that have occurred since the last change in the seconds register. The source of the clock to the divider register is selected by control register bits 0 and 4. The divider register is reset to 0x0000 whenever the seconds count register is written. The divider register is written by the core (for testing purposes) only when the RTC is disabled by setting the control register bit 1 to 0. If the CPU attempts to write into the divider register while it is enabled to increment, an interrupt is generated if enabled, and the illegal write bit in the control register is set to 1.

Table 7.11-6 Divider Register (RTDIV), Address (0x700CC00C)

Bit	31—15	14—0
Name	RSVD	Clock cycle count (CCC)

Bit	Name	Description
31—15	RSVD	Reserved.
14—0	CCC	1/32, 768 of a second.

7.11.3 Operation with External Crystal

Figure 7.11-2 shows the basic connection diagram for the 32 kHz crystal oscillator when used with a crystal.

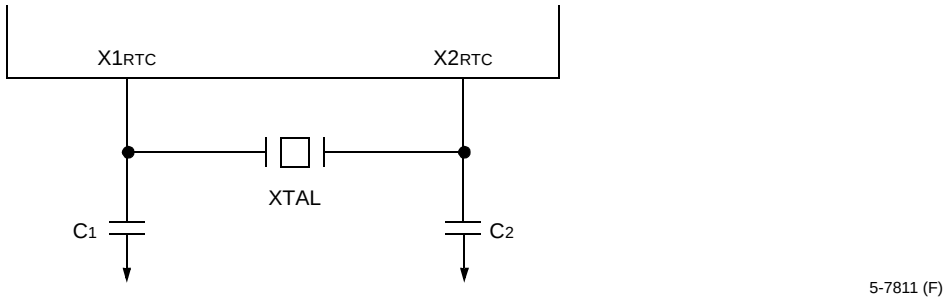


Figure 7.11-2 32.768 kHz Crystal Configuration

7 Call Processor (CP) Block (continued)

7.11 Real-Time Clock (RTC) (continued)

There is a start-up time of up to several seconds associated with the crystal oscillator circuit when an actual crystal is being used.

The crystal oscillator utilizes the V_{RTC} supply pin. This pin should be bypassed as close to the device as possible. Note that when an actual 32 kHz crystal is in use, the X1RTC and X2RTC pins will be operating with a very low-level (approximately 200 mVpp) signal present, and, therefore, care must be taken to avoid coupling noise into these signals. Care should also be taken in the layout of the printed-circuit board to minimize the trace length of these signals and to avoid coupling from digital signals with fast edge rates.

Table 7.11-7 lists requirements for the external components to be used with the 32 kHz oscillator circuit.

Table 7.11-7 32.768 kHz Oscillator External Component Requirements

Parameter	Min	Typ	Max	Unit
Crystal Frequency	32	—	33	kHz
External Capacitors, C_1, C_2	21	25	27	pF
Crystal Internal Resistance	—	—	50	k.
Crystal Motional Capacitance	—	2	—	fF
Crystal Shunt Capacitance	—	0.85	—	pF
Crystal Load Capacitance	10.5	12.5	13.5	pF
Crystal Maximum Drive Level	1	—	—	∝W

Note that the crystal load capacitance is equal to the series combination of C_1 and C_2.

Recommended crystals include those in Table 7.11-8.

Table 7.11-8 Recommended Crystals

Manufacturer	Model Number
Epson® America	MC-405 or MC-156

The electrical specifications of the crystal oscillator circuit can be found in Section 11.2.

7 Call Processor (CP) Block (continued)

7.12 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support

7.12.1 Operation Modes

T8307 *ARM*-side SSPI^2S port supports all features of *ARM Primecell* PL022 (e.g., *Motorola* SPI, *Texas Instruments SSI*, and *National Semiconductor MICROWIRE*) and *Philips* I^2S formats. See Section 7.12.3 through Section 7.12.5 for SPI, SSI, and I^2S formats, and refer to *ARM PrimeCell* PL022 document for *MICROWIRE* format.

In SSP modes (SPI, SSI, and *MICROWIRE*), the *ARM*-side serial bus interface consists of the following four pins: SPTXD0_I2SD, SPRXD0, SPCLK0, and SPFS0. Dynamic master/slave switching capability is provided for SPI, SSI, MW modes because these modes use separate transmit and receive data pins. This feature allows the user to switch the function of SPTXD0_I2SD and SPRXD0 such that the SSPI^2S port can be configured as master or slave without changing pin connections on the board. In particular, if this function is enabled (by setting DS bit field of SSPCR0 to 0 which is the default value) and if the slave mode is selected (by setting MS bit field of SSPCR0 to 1), the SPTXD0_I2SD pin is an input pin while the SPRXD0 pin is an output pin. See Table 7.12-1 for a summary of the input/output status for all options.

Table 7.12-1 Functions of the SSP Bus Interface Pins

MS (SSPCR0 Bit 2)	DS (SSPCR0 Bit 7)	SPTXD0_I2SD Pin	SPRXD0 Pin	SPFS0 Pin	SPCLK0 Pin
0 (default)	0 (default)	Output	Input	Output	Output
1	0	Input	Output	Input	Input
0	1	Output	Input	Output	Output
1	1	Output	Input	Input	Input

In I^2S mode, the interface consists of three pins: SPTXD0_I2SD, SPCLK0 and SPFS0. The function of these pins are determined by the MS bit and the I^2STX bit of SSPCR0 register as summarized in Table 7.12-2.

Table 7.12-2 Functions of the I^2S Bus Interface Pins

MS (SSPCR0Bit 2)	I^2STX (SSPCR0 Bit 6)	SPTXD0_I2SD Pin	SPFS0 Pin	SPCLK0 Pin
0 (default)	0(default)	Input	Output	Output
1	0	Input	Input	Input
0	1	Output	Output	Output
1	1	Output	Input	Input

In both SSP modes and I^2S mode, the SSPI^2S supports programmable data sizes of 4 bits to 16 bits. To ensure correct device operation, the maximum expected frequency of SPCLK0 should not exceed 1/24 of the *ARM*-system clock frequency when SPCLK0 is configured as an input pin. In addition, the polarity of the clock signal to or from SPCLK0 pin are programmable through SSPCR0 register.

7 Call Processor (CP) Block (continued)

7.12 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

7.12.2 Interrupts

The SSP/I^2S block generates interrupt request (IRQ14, see Table 7.5-2 for *ARM* interrupt vector assignments) based on the status of transmit and receive FIFOs. Both the transmit and receive FIFOs are 16-bit wide, 8-location deep. Data from the *ARM* core is stored in the transmit FIFO until read out by the transmit logic, while received data from the serial interface are stored in the receive FIFO until read out by the *ARM* core.

IRQ14 is asserted if any of the four individual interrupts below are asserted and enabled. The status of the individual interrupt sources are maskable and can be read from SSPRIS and SSPMIS registers.

7.12.2.1 Receive FIFO Service Interrupt Request (SSPRXINTR)

The receive interrupt is asserted when there are four or more valid entries in the receive FIFO.

7.12.2.2 Transmit FIFO Service Interrupt Request (SSPTXINTR)

The transmit interrupt is asserted when there are four or less valid entries in the transmit FIFO. The transmitter interrupt SSPTXINTR is not qualified with the SSP enable signal, which allows operation in one of two ways. Data can be written to the transmit FIFO prior to enabling the SSPI^{2}S and the interrupts. Alternatively, the SSPI^{2}S and interrupts can be enabled so that data can be written to the transmit FIFO by an interrupt service routine.

7.12.2.3 Receive Overrun Interrupt Request (SSPRORINTR)

The receive overrun interrupt SSPORINTR is asserted when the FIFO is already full and an additional data frame is received, causing an overrun of the FIFO. Data is overwritten in the receive shift register but not the FIFO.

7.12.2.4 Time-Out Interrupt Request (SSPRTINTR)

The receive time-out interrupt is asserted when the receive FIFO is not empty and the SSPI^{2}S has remained idle for a fixed 32-bit period. This mechanism ensures that the user is aware that data is still present in the receive FIFO and requires servicing.

7 Call Processor (CP) Block (continued)

7.12 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

7.12.3 SSI

To operate in SSI mode, set FRF bit field of control register 0 (SSPCR0) to binary 01. In master mode, SPCLK0 and SPFS0 are forced low, and the transmit data line SPTXD0_I2SD is 3-stated whenever the SSPI^{2}S is idle. Once the bottom entry of the transmit FIFO contains data, SPFS0 is pulsed high for one SPCLK0 clock period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of SPCLK0, the MSB of the 4-bit to 16-bit data frame is shifted out on SPTXD0_I2SD. Likewise, the MSB of the received data is shifted onto SPRXD0 by the off-chip serial slave device. Both the SSPI^{2}S and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each SPCLK0. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SPCLK0 after the LSB has been latched.

Figure 7.12-1 shows the SSI frame format for a single transmitted frame.The nSSPOE signal is the internal output enable control for transmit pin SPTXD0_I2SD, in this case.

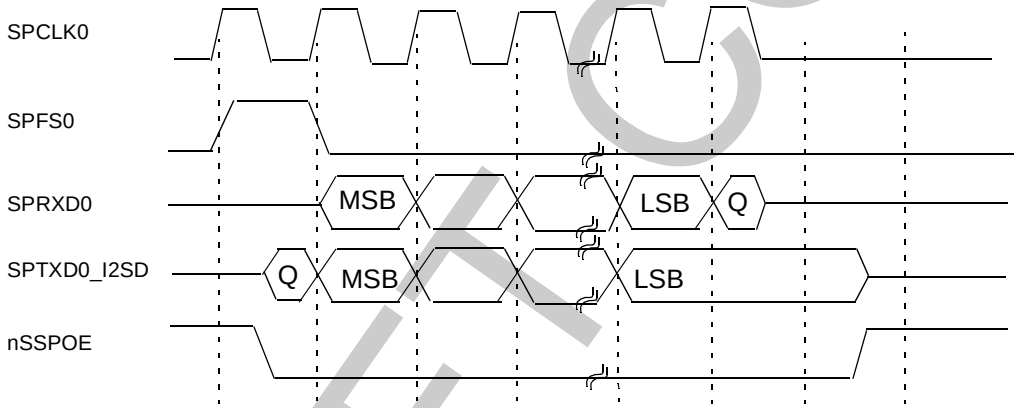


Figure 7.12-1 *Texas Instruments* Synchronous Serial Frame Format (Single Transfer)

Figure 7.12-2 shows the SSI frame format when back-to-back frames are transmitted

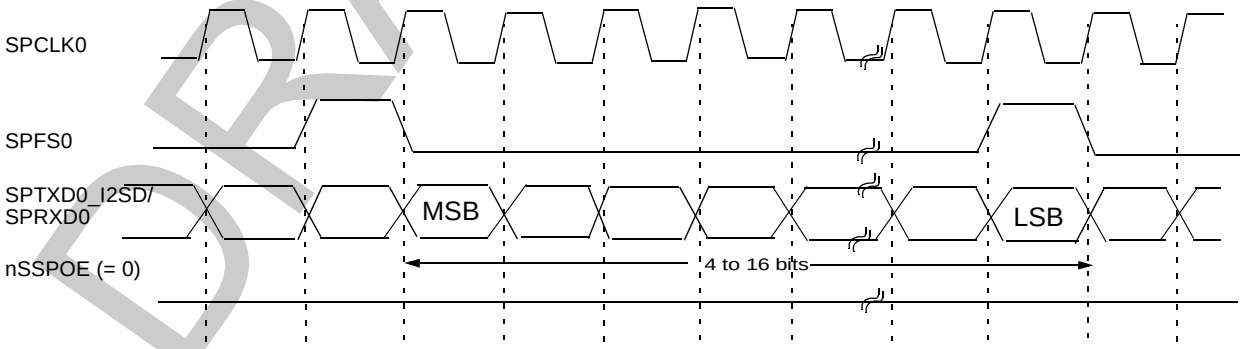


Figure 7.12-2 *Texas Instruments* Synchronous Serial Frame Format (Continuous Transfer)

7 Call Processor (CP) Block (continued)

7.12 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

7.12.4 SPI

To operate in SPI mode, set FRF bit field of SSP control register 0 (SSPCR0) to binary 00. In this mode, the SPFS0 signal behaves as a slave select. Another feature is that the inactive state and phase of the SPCLK0 signal are programmable through the SPO and SPH bits within the SSPSCR0 control register.

When the SPO clock polarity control bit is low, it produces a steady state low value on SPCLK0. If the SPO clock polarity control bit is high, a steady-state high value is placed on SPCLK0 when data is not being transferred.

The SPH control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge.When the SPH phase control bit is low, data is captured on the first clock edge transition. If the SPH clock phase control bit is high, data is captured on the second clock edge transition.

7.12.4.1 *Motorola* SPI Format with SPO = 0, SPH = 0

Single and continuous transmission signal sequences for *Motorola* SPI format with SPO = 0, SPH = 0 are shown in Figure 7.12-3 and Figure 7.12-4.

In this configuration, during idle periods, the following occurs:

- The SPCLK0 pin is forced low in master mode, or high impedance in slave mode.
- SPFS0 is forced high.
- The transmit data line SPTXD0_I2SD is high impedance.

If the SSPI^{2}S is enabled and there is valid data in the transmit FIFO, the start of transmission is signified by the SPFS0 master signal being driven low. This causes slave data to be enabled onto the SPRXD0 line of the master. The master SSPTXD output is enabled. One-half SPCLK0 clock period later, valid master data is transferred to SPTXD0_I2SD. Now that both the master and slave data have been set, the SPCLK0 master clock goes high after one further half SPCLK0 period. The data is now captured on the rising and propagated on the falling edges of the SPCLK0 signal.

In the case of a single word transmission, after all bits of the data word have been transferred, the SPFS0 pin is returned to its idle high state one SPCLK0 period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SPFS0 signal must be pulsed high between each data word transfer. This is because the slave select signal freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is logic zero. Therefore, the master device must raise the SPFS0 signal for the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, SPFS0 is returned to its idle state one SPCLK0 period after the last bit has been captured.

7 Call Processor (CP) Block (continued)

7.12 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued).

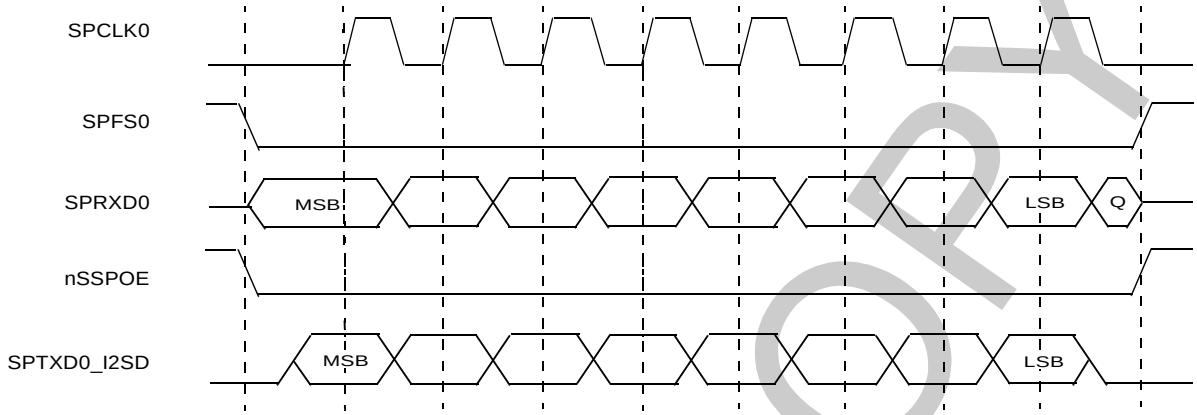


Figure 7.12-3 *Motorola* SPI Frame Format (Single Transfer) SPO = 0, SPH = 0

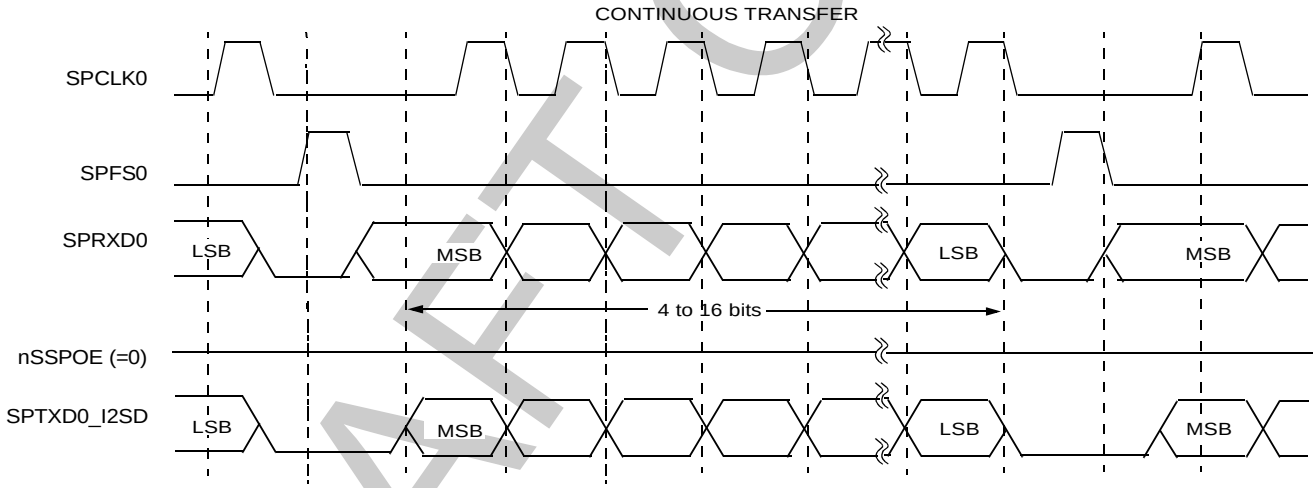


Figure 7.12-4 *Motorola* SPI Frame Format (Continuous Transfer) SPO = 0, SPH = 0

7 Call Processor (CP) Block (continued)

7.12 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

7.12.4.2 *Motorola* SPI Format with SPO = 0, SPH = 1

The transfer signal sequence for *Motorola* SPI format with SPO = 0, SPH = 1 is shown in Figure 7.12-5, which covers both single and continuous transfers.

In this configuration, during idle periods, the following occurs:

- The SPCLK0 pin is forced low in master mode, or high impedance in slave mode.
- SPFS0 is forced high.
- The transmit data line SPTXD0_I2SD is high impedance.

If the SSPI^{2}S is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SPFS0 master signal being driven low. The nSSPOE line is driven low, enabling the master SPTXD0_I2SD output. After a further one-half SPCLK0 period, both master and slave valid data is enabled onto their respective transmission lines. At the same time, the SPCLK0 is enabled with a rising edge transition. Data is then captured on the falling edges and propagated on the rising edges of the SPCLK0 signal.

In the case of a single word transfer, after all bits have been transferred, the SPFS0 line is returned to its idle high state one SPCLK0 period after the last bit has been captured.

For continuous back-to-back transfers, SPFS0 is held low between successive data words and termination is the same as that of the single word transfer.

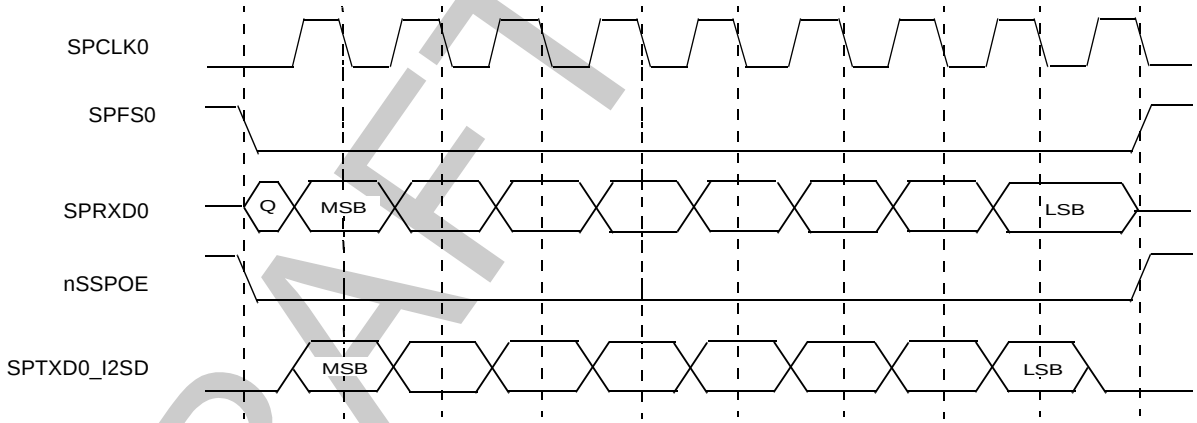


Figure 7.12-5 *Motorola* SPI Frame Format SPO = 0, SPH = 1

7 Call Processor (CP) Block (continued)

7.12 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

7.12.4.3 *Motorola* SPI Format with SPO = 1, SPH = 0

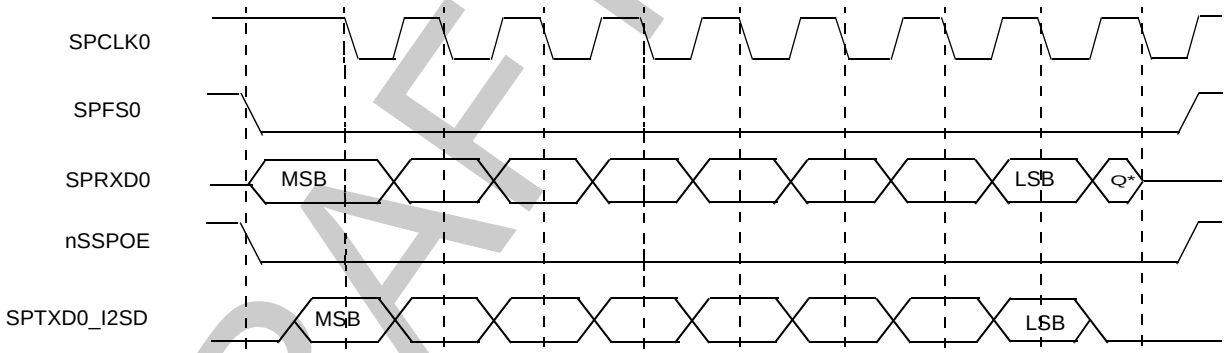
Single and continuous transmission signal sequences for *Motorola* SPI format with SPO = 1, SPH = 0 are shown in Figure 7.12-6 and Figure 7.12-7. In this configuration, during idle periods, the following occurs:

- The SPCLK0 pin is forced high in master mode, or high impedance in slave mode.
- SPFS0 is forced high.
- The transmit data line SPTXD0_I2SD is high impedance.

If the SSPI^{2}S is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SPFS0 master signal being driven low, which causes slave data to be immediately transferred onto the SPRXD0 line of the master. The nSSPOE line is driven low, enabling the master SPTXD0_I2SD output. One-half period later, valid master data is transferred to the SPTXD0_I2SD line. Now that both the master and slave data have been set, the SPCLK0 master clock signal becomes low after one further half SPCLK0 period. This means that data is captured on the falling edges and be propagated on the rising edges of the SPCLK0 signal.

In the case of a single-word transmission, after all bits of the data word are transferred, the SPFS0 line is returned to its idle high state one SPCLK0 period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SPFS0 signal must be pulsed high between each data word transfer. This is because the slave select signal freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is logic 0. Therefore, the master device must raise the SPFS0 signal for the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SPFS0 signal is returned to its idle state one SPCLK0 period after the last bit has been captured.



* Q is an undefined signal.

Figure 7.12-6 *Motorola* SPI Frame Format (Single Transfer) with SPO = 1, SPH = 0

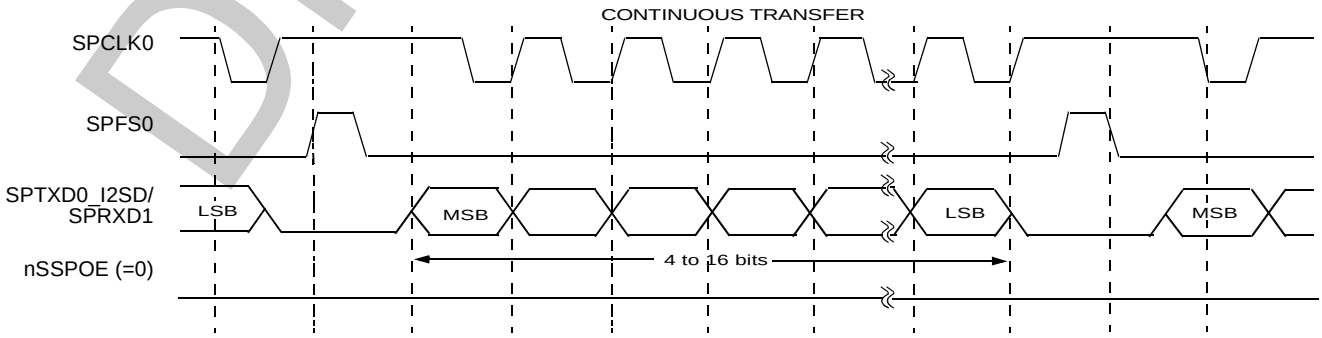


Figure 7.12-7 *Motorola* SPI Frame Format (Continuous Transfer) with SPO = 1, SPH = 0

7 Call Processor (CP) Block (continued)

7.12 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

7.12.4.4 *Motorola* SPI Format with SPO = 1, SPH = 1

The transfer signal sequence for *Motorola* SPI format with SPO = 0, SPH = 1 is shown in Figure 7.12-8, which covers both single and continuous transfers. In this configuration, during idle periods, the following occurs:

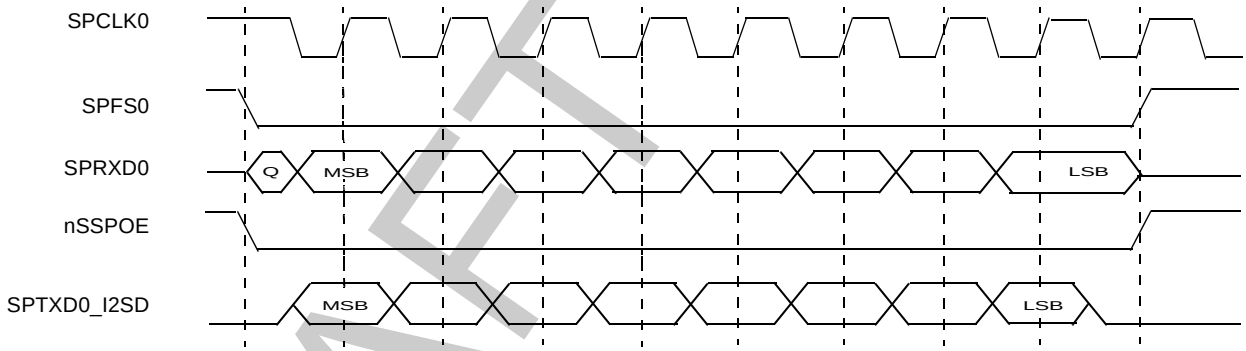
- The SPCLK0 pin is forced high in master mode, or high impedance in slave mode.
- SPFS0 is forced high.
- The transmit data line SPTXD0_I2SD is high impedance.

If the SSPI^{2}S is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SPFS0 master signal being driven low. The nSSPOE line is driven low, enabling the master SPTXD0_I2SD output. After a further one-half SPCLK0 period, both master and slave data are enabled onto their respective transmission lines. At the same time, the SPCLK0 is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SPCLK0 signal.

After all bits have been transferred, in the case of a single word transmission, the SPFS0 line is returned to its idle high state one SPCLK0 period after the last bit has been captured.

For continuous back-to-back transmissions, the SPFS0 signal remains in its active-low state, until the final bit of the last word has been captured, and then returns to its idle state as described above.

For continuous back-to-back transfers, the SPFS0 signal is held low between successive data words and termination is the same as that of the single word transfer.



* Q is an undefined signal.

Figure 7.12-8 *Motorola* SPI Frame Format with SPO = 1, SPH = 1

7 Call Processor (CP) Block (continued)

7.12 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

7.12.5 I^2S

To operate in I^2S mode, set FRF bit field of control register 0 (SSPCR0) to binary 11.

In I^2S mode, the serial interface consists of three pins. The SPCLK0 pin and SPFS0 pin are the clock line and the word select line, respectively. The SPTXD0_I2SD pin is used for time-multiplexed left/right audio data channels, while the word select line SPFS0 also acts as the left/right channel select.

The device that generates the serial clock and word select is the master.

In master mode, SPCLK0 and SPFS0 are forced low, and the transmit data line SPTXD0_I2SD is high impedance whenever the SSPI^{2}S is idle. The idle state of SPCLK0 is utilized by the receiver to provide a receive time-out indication that occurs when the receive FIFO still contains data after a time-out period. Once the transmit FIFO contains some data, SPFS0 is synchronized to the trailing edge of SPCLK0 and the value to be transmitted is shifted from transmit FIFO to the serial shifter. On the next falling edge of SPCLK0, the MSB of the data word is shifted out on SPTXD0_I2SD.

In slave mode, the SPCLK0 input signal generated by external master is double synchronized and then delayed to detect an edge. It takes three *ARM* system clocks to detect an edge on SPCLK0. The MSB of the receiving data is shifted onto SPTXD0_I2SD pin. The receiver latches the data on the rising edge of SPCLK0. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SPCLK0 after the LSB has been latched.

The SSPI^{2}S supports programmable data word size from 4 bits to 16 bits. Varying bit rates can be obtained by programming registers SSPCPSR and SSPCR0. Serial data is transmitted in 2's complement with the MSB first. It isn't necessary for the transmitter to know how many bits the receiver can handle, nor does the receiver need to know how many bits are being transmitted.

The following are recommended programming sequence for I^2S mode:

1. Enable the interrupts (if needed).
2. Write to the various fields of the SSPCR0 register.
3. Write to the various bits in SSPCR0 register, while keeping the SSE bit at 0.
4. Write data to the TxFIFO, if transmitting.
5. Enable SSE bit, to start the operation.

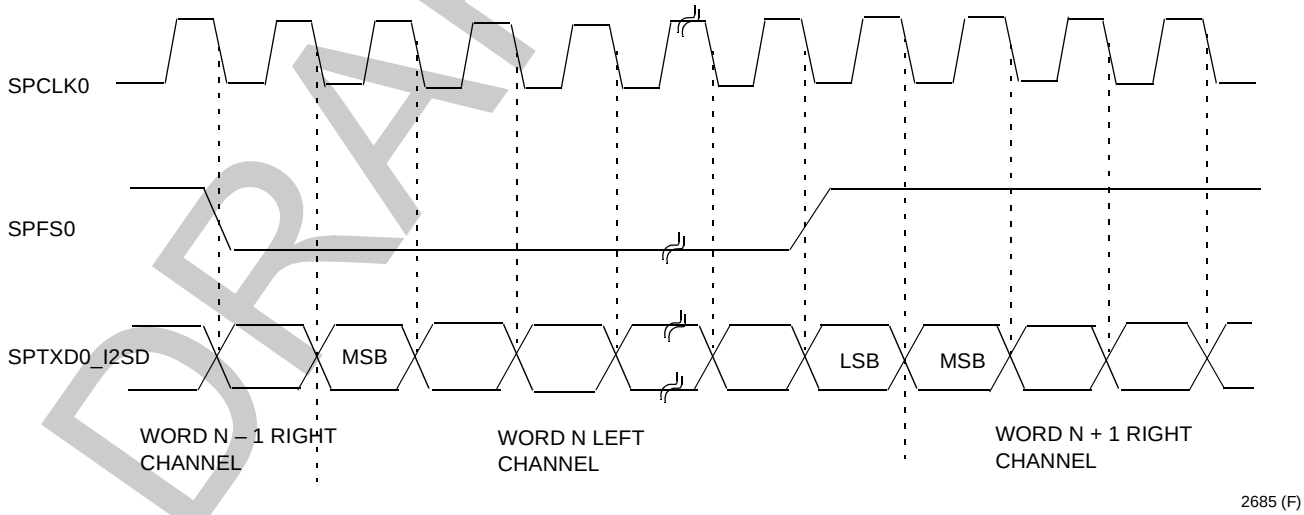


Figure 7.12-9 I^2S Serial Bus Frame Format

7 Call Processor (CP) Block (continued)

7.12 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

7.12.6 Registers

The synchronous serial port (SSP) consists of 9 registers, shown in Table 7.12-3. In this table, SSP_BASE_ADDR = 0x700C3000.

Table 7.12-3 SSP Interface Register Map

Register	Address	Reset Value
Control Register 0 (SSPCR0)	SSP_BASE_ADDR + 0x00	0x0
Control Register 1 (SSPCR0)	SSP_BASE_ADDR + 0x02	0x0
Data Register (SSPDR)	SSP_BASE_ADDR + 0x04	Unknown
Status Register (SSPSR)	SSP_BASE_ADDR + 0x06	0x3
Clock Prescale Register (SSPCPSR)	SSP_BASE_ADDR + 0x08	0x0
Interrupt Mask Set or Clear Register (SSPIMSC)	SSP_BASE_ADDR + 0x0A	0x0
Raw Interrupt Status Register (SSPRIS)	SSP_BASE_ADDR + 0x0C	0x8
Masked Interrupt Status Register (SSPMIS)	SSP_BASE_ADDR + 0x0E	0x0
Interrupt Clear Register (SSPICR)	SSP_BASE_ADDR + 0x10	0x0

7 Call Processor (CP) Block (continued)

7.12 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

7.12.6.1 Control Register 0 (SSPCR0)

SSPCR0 is control register 0 and contains five bit fields that control various functions within the SSPI^{2}S. Table 7.12-4 shows the bit assignments for SSPCR0.

Table 7.12-4 Control Register 0 (SSPCR0), Address (0x700C3000)

Bit	15—8	7	6	5—4	3—0
Name	SCR	SPH	SPO	FRF	DSS

Bit	Name	Type	Function
15—8	SCR	Read/write	Serial clock rate. The value SCR is used to generate the transmit and receive bit rate of the SSPI^{2}S. The bit rate is: $\frac{F_{SSPCLK}}{CPSDVR \times (1 + SCR)}$ where CPSDVSR is an even value from 2 to 254, programmed through the SSPCPSR register, and SCR is a value from 0 to 255. The SSPCLK frequency is equal to half of the *ARM* core frequency.
7	SPH	Read/write	SPCLK output phase (applicable to *Motorola* SPI frame format only).
6	SPO	Read/write	SPCLK output polarity (applicable to *Motorola* SPI frame format only).
5—4	FRF	Read/write	Frame format: 00 *Motorola* SPI frame format. 01 *Texas Instruments* synchronous serial frame format. 10 *National MICROWIRE* frame format. 11 I^2S serial bus format.
3—0	DSS	Read/write	Data size select: 0000 Reserved, undefined operation. 0001 Reserved, undefined operation. 0010 Reserved, undefined operation. 0011 4-bit data. 0100 5-bit data. 0101 6-bit data. 0110 7-bit data. 0111 8-bit data. 1000 9-bit data. 1001 10-bit data. 1010 11-bit data. 1011 12-bit data. 1100 13-bit data. 1101 14-bit data. 1110 15-bit data. 1111 16-bit data.

7 Call Processor (CP) Block (continued)

7.12 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

7.12.6.2 Control Register 1 (SSPCR0)

SSPCR0 is the control register 1 and contains four different bit fields, which control various functions within the SSPI^2S. Table 7.12-5 shows the bit assignments for SSPCR0.

Table 7.12-5 Control Register 1 (SSPCR1), Address (0x700C3004)

Bit	15—10	9	8	7	6	5	4	3	2	1	0
Name	RSVD	nCLKIN	nCLKOUT	DS	I2STX	I2STP	I2SRP	SOD	MS	SSE	LBM

Bit	Name	Type	Function
15—10	RSVD	Read	Reserved. Unpredictable results on reads.
9	nCLKIN	Read/Write	When set to 0, no inversion of SPCLK0 input (default). When set to 1, inverts SPCLK0 input.
8	nCLKOUT	Read/write	When set to 0, no inversion of SPCLK0 output (default). When set to 1, inverts SPCLK0 output.
7	DS	Read/Write	Disable dynamic master/slave switching. When set to 1, dynamic master/slave switching is off. When set to 0, dynamic master/slave switching is on (default).
6	I2STX	Read/write	This bit works in conjunction with the MS bit (2). When set to 1, I^2S is in transit mode. When set to 0, I^2S is in receive mode (default).
5	I2STP	Read/write	This bit applies to the I^2S transmitter in master mode. When set to 0 (default), the word select pin (SPFS pin) is low for all odd numbered transmissions, and high for all even numbered transmissions. When set to 1, the polarity of the word select pin is inverted. Toggling the SSE bit has no effect on the polarity of the word select pin, since the status of this pin is maintained for I^2S mode whether this block is disabled or switched into other serial formats. In order to achieve left/right channel synchronization, the software can track the total number of words transmitted in I^2S mode since reset, and adjust the I2STP bit accordingly. It is recommended that the software make sure an even number of words are written to TX FIFO during each session of I^2S transmission before pausing it or switching to other serial transmission modes.
4	I2SRP	Read/write	During receive, this bit is set to 0 if the left channel word is received first (default). This bit is set to 1 if the first word received is for the right channel. This bit is valid only after the receive FIFO is emptied and then at least one word is received by the SSP block.
3	SOD	Read/write	Slave-mode output disable. This bit is relevant only in the slave mode (MS = 1). In multiple-slave systems, it is possible for an SSP master to broadcast a message to all slaves in the system while ensuring that only one slave drives data onto its serial output line. In such systems, the RXD lines from multiple slaves could be tied together. To operate in such systems, the SOD bit can be set if the SSP slave is not supposed to drive the SSPTXD line. 0 = SSP can drive the SSPTXD output in slave mode (default). 1 = SSP must not drive the SSPTXD output in slave mode.

7 Call Processor (CP) Block (continued)

7.12 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

Table 7.12-5 Control Register 1 (SSPCR1), Address (0x700C3004) (continued)

Bit	Name	Type	Function
2	MS	Read/write	Master or slave mode select. This bit can be modified only when the SSPI^{2}S is disabled (SSE = 0): 0 = Device configured as master (default). 1 = Device configured as slave.
1	SSE	Read/write	Synchronous serial port enable. 0 = SSP operation disabled (default). 1 = SSP operation enabled.
0	LBM	Read/write	Loopback mode. 0 = Normal serial port operation enabled (default). 1 = Output of transmit serial shifter is connected to input of receive serial shifter internally.

7.12.6.3 Data Register (SSPDR)

SSPDR is the data register and is 16 bits wide. When SSPDR is read, the entry in the receive FIFO (pointed to by the current FIFO read pointer) is accessed. As data values are removed by the SSPI^{2}S receive logic from the incoming data frame, they are placed into the entry in the receive FIFO (pointed to by the current FIFO write pointer).

When SSPDR is written to, the entry in the transmit FIFO (pointed to by the write pointer) is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. It is loaded into the transmit serial shifter, and then serially shifted out onto SPTXD0_I2SD at the programmed bit rate.

When the data size of less than 16 bits is selected, the user must right justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 bits is automatically right-justified in the receive buffer.

When the SSPI^{2}S is programmed for *National MICROWIRE* frame format, the default size for transmit data is eight bits (the most significant byte is ignored). The receive data size is controlled by the programmer. The transmit FIFO and the receive FIFO are not cleared even when SSE is set to zero. This allows the software to fill the transmit FIFO before enabling the SSPI^{2}S. Table 7.12-6 shows the bit assignments for SSPDR.

For I^2S, when the system length is greater than the transmitter word length, the word is truncated for data transmission. If the receiver sends more bits than its word length, the bits after the LSB are ignored. If the receiver sends fewer bits than its word length, the missing bits are set to zero internally.

Table 7.12-6 Data Register (SSPDR), Address (0x700C3008)

Bit	15—0
Name	DATA

Bit	Name	Type	Function
15—0	DATA	Read/write	Transmit/receive FIFO: Read = Receive FIFO. Write = Transmit FIFO. Data must be right-justified when the SSPI^{2}S is programmed for a data size that is less than 16 bits. Unused bits at the top are ignored by transmit logic. The receive logic automatically right-justifies.

7 Call Processor (CP) Block (continued)

7.12 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

7.12.6.4 Status Register (SSPSR)

SSPSR is a read-only status register that contains bits that indicate the FIFO fill status and the SSPI^{2}S busy status. Table 7.12-7 shows the bit assignments for SSPSR.

Table 7.12-7 Status Register (SSPSR), Address (0x700C300C)

Bit	15—5	4	3	2	1	0
Name	RSVD	BSY	RFF	RNE	TNF	TFE

Bit	Name	Type	Function
15—5	RSVD	—	Reserved. Unpredictable results on reads, should be written as 0.
4	BSY	Read	SSP busy flag (read-only): 0 = SSP is idle. 1 = SSP is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.
3	RFF	Read	Receive FIFO full (read-only): 0 = Receive FIFO is not full. 1 = Receive FIFO is full.
2	RNE	Read	Receive FIFO not empty (read-only): 0 = Receive FIFO is empty. 1 = Receive FIFO is not empty.
1	TNF	Read	Transmit FIFO not full (read-only): 0 = Transmit FIFO is full. 1 = Transmit FIFO is not full.
0	TFE	Read	Transmit FIFO empty (read-only): 0 = Transmit FIFO is not empty. 1 = Transmit FIFO is empty.

7.12.6.5 Clock Prescale Register (SSPCPSR)

SSPCPSR is the clock prescale register and specifies the division factor by which the SSPCLK (half of *ARM* system clock frequency) must be internally divided before further use. The value programmed into this register must be an even number between 2 to 254. The least significant bit of the programmed number is hardcoded to zero. If an odd number is written to this register, data read back from this register has the least significant bit as zero. Table 7.12-8 shows the bit assignments for SSPCPSR.

Table 7.12-8 Clock Prescale Register (SSPCPSR), Address (0x700C3010)

Bit	15—8	7—0
Name	RSVD	CPSDVSR

Bit	Name	Type	Function
15—8	RSVD	—	Reserved. Unpredictable results on reads, must be written as 0.
7—0	CPSDVSR	Read/write	Clock prescale divisor. Must be an even number from 2 to 254, depending on the frequency of SPCLK0. The least significant bit always returns 0 on reads.

7 Call Processor (CP) Block (continued)

7.12 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

7.12.6.6 Interrupt Mask Set or Clear Register (SSPIMSC)

The SSPIMSC register is the interrupt mask set or clear register. It is a read/write register. On a read, this register gives the current value of the mask on the relevant interrupt. A write of 1 to the particular bit sets the mask, enabling the interrupt to be read. A write of 0 clears the corresponding mask. All the bits are cleared to 0 when reset. Table 7.12-9 shows the bit assignment of the SSPIMSC register.

Table 7.12-9 Interrupt Mask Register (SSPIMSC), Clear/Set Address (0x700C3014)

Bit	15—4	3	2	1	0
Name	RSVD	TxIM	RxIM	RTIM	RORIM

Bit	Name	Type	Function
15—4	RSVD	—	Reserved. Read as zero, do not modify.
3	TxIM	Read/write	Transmit FIFO interrupt mask: 0 = Tx FIFO half empty or less condition interrupt is masked. 1 = Tx FIFO half empty or less condition interrupt is not masked.
2	RxIM	Read/write	Receive FIFO interrupt mask: 0 = Rx FIFO half full or less condition interrupt is masked. 1 = Rx FIFO half full or less condition interrupt is not masked.
1	RTIM	Read/write	Receive time-out interrupt mask: 0 = RxFIFO not empty and no read prior to time-out period interrupt is masked. 1 = RxFIFO not empty and no read prior to time-out period interrupt is not masked.
0	RORIM	Read/write	Receive overrun interrupt mask: 0 = RxFIFO written to while full condition interrupt is masked. 1 = RxFIFO written to while full condition interrupt is not masked.

7.12.6.7 Raw Interrupt Status Register (SSPRIS)

The SSPRIS register is the raw interrupt status register. It is a read-only register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect. Table 7.12-10 shows the bit assignment of the SSPRIS register.

When In I^2S mode, the raw interrupt signals are suppressed in the following way. When in transmit mode, the receive interrupts (RXRIS, RTRIS, and RORIS) are held at 0. When in receive mode the transmit interrupt (TXRIS) is held at 0. If I^2S is in slave receive mode and the master sends in a word of size less than what is programmed in DSS, the RTRIS (receive time-out interrupt) may become active. If there is possibility for a such a situation, then RTIM should be set to 0 masking receive time-out interrupt.

Table 7.12-10 Raw Interrupt Status Register (SSPRIS), Address (0x700C3018)

Bit	15—4	3	2	1	0
Name	RSVD	TxRIS	RxRIS	RTRIS	RORRIS

Bit	Name	Type	Function
15—4	RSVD	—	Reserved. Read as zero, do not modify.
3	TxRIS	Read	Gives the raw interrupt state (prior to masking) of the SSPTXINTR interrupt.
2	RxRIS	Read	Gives the raw interrupt state (prior to masking) of the SSPRXINTR interrupt.
1	RTRIS	Read	Gives the raw interrupt state (prior to masking) of the SSPRTINTR interrupt.
0	RORRIS	Read	Gives the raw interrupt state (prior to masking) of the SSPRORINTR interrupt.

7 Call Processor (CP) Block (continued)

7.12 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

7.12.6.8 Masked Interrupt Status Register (SSPMIS)

The SSPMIS register is the masked interrupt status register. It is a read-only register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect. Table 7.12-11 shows the bit assignment of the SSPMIS register.

Table 7.12-11 Masked Interrupt Status Register (SSPMIS), Address (0x700C301C)

Bit	15—4	3	2	1	0
Name	RSVD	TxMIS	RxMIS	RTMIS	RORMIS

Bit	Name	Type	Function
15—4	RSVD	—	Reserved. Read as zero, do not modify.
3	TxMIS	Read	Gives the transmit FIFO masked interrupt state (after masking) of the SSPTXINTR interrupt.
2	RxMIS	Read	Gives the receive FIFO masked interrupt state (after masking) of the SSPRXINTR interrupt.
1	RTMIS	Read	Gives the receive time-out masked interrupt state (after masking) of the SSPRTINTR interrupt.
0	RORMIS	Read	Gives the receive over run masked interrupt status (after masking) of the SSPRORINTR interrupt.

7.12.6.9 Interrupt Clear Register (SSPICR)

The SSPICR register is the interrupt clear register and is write-only. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect. Table 7.12-12 shows the bit assignment of the SSPICR register.

Table 7.12-12 Interrupt Clear Register (SSPICR), Address (0x700C3020)

Bit	15—2	1	0
Name	RSVD	RTIC	RORIC

Bit	Name	Type	Function
15—2	RSVD	—	Reserved. Read as zero, do not modify.
1	RTIC	Write	Clears the SSPRTINTR interrupt.
0	RORIC	Write	Clears the SSPRORINTR interrupt.

7 Call Processor (CP) Block (continued)

7.13 Subscriber Identity Module (SIM) Interface

The subscriber identity module (SIM) interface supports communication between a GSM cellular phone (mobile equipment—ME) and a plug-in SIM card. The SIM interface conforms to GSM-TS 11.11. A set of control registers allows the operating modes of the interface to be configured under software control. The following is a list of features in the SIM interface:

- 4 bytes of FIFO for both receive and transmit.
- Single interrupt routed to the programmable interrupt controller.
- Programmable baud rate derived from the call processor clock.
- Complete status reporting.
- Support for DMA transfers.
- SIM interface:
  - Asynchronous half-duplex communication conforming to GSM-TS 11.11.
  - Support for enhanced-speed SIM cards.
  - One start bit, 8 data bits, 1 optional parity bit, and 2 stop bits.
  - Open-drain transmit pin.
  - Sample clock 16, 32, 64, 96, or 372 times baud rate.
  - Receive and transmit data is 8 bits with LSB or MSB first.
  - Automatic retransmit of last character when error detected (up to 2 times).
  - Extra guard time is provided as an option.

7.13.1 Operation

As shown in Figure 7.13-1, the function of the SIM interface is to convert incoming serial data on the Rx line to parallel data and convert parallel data from the CPU to serial data on the Tx line. The outgoing/incoming serial data can be programmed to be transmitted/received at different baud rates by programming corresponding values in the baud rate register. The programmable baud rate generator can divide the incoming clock by 1 or 4 to 131,072. The output of the baud rate generator can be configured to be 16, 32, 64, 96, or 372 times the serial data baud rate. To conform to GSM-TS 11.11 powerup requirements, the SIMTx output freezes at low. The SIM interface must be programmed to unfreeze SIMTx output by setting bit 6 of the transmitter control register to 1.

A single interrupt line is used to generate an interrupt to the CPU. The interrupt type can be read from the SIM status register.

7 Call Processor (CP) Block (continued)

7.13 Subscriber Identity Module (SIM) Interface (continued)

7.13.2 SIM Mode Operation

SIM mode is used to interface to a standard SIM card connected to a GSM cellular phone. SIM mode configures the SIM interface for half-duplex communications with an open-drain transmit output, and selects either the 16 times-sample clock or 372 times-sample clock. SIM mode also enables the transmit error detection and retransmission logic. The transmitter and receiver parity logic must be enabled separately and should be set to even parity when communicating with a SIM card.

Before starting a receive or a transmit operation on the SIM interface, a RESET signal can be generated from the programmable peripheral interface. In order to transmit data on the SIMTx line, program an appropriate divisor in the baud rate register, set the mode control register, set the transmitter control register, and then write data in the transmitter FIFO. At this point, the transmitter waits for a pending receive operation to finish. If there is a valid start bit at the same time data is first available from the transmitter FIFO, the receiver takes precedence and receives the character. Pending receive operations are finished, the data in the transmitter FIFO is transferred to the transmitter shift register, a start bit is generated, and the data is shifted to the output one bit at a time at the rate programmed. The data transmitted is synchronized to the baud rate generator so the width of the start bit does not vary. The parity bit, if enabled, is generated and shifted out after the 8 data bits. After the transmitter shift register completes the shift operation, the transmitter waits for 2 stop bits after the parity bit to detect any error. If the SIMRx line is low at the end of the first stop bit, due to a parity error encountered by the receiver, the data is retransmitted and the SIM retransmit flag is set. When an error is detected, a third stop bit is inserted before the next transmission, regardless of whether the extra stop bit (guard time) option is activated or not. The data is retransmitted for a total of two times if the error repeats, and on the third error the parity error flag is set, that can generate an interrupt if the interrupt is enabled. If the extra stop bit option is activated, error checking will be performed at the end of every stop bit except the last stop bit. If a character was transmitted/retransmitted and no error was detected by the receiver, the next byte in the transmitter FIFO is loaded into the transmitter shift register. When the number of characters remaining in the transmitter FIFO is below the transmitter FIFO threshold programmed in the transmitter control register, the transmitter FIFO threshold bit is set in the SIM status register. An interrupt can also be generated on this condition if the interrupt is enabled in the transmitter control register.

To receive serial data from the SIMRx pin, program an appropriate divisor in the baud rate register, set the mode control register, and set the receiver control register. The receiver looks for a start bit only when the transmitter is not currently transmitting. If the transmitter is transmitting data, the receiver waits until the transmitter is done, including waiting for the stop bits inserted by the transmitter. When the transmitter is done, the receiver immediately begins to look for a start bit. After a start bit is detected, the data on the SIMRx line is shifted into the receiver shift register. This is done by delaying one-half bit time and then sampling each data bit in the center of its ideal bit time. There can be some variance to when data is sampled because the state of the baud rate counter can vary from character to character. The variance introduced is no greater than 1/16 of a bit time. After shifting one character and the optional parity bit into the receiver shift register, the data is tested for a parity error. If a parity error is detected, a low signal is asserted by the receiver on the SIMTx line halfway through the first stop bit.

If a parity error is not detected, the data is transferred to the receiver FIFO. When the number of characters in the receiver FIFO exceeds the receiver FIFO threshold programmed in the receiver control register, the receiver FIFO threshold bit in the SIM status register is set. An interrupt can also be generated on this condition if the interrupt is enabled in the receiver control register. Only one stop bit is checked when receiving, but the transmitter cannot start transmitting until after the time for two full stop bits has elapsed.

If the MSB first is set, data bits are inverted (i.e., data bits are transmitted/received as active-low signals). If LSB first is set, data bits are not inverted (i.e., data bits are transmitted/received as active-high signals).

The SIM interface clock output is active while in SIM mode. When in SIM mode, the clock output is a 50% duty cycle signal. The frequency of the SIM interface clock output corresponds to the rate programmed for the baud rate generator.

The SIM interface clock output can be stopped optionally when the SIM interface is shut down using the system configuration register described in Table 7.13-9. Also, the SIM interface clock output can be shut down and left in either a high or low state.

To conform to GSM-TS 11.11 powerup requirements, the SIMTx output freezes at low. The SIM interface must be programmed to unfreeze SIMTx output by setting bit 6 of the transmitter control register to 1.

7 Call Processor (CP) Block (continued)

7.13 Subscriber Identity Module (SIM) Interface (continued)

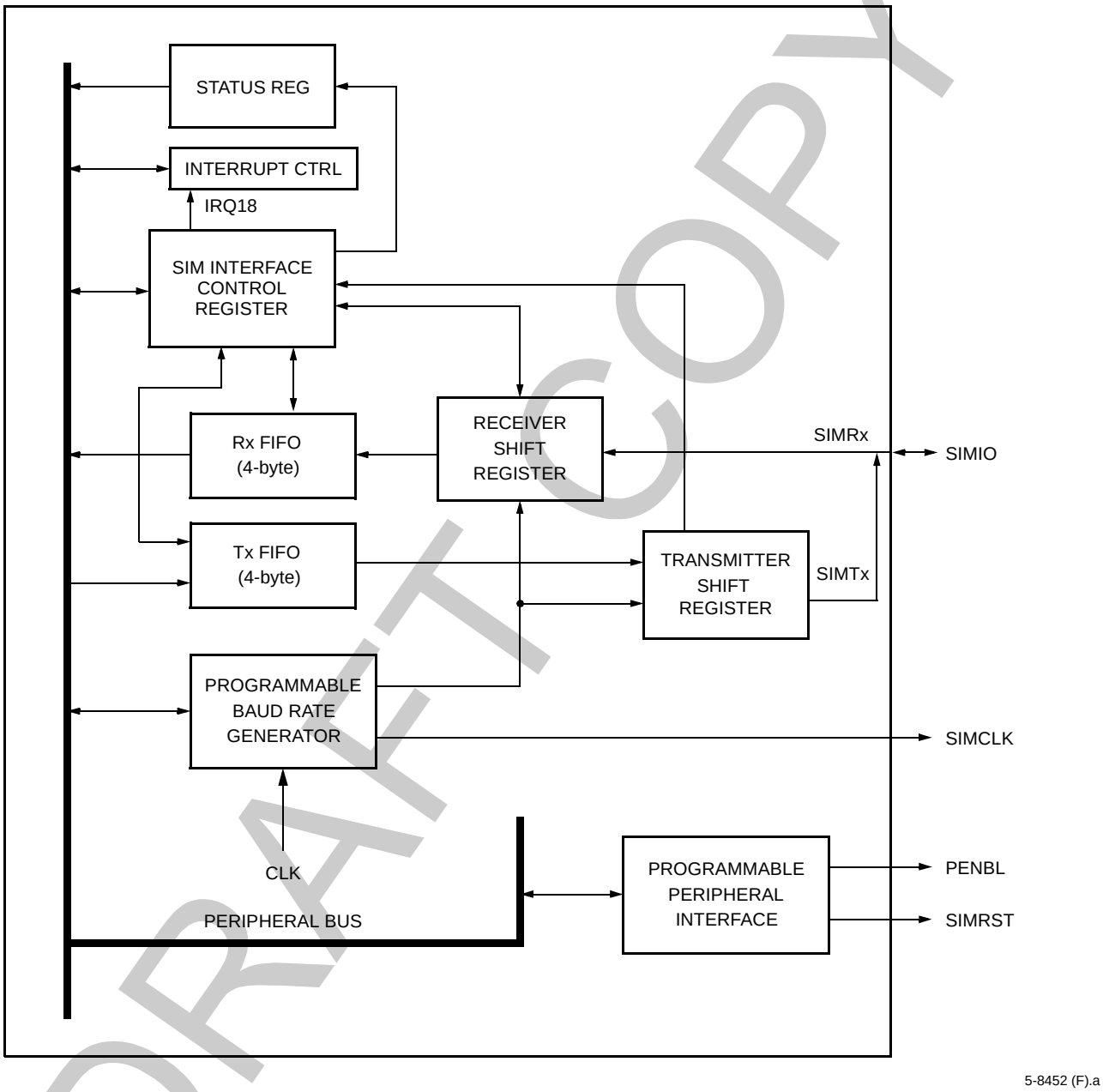


Figure 7.13-1 SIM Interface Block Diagram

7 Call Processor (CP) Block (continued)

7.13 Subscriber Identity Module (SIM) Interface (continued)

7.13.3 Registers

7.13.3.1 Baud Rate Register (SIMBRR)

The baud rate register is used to divide the system clock to generate different baud rates. The SIM baud rate generator is 16 bits wide; hence division factors of 1—65,536 can be programmed, with an additional divide by 2 if the baud rate register is not zero. The output of the baud rate generator is treated as either 16 or 372 times (as selected by the mode control register) the required baud rate when in SIM mode.

Table 7.13-1 Baud Rate Register, Address (0x700CB000)

Bit	31—16	15—0
Name	RSVD	Baud Rate Divisor

Bit	Name	Description
31—16	RSVD	Reserved.
15—0	Baud Rate Divisor	Bits[15:0] specify the baud rate divisor. The divisor is 1 for a value of all zeros in bits[15:0] and 131,072 for a value of all ones in bits[15:0].

The following equation gives the baud clock divisor value for a given baud rate:

CLK: System clock in MHz.

BR: Baud rate in bits/s.

BCD16: Baud clock division factor for a sampling divisor of 16.

BCD372: Baud clock division factor for a sampling divisor of 372.

$BCD16 = (CLK \times 10^6)/(16 \times BR)$.

$BCD372 = (CLK \times 10^6)/(372 \times BR)$.

The divisor of 372 is normally used when the SIM interface is providing the clock to a SIM card, per GSM-TS 11.11. The other possible divisors (32, 64, 96) are available to support enhanced speed SIM cards.

For example, to obtain a baud rate of 9600, using a system clock of 13 MHz, BCD372 = 3.64. Since the baud rate register can be programmed using only an integer value and there is an additional divide by two when in SIM mode, the value of BCD372 should be the nearest even integer value, which is 4. Hence, using a BCD372 of 4, the baud rate obtained is 8737, which is 8.99% lower than 9600. The 8.99% variation would only change the data rate and would not introduce an error because a common clock is used by the SIM interface and the SIM card with which it is communicating.

The value to be written to the baud rate register should be ((BCD[16|32|64|96|372]/2) – 1). The only exception is that a baud rate register value of all zeros corresponds to a BCD[16|32|64|96|372] of 1.

7 Call Processor (CP) Block (continued)

7.13 Subscriber Identity Module (SIM) Interface (continued)

7.13.3.2 Baud Rate Counter (SIMBRC)

The baud rate counter is a read-only register that returns the current value of the baud rate counter used to generate the required baud rate. The baud rate counter is a 16-bit down counter that decrements by 1 every clock cycle (the clock used operates at the same frequency as CKI, the system clock input). This counter is initialized with the value in the baud rate register after the counter counts down to 0, or if the baud rate register is written.

Table 7.13-2 Baud Rate Counter (SIMBRC), Address (0x700CB004)

Bit	31—16	15—0
Name	RSVD	Baud Rate Counter

Bit	Name	Description
31—16	RSVD	Reserved.
15—0	Baud Rate Counter	Bits[15:0] are the current value of the baud rate counter.

7.13.3.3 FIFO Status Register (SIMFIFOS)

The FIFO status register is used to inform the CPU of the status of the transmitter and receiver FIFOs. The FIFO status register is a read-only register. Writes to its address are ignored. Table 7.13-3 depicts the format of the FIFO status register.

Table 7.13-3 FIFO Status Register (SIMFIFOS), Address (0x700CB008)

Bit	31—6	5	4	3	2	1	0
Name	RSVD	Tx FIFO Full	Tx FIFO Half Full	Tx FIFO Empty	Rx FIFO Full	Rx FIFO Half Full	Rx FIFO Empty

Bit	Name	Description
31—6	RSVD	Reserved.
5	Tx FIFO Full	If 1, the transmitter FIFO is full. If 0, the transmitter FIFO is not full.
4	Tx FIFO Half Full	If 1, the transmitter FIFO is at least half full. If 0, the transmitter FIFO is less than half full.
3	Tx FIFO Empty	If 1, the transmitter FIFO is empty. If 0, the transmitter FIFO is not empty, and the transmitter ready signal to the DMA controller is asserted.
2	Rx FIFO Full	If 1, the receiver FIFO is full. If 0, the receiver FIFO is not full.
1	Rx FIFO Half Full	If 1, receiver FIFO is at least half full. If 0, receiver FIFO is less than half full.
0	Rx FIFO Empty	If 1, the receiver FIFO is empty. If 0, information is read into the register, and receiver ready signal to the DMA controller is asserted.

7 Call Processor (CP) Block (continued)

7.13 Subscriber Identity Module (SIM) Interface (continued)

7.13.3.4 SIM Status Register (SIMS)

The SIM status register is used to inform the CPU of the status of the SIM interface. The SIM status register is a read-only register. Writes to its address are ignored. Table 7.13-4 depicts the format of the SIM status register.

Table 7.13-4 SIM Status Register (SIMS), Address (0x700CB00C)

Bit	31—8	7	6	5	4	3	2	1	0
Name	RSVD	Retransmit Performed	SIM Receive Parity Error	Tx Parity Error	Tx FIFO Threshold	Rx Framing Error	Rx Overrun Error	RSVD	Rx FIFO Threshold

Bit	Name	Description
31—8	RSVD	Reserved.
7	Retransmit Performed	Bit 7 is used to indicate that a transmitter parity error has occurred and that a character has been retransmitted. If 1, a retransmit has been completed. This bit is reset when the SIM status register is read.
6	SIM Receive Parity Error	If 1, a parity error has been detected in a received character. This bit is reset when the SIM status register is read.
5	Tx Parity Error	If 1, a transmitter parity error has occurred. This bit is only set after a character has been retransmitted 2 times with a parity error each time. This bit is reset when the SIM status register is read.
4	Tx FIFO Threshold	Bit 4 is the transmitter FIFO threshold event indicator. If 1, the transmitter FIFO threshold condition is met. If 0, the transmitter FIFO condition is not met, due to a FIFO write or control change.
3	Rx Framing Error	If bit 3 is 1, a framing error has occurred, i.e., the received character did not have a valid stop bit. This bit is reset when the SIM status register is read.
2	Rx Overrun Error	Bit 2 is the overrun indicator. If bit 2 is 1, a character was received at a time when the receiver FIFO was full. This bit is reset when the SIM status register is read.
1	RSVD	Reserved.
0	Rx FIFO Threshold	Bit 0 is the receiver FIFO threshold event indicator. If 1, receiver FIFO threshold condition is met. If 0, the FIFO condition is not met, due to a FIFO read or control change.

7 Call Processor (CP) Block (continued)

7.13 Subscriber Identity Module (SIM) Interface (continued)

7.13.3.5 Receiver Control Register (SIMRXC)

The receiver control register is used to control the receiver FIFO, interrupts, and parity generation. On reset, the receiver control register is set to all zeros. Table 7.13-5 depicts the receiver control register layout.

Table 7.13-5 Receiver Control Register (SIMRXC), Address (0x700CB010)

Bit	31—6	5	4—3	2—1	0
Name	RSVD	Rx Error Interrupt Enable	Parity Control	FIFO Interrupt Control Enable	FIFO Reset

Bit	Name	Description
31—6	RSVD	Reserved.
5	Rx Error Interrupt Enable	Bit 5 is used to enable receiver error interrupts (parity, frame, and overrun). If 0, disables receiver error interrupts. If 1, receiver error interrupts are enabled. This bit is set to 0 upon reset.
4—3	Parity Control	Bits[4:3] are used to control receiver parity checking. Table 7.13-7 depicts the encoding for this field. Parity checking is disabled upon reset.
2—1	FIFO Interrupt Control Enable	Bits[2:1] are used to control the receiver FIFO interrupt. Table 7.13-6 depicts the encoding for this field. Receiver FIFO interrupts are disabled upon reset.
0	FIFO Reset	Bit 0 is used to reset the receiver FIFO. If 1, this bit resets the receiver FIFO, discarding any data still there and marking it empty. Bit 0 must be written to 0 before the FIFO can accept new data; the receiver FIFO is reset upon reset to the SIM interface.

Table 7.13-6 Receiver FIFO Threshold Interrupt Control Encoding

Bits[2:1]	FIFO Threshold Interrupt Control
0 0	FIFO threshold interrupts disabled.
0 1	Generates an interrupt when the receiver FIFO is not empty.
1 0	Generates an interrupt when the receiver FIFO is at least half full.
1 1	Generates an interrupt when the receiver FIFO is full.

Table 7.13-7 Parity Control Encoding

Bits[4:3]	Parity
0 0	No parity.
0 1	Mark parity (always send a 1).
1 0	Even parity.
1 1	Odd parity.

7 Call Processor (CP) Block (continued)

7.13 Subscriber Identity Module (SIM) Interface (continued)

7.13.3.6 Transmitter Control Register (SIMTXC)

The transmitter control register is used to control the transmitter FIFO, interrupts, and parity generation. On any reset, the transmitter control register is set to all zeros. Table 7.13-8 depicts the transmitter control register.

Table 7.13-8 Transmitter Control Register (SIMTXC), Address (0x700CB014)

Bit	31—9	8—7	6	5	4—3	2—1	0
Name	RSVD	Guard Time	Freeze SIMTx	Tx Parity Error Interrupt Enable	Parity Control	FIFO Interrupt Control Enable	FIFO Reset

Bit	Name	Description
31—9	RSVD	Reserved.
8—7	Guard Time	Bits[8:7] are used to control the number of stop bits. See Table 7.13-10.
6	Freeze SIMTx	Bit 6 is used to control the SIMTx output. If 0, SIMTx is driven low. If 1, the SIMTx is set to a high-impedance state.
5	Tx Parity Error Interrupt Enable	Bit 5 is used to enable transmitter parity error interrupts when in SIM mode. If 0, transmitter parity error interrupts are disabled. If 1, transmitter parity error interrupts are enabled. This bit is set to 0 upon reset.
4—3	Parity Control	Bits[4:3] are used to control transmitter parity generation. Parity generation is disabled upon reset. Table 7.13-7 depicts the encoding for this field.
2—1	FIFO Interrupt Control Enable	Bits[2:1] are used to control the transmitter FIFO interrupt. Transmitter FIFO interrupts are disabled upon reset. Table 7.13-9 depicts the encoding for this field.
0	FIFO Reset	Bit 0 is used to reset the transmitter FIFO. If 1, this bit resets the transmitter FIFO, discarding any data still there and marking it empty. If 0, the FIFO can accept new data. The transmitter FIFO is reset upon any reset to the SIM interface.

Table 7.13-9 Transmitter FIFO Threshold Interrupt Control Encoding

Bits[2:1]	FIFO Threshold Interrupt Control
0 0	FIFO threshold interrupts disabled.
0 1	Generates an interrupt when the transmitter FIFO is not full.
1 0	Generates an interrupt when the transmitter FIFO is less than half full.
1 1	Generates an interrupt when the transmitter FIFO is empty.

Table 7.13-10 Guard Time Control Encoding

Bits[8:7]	Extra Guard Time Control
0 0	2 stop bits.
0 1	3 stop bits.
1 0	4 stop bits.
1 1	5 stop bits.

7 Call Processor (CP) Block (continued)

7.13 Subscriber Identity Module (SIM) Interface (continued)

7.13.3.7 Mode Control Register (SIMMODEC)

The mode control register is used to select the SIM mode options. On any reset, the mode control register is set to all zeros. Table 7.13-11 depicts the mode control register.

Table 7.13-11 Mode Control Register (SIMMODEC), Address (0x700CB018)

Bit	31—8	7—6	5	4	3	2	1	0
Name	RSVD	Extended Sample Clock Selection	SIMCLKOE	Sample Clock Selection	LSM/MSB First	SIMCLK Stop High	SIMCLK Enable	RSVD

Bit	Name	Description
31—8	RSVD	Reserved.
7—6	Extended Sample Clock Selection	Bits[7:6] are used to select the input sample clock when bit 4 is 1 and bit 0 is 0. Table 7.13-12 depicts the encoding of bits[7:6].
5	SIMCLKOE	SIM clock output enable. If 0: high impedance; If 1: SIMCLK output is enabled.
4	Sample Clock Selection	Bit 4 is used, in conjunction with bits[7:6], to select the input sample clock. If 0, the divided by 16 sample clock is selected. If 1, the sample clock is specified by bits[7:6].
3	LSM/MSB First	Bit 3 is used to set up SIM interface to transmit/receive MSB or LSB of the character first. If 1, MSB is received/transmitted first and LSB last. If 0, LSB is transmitted/received first and MSB last. When the SIM interface is set up to transmit/receive MSB first, the data bits are inverted, i.e., data bits are active-low. Parity generation and checking is based on the bit values actually transmitted and received. When the SIM interface is set up to transmit/receive LSB first, the data bits are noninverted, i.e., the data bits are active-high.
2	SIMCLK Stop High	Bit 2 is used to specify the state of the SIMCLK output when it is disabled by setting bit 1 to a 0. If bit 2 is 1, the SIMCLK output is held high when disabled. If bit 2 is 0, the SIMCLK output is held low when disabled.
1	SIMCLK Enable	Bit 1 is used to control the SIMCLK output. If 1, the SIMCLK output's clock remains active. If 0, it is inactive and held at the level programmed in bit 2.
0	RSVD	Reserved. Must be set to 0.

Table 7.13-12 Extended Sample Clock Selection Encoding in Bits[7:6]

Bits[7:6]	Sample Clock
00	372
01	32
10	64
11	96

7 Call Processor (CP) Block (continued)

7.13 Subscriber Identity Module (SIM) Interface (continued)

7.13.3.8 Tx/Rx FIFO Register (SIMFIFO)

The Tx/Rx FIFO register provides access to the transmitter and receiver FIFOs. A write to this register writes a character to the transmitter FIFO. A read from this register reads a character from the receiver FIFO. Both FIFOs are reset upon any reset to the SIM interface. The contents of the FIFOs are indeterminate upon reset.

Both FIFOs provide status information for the FIFO status register and the SIM status register. This information is also used to generate the transmitter and receiver FIFO threshold interrupts.

The FIFOs do not store the characters currently being transmitted from the transmitter shift register or received in the receiver shift register.

A read from an empty Rx FIFO returns the byte from the FIFO position just after the last Rx FIFO read, but it does not change the status of the Rx FIFO. A write to a full Tx FIFO is ignored.

Table 7.13-13 Tx/Rx FIFO Register (SIMFIFO), Address (0x700CB01C)

Bit	31—8	7—0
Name	RSVD	Character

Bit	Name	Description
31—8	RSVD	Reserved.
7—0	Character	Character to transmit when written to. Character received when read from.

7.13.4 DMA Support

The SIM interface provides two ready signals to the DMA controller, one for transmit and the other for receive. The transmit-ready signal is asserted when the transmit FIFO is empty. The receive ready signal is asserted when the receive FIFO has at least one valid character in it (it is not empty). The DMA controller must be programmed to use the required ready signals when it is set up.

7.13.5 Operation on Reset

Upon any reset, the SIM interface performs the following:

- All ongoing transfers are aborted.
- Both transmitter and receiver FIFOs are reset (the contents of the FIFOs are indeterminate upon any reset).
- The mode control register is reset to all zeros, suspend the SIMCLK output in a low state, select LSB noninverted transmission, and select the 16 times-sample clock. See the second note for Figure 7.13-4 for the SIMTx output.
- The transmitter control register is reset to all zeros to disable transmitter FIFO interrupts, disable transmitter parity generation, and disable transmitter parity error interrupts.
- The receiver control register is reset to all zeros to disable receiver FIFO interrupts, disable receiver parity checking, and disable receiver error interrupts.
- The SIM status register is reset to all zeros.
- The FIFO status register is set to reflect the current status of both transmitter and receiver FIFOs (empty).
- The baud rate register is reset to all zeros.

7 Call Processor (CP) Block (continued)

7.13 Subscriber Identity Module (SIM) Interface (continued)

7.13.6 External Interface

The external interface of the SIM interface supports connection to a SIM card and to standard serial interface drivers/receivers (e.g., RS-232C, RS-422). A start bit is transmitted using a low output signal, while a stop bit is transmitted using a high signal. Figure 7.13-2 depicts a timing diagram when LSB first is selected while in SIM mode. Figure 7.13-3 depicts a timing diagram when MSB first is selected while in SIM mode. Note that the MSB first waveform uses inverted data.

Interfacing to a SIM card requires an open-drain transmit output that is provided. Figure 7.13-4 depicts a direct connection from a SIM card to T8307.

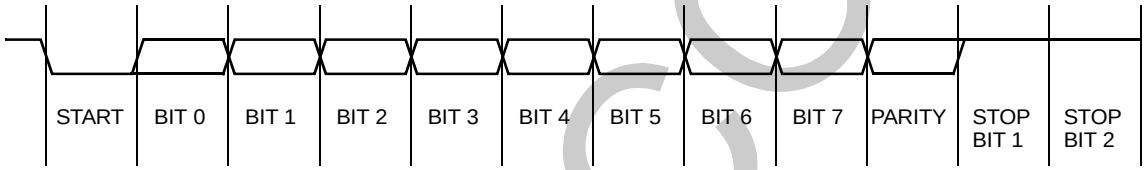


Figure 7.13-2 SIM Least Significant Bit First Timing Diagram

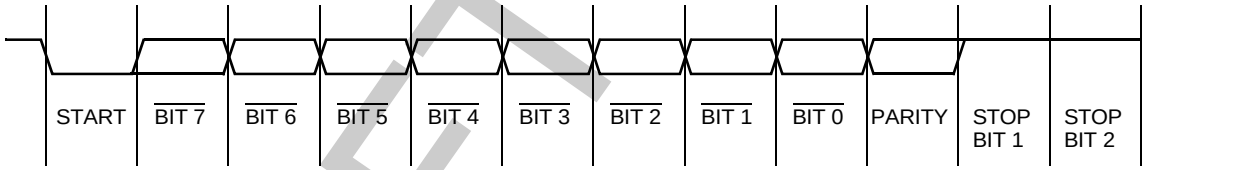
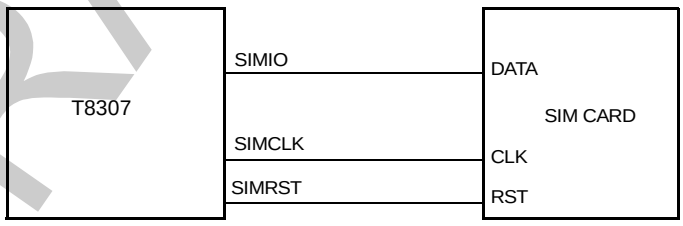


Figure 7.13-3 SIM Most Significant Bit First Timing Diagram



Notes:
For RESET, a programmable peripheral interface port pin can be used.

Upon reset, the SIMTx output freezes at low to conform to GSM-TS 11.11 powerup requirements; it is recommended to set bit 6 to 1 in the transmitter control register in the boot code. This may avoid power drain through the pull-up register.

Figure 7.13-4 SIM Card Connection

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC)

USB device controller (USBDC) is an integrated Agere USS820 core for data transfers between mobile terminal applications (device function) and a personal computer (the host). USBDC supports the following features:

- USB specification V1.1 compliant.
- Full-speed (12 Mbits/s) device operation.
- 16 unidirectional endpoints. Each endpoint capable of supporting control, interrupt, isochronous and bulk transfer.
- Support memory-to-memory DMA transfers.
- Supports external USB 1.1 transceivers with 1.8 V digital I/O interface such as Agere's USS-810*, Philips* ISP1105, 1106, 1107, and *Micrel* MIC2551.
- Programmable endpoint types and FIFO sizes and internal 1120-byte logical (2240-byte physical for dual-packet mode) shared FIFO storage allow a wide variety of configurations.

USB packet belongs to one of the following categories: token, SOF, data, handshake, and special.

Token packet is used for USB transaction initiation.

SYNC	PID	ADDR	ENDP	CRC

SYNC — 8-bit, to align incoming data with local clock.

PID — packet identifier, 4 bit + 4 bit check field to indicate type of packet transmitted.

ADDR — 7-bit function address to specify which device the packet is for.

ENDP — 4-bit endpoint specifies the tiny device pipe this packet is for.

CRC — error checking for non-PID fields.

Start of frame (SOF) packets are broadcast by the host once every 1.00 ± 0.05 ms.

SYNC	PID	Frame Number	CRC

Frame number — 11-bit frame number.

Data packet is used for data transaction.

SYNC	PID	DATA	CRC

DATA — data portion for USB communication.

Handshake packets are used to report ACK, NACK, or STALL indicated by PID.

SYNC	PID

Special packets includes the special preamble (PRE) packets used for 1.5 Mbits/s low speed data. The host will send this PRE first before communication. The special packet is not supported in T8307 USBDC.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

7.14.1 Connection of USB Transceiver to T8307

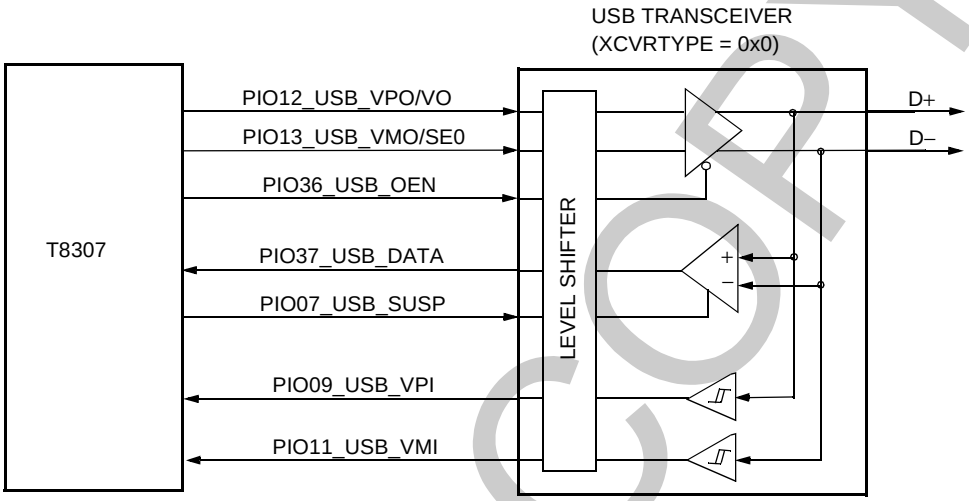


Figure 7.14-1 Connection to Single-Ended Type Transceiver, Agere USS810 (FSE0 = H)

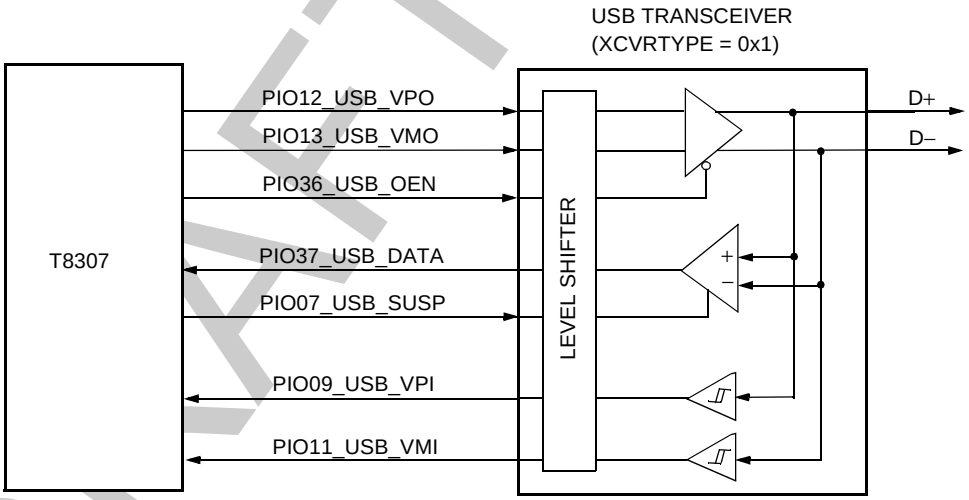


Figure 7.14-2 Connection to Differential Type Transceiver, *Philips* Agere USS810 (FSE0 = L)

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

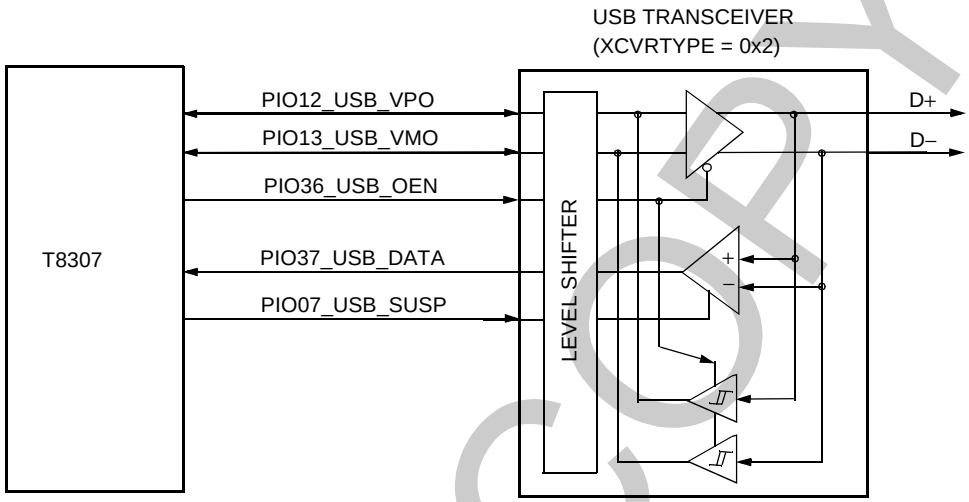


Figure 7.14-3 Connection to Bidirectional Differential Type Transceiver, *Micrel* MIC2551

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

7.14.2 USB Controller Functional Description

Figure 7.14-4 shows the top-level diagram of the USB controller block.

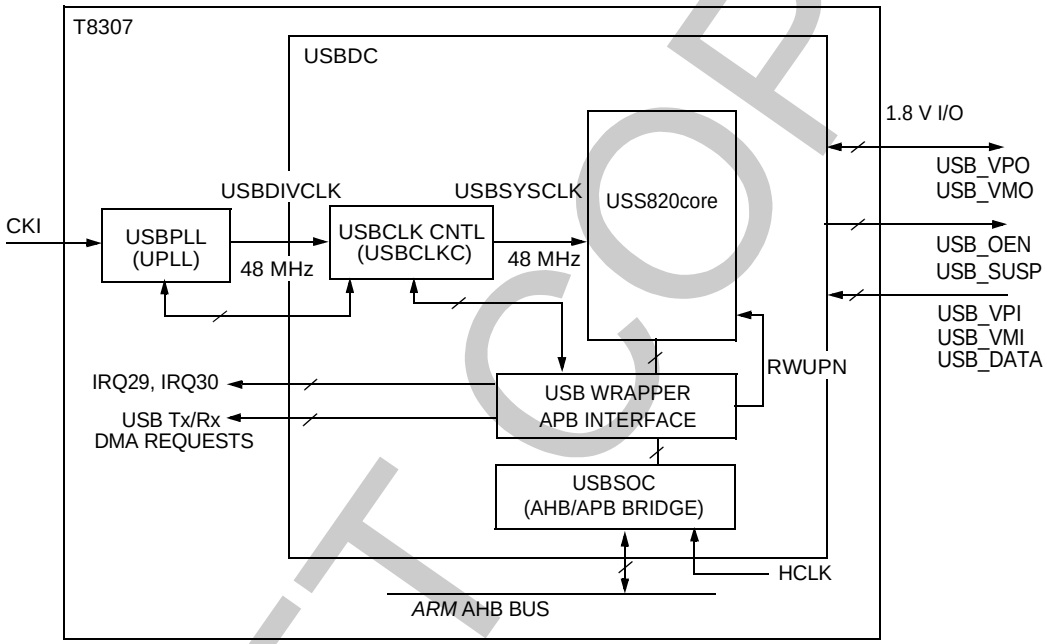


Figure 7.14-4 Block Diagram of USB Controller

USS820core is a USB device controller that provides a programmable bridge between the USB and the local microprocessor bus. It is programmable through a simple read/write register interface.

USS820core FIFO options support all four transfer types: control, interrupt, bulk, and isochronous, as described in *Universal Serial Bus Specification Revision 1.1,* with a wide range of packet sizes. Its double sets of FIFO enable the dual-packet mode feature. The dual-packet mode feature reduces latency by allowing simultaneous transfers on the host and microprocessor sides of a given unidirectional endpoint.

The USS820core supports a maximum of eight bidirectional endpoints with 16 FIFOs (eight for transmit and eight for receive) associated with them. The FIFOs are on-chip, and sizes are programmable up to a total of 1120 logical bytes. When the dual-packet mode feature is enabled, the device uses a maximum of 2240 bytes of physical storage. This additional physical FIFO storage is managed by the device hardware and is transparent to the user.

The FIFO sizes supported are 8 bytes, 16 bytes, 32 bytes, and 64 bytes for nonisochronous pipes, and 64 bytes, 256 bytes, 512 bytes, and 1024 bytes for isochronous pipes. The FIFO size of a given endpoint defines the upper limit to maximum packet size that the hardware can support for that endpoint. This flexibility covers a wide range of data rates, data types, and combinations of applications.

The USS820core is clocked by a 48 MHz clock generated by an on-chip USB PLL. The internal 12 MHz clock period, which is derived from the 48 MHz clock, is referred to as the device clock period (tCLK) in the following sections.

Figure 7.14-5 shows the functional diagram of the USS820core block.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

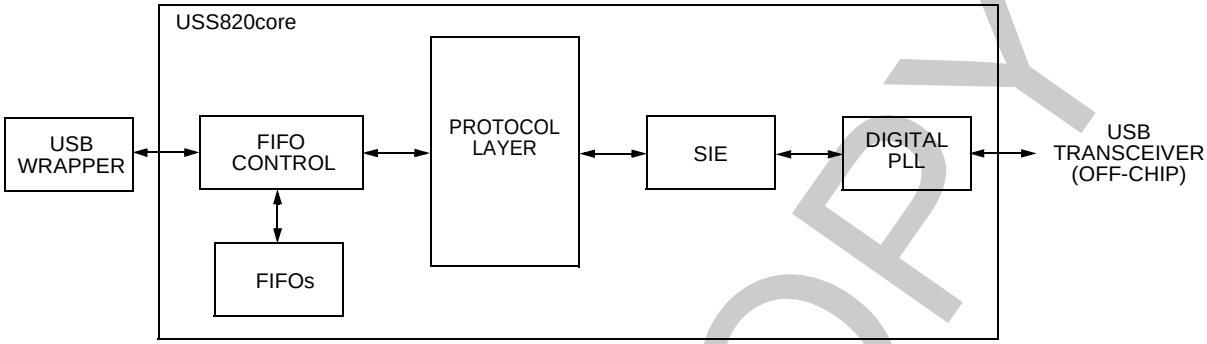


Figure 7.14-5 Block Diagram of USS820core

7.14.2.1 Serial Interface Engine

The SIE is the USB protocol interpreter. It serves as a communicator between the device-side USS820core and the USB host.

The SIE functions include the following:

- Package protocol sequencing.
- SOP (start of packet), EOP (end of packet), RESUME, and RESET signal detection and generation.
- NRZI data encoding/decoding and bit stuffing.
- CRC generation and checking for token and data.
- Serial-to-parallel and parallel-to-serial data conversion.

7.14.2.2 Protocol Layer

The protocol layer manages the interface between the SIE and FIFO control blocks. It passes all USB OUT and SETUP packets through to the appropriate FIFO. It is the responsibility of firmware to correctly interpret and execute each USB SETUP command (as documented in Section 7.14.6) via the register interface. The protocol layer tracks the setup, data, and status stages of control transfers.

7.14.2.3 FIFO Control

USS820core's FIFO control manager handles the data flow between the FIFOs and the device controller's protocol layer. It handles flow control and error handling/fault recovery to monitor transaction status and to relay control events via interrupt vectors.

7.14.2.4 FIFO Programmability

Table 7.14-1 shows the programmable FIFO sizes. The size of the FIFO determines the maximum packet size that the hardware can support for a given endpoint. An endpoint is only allocated space in the shared FIFO storage if its RXEPEN/TXEPEN bit = 1. If the endpoint is disabled (RXEPEN/TXEPEN = 0), it is allocated 0 bytes. Register changes that affect the allocation of the shared FIFO storage among endpoints must not be made while there is valid data present in any of the enabled endpoints' FIFOs. Any such changes will render all FIFO contents undefined. Register bits that affect the FIFO allocation are the endpoint enable bits (the TXEPEN and RXEPEN bits of EPCON), the size bits of an enabled endpoint (FFSZ bits of TXCON and RXCON), the isochronous bit of an enabled endpoint (TXISO bit of TXCON and RXISO bit of RXCON), and the FEAT bit of the MCSR register.

If the MCSR.FEAT register bit is set to 1, additional FIFO sizes are enabled for nonisochronous endpoints, as shown in Table 7.14-1.

Table 7.14-1 Programmable FIFO Sizes

FFSZ[1:0]	00	01	10	11
Non-isochronous	16 bytes	64 bytes	8 bytes*	32 bytes*
Isochronous	64 bytes	256 bytes	512 bytes	1024 bytes

* Assumes MCSR.FEAT = 1. If this bit is 0 and FFSZ = 10 or 11, both indicate a size of 64 bytes.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Each FIFO can be programmed independently via the TXCON and RXCON registers, but the total logical size of the enabled endpoints (Tx FIFOs + Rx FIFOs) must not exceed 1120 bytes. The 1120-byte total allows a configuration with a full-sized, 1024-byte isochronous endpoint, a minimum-sized, 64-byte isochronous feedback endpoint, and the required, bidirectional, 16-byte control endpoint. When the dual-packet mode feature is enabled, the device uses a maximum of 2240 bytes of physical storage. This additional physical FIFO storage is managed by the device hardware and is transparent to the user.

7.14.2.5 FIFO Access

The transmit and receive FIFOs are accessed by the application through the register interface (see Tables 7.14-22—7.14-25 for transmit FIFO registers and Tables 7.14-26—7.14-29 for receive FIFO registers).

The transmit FIFO is written to via the TXDAT register, and the receive FIFO is read via the RXDAT register. The particular transmit/receive FIFO is specified by the EPINDEX register. Each FIFO is accessed serially, each RXDAT read increments the receive FIFO read pointer by 1, and each TXDAT write increments the transmit FIFO write pointer by 1.

Each FIFO consists of two data sets to provide the capability for simultaneous read/write access. Control of these pairs of data sets is managed by the hardware, invisible to the application, although the application must be aware of the implications. The receive FIFO read access is advanced to the next data set by firmware setting the RXFFRC bit of RXCON. This bit clears itself after the advance is complete. The transmit FIFO write access is advanced to the next data set by firmware writing the byte count to the TXCNTH/L registers.

The USB access to the receive and transmit FIFOs is managed by the hardware, although the control of the nonisochronous data sets can be overridden by the *ARM* and ATM bits of RXCON and TXCON, respectively. A successful USB transaction causes FIFO access to be advanced to the next data set. A failed USB transaction (e.g., for receive operations, FIFO overrun, data time-out, CRC error, bit stuff error; for transmit operations, FIFO underrun, no ACK from host) causes the FIFO read/write pointer to be reversed to the beginning of the data set to allow transmission retry for nonisochronous transfers.

Transmit FIFO

The transmit FIFOs are circulating data buffers that have the following features:

- Support up to two separate data sets of variable sizes (dual-packet mode).
- Include byte counter register for storing the number of bytes in the data sets.
- Protect against overwriting data in a full FIFO.
- Can retransmit the current data set.

All transmit FIFOs use the same architecture (see Figure 7.14-6). The transmit FIFO and its associated logic can manage up to two data sets: data set 0 (ds0) and data set 1 (ds1). Since two data sets can be used in the FIFO, back-to-back transmissions are supported. Dual-packet mode for transmit FIFOs is enabled by default. Single-packet mode can be enforced by firmware convention (see TXFIF register bits).

The CPU writes to the FIFO location that is specified by the write pointer. After a write, the write pointer automatically increments by 1. The read marker points to the first byte of data written to a data set, and the read pointer points to the next FIFO location to be read by the USB interface. After a read, the read pointer automatically increments by 1.

When a good transmission is completed, the read marker can be advanced to the position of the read pointer to set up for reading the next data set. When a bad transmission is completed, the read pointer can be reversed to the position of the read marker to enable the function interface to reread the last data set for retransmission. The read marker advance and read pointer reversal can be achieved two ways: explicitly by firmware or automatically by hardware, as indicated by bits in the transmit FIFO control register (TXCON).

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

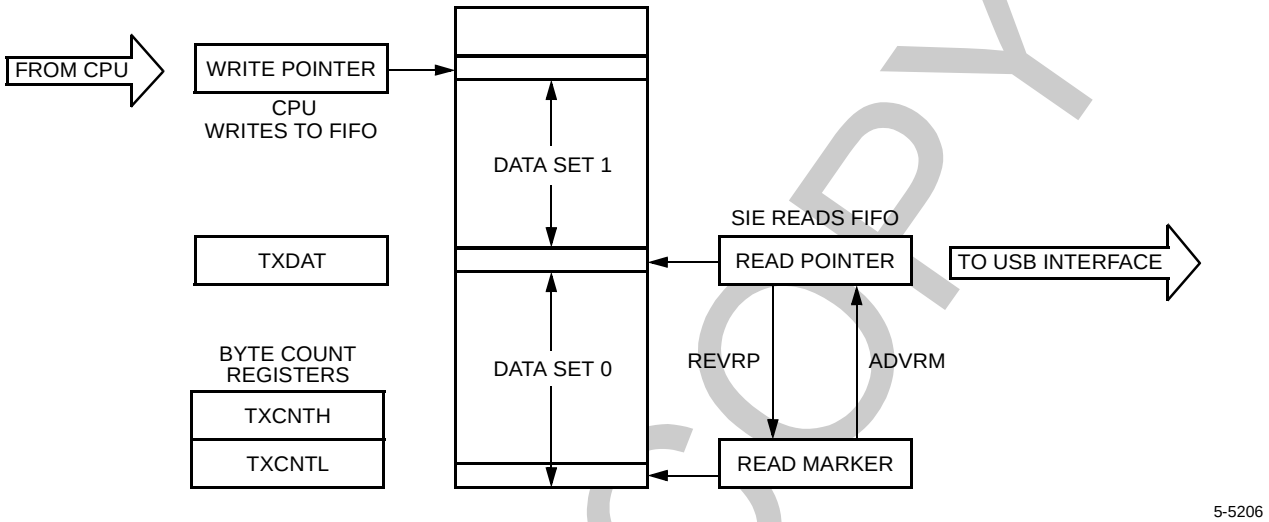


Figure 7.14-6 Transmit FIFO

Receive FIFO

The receive FIFOs are circulating data buffers that have the following features:

- Support up to two separate data sets of variable sizes (dual-packet mode).
- Include byte count register that accesses the number of bytes in data sets.
- Include flags to signal a full FIFO and an empty FIFO.
- Can reread the last data set.

Figure 7.14-7 shows a receive FIFO. A receive FIFO and its associated logic can manage up to two data sets: data set 0 (ds0) and data set 1 (ds1). Since two data sets can be used in the FIFO, back-to-back transmissions are supported. Single-packet mode is established by default after a USS820core reset, which sets the RXSPM register bit. Firmware can enable dual-packet mode by clearing the RXSPM bit to 0.

The receive FIFO is symmetrical to the transmit FIFO in many ways. The SIE writes to the FIFO location specified by the write pointer. After a write, the write pointer automatically increments by 1. The write marker points to the first byte of data written to a data set, and the read pointer points to the next FIFO location to be read by the CPU. After a read, the read pointer automatically increments by 1.

When a good reception is completed, the write marker can be advanced to the position of the write pointer to set up for writing the next data set. When a bad transmission is completed, the write pointer can be reversed to the position of the write marker to enable the SIE to rewrite the last data set after receiving the data again. The write marker advance and write pointer reversal can be achieved two ways: explicitly by firmware or automatically by hardware, as specified by bits in the receive FIFO control register (RXCON).

The CPU should not read data from the receive FIFO before all bytes are received and successfully acknowledged because the reception may be bad.

To avoid overwriting data in the receive FIFO, the SIE monitors the FIFO full flag (RXFULL bit in RXFLG). To avoid reading a byte when the FIFO is empty, the CPU can monitor the FIFO empty flag (RXEMP bit in RXFLG).

The CPU must not change the value of the EPINDEX register during the process of reading a data set from a particular receive FIFO. Once the CPU has read the first byte of a data set, the processor must ensure that the EPINDEX register setting remains unchanged until after the last byte is read from that data set. Registers other than EPINDEX may be read or written during this period, except for registers that affect the overall FIFO configuration, as described in Section 7.14.2.4. If EPINDEX is allowed to change during a data set read, incorrect data will be returned by the USS820core when subsequent bytes are read from

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

the partially read data set. There is no such restriction when writing FIFOs.

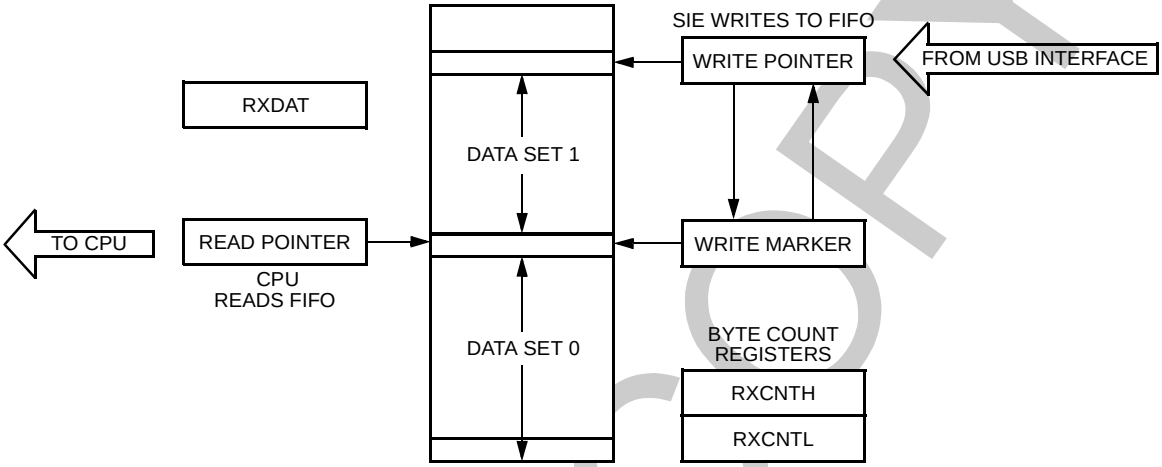


Figure 7.14-7 Receive FIFO

7.14.3 USB Transceiver Impedance Requirement

In order to meet the USB impedance requirement, an external resistor with 1% tolerance must be connected in series with each of the differential I/O pins (D+ and D–). The value of the resistor could be range between 20 and 39 Ohms. Refer to USB transceiver manufacturer's data sheet for a specific resistance value.

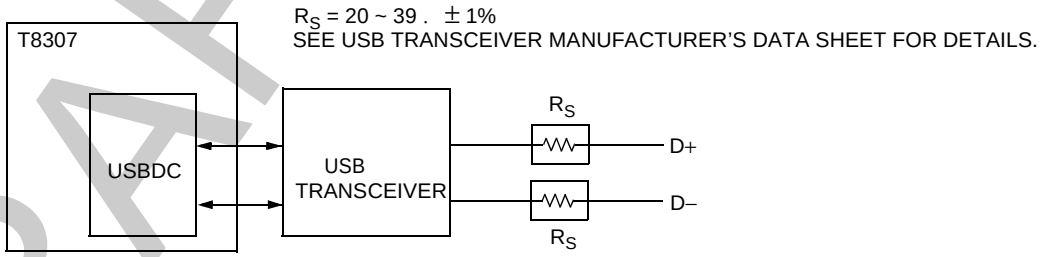


Figure 7.14-8 USB Transceiver Impedance Requirement

7.14.4 DMA Operation for USB

T8307 CP block DMAC can transfer data between the USB FIFO and the CP block memory at a rate of up to 2 KBytes per DMA transfer. Use of the DMA is the easiest way to ensure *ARM* AHB bus efficiency in transferring data to/from the USB FIFO.

T8307 USB device controller only support memory-to-memory DMA transfers.

Since the USB FIFO is only 1 byte wide, FIFO accesses must be byte-only operation. This means 1 byte is transferred for each DMA read/write access to the USB FIFO.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

7.14.5 Interrupts

Figure 7.14-9 describes the device interrupt logic. Each of the indicated USB events are logged in a status register bit. Each status bit has a corresponding enable bit that allows the event to cause an interrupt. Interrupts can be masked globally by the T_IRQ bit of the SCR register. The active level and signaling mode (level vs. pulse) of the IRQ29 output signal can be controlled by the IRQPOL and IRQLVL bits of the SCR register. All interrupts have equal priority—firmware establishes its own priority by the order in which it checks these status bits during interrupt processing.

In addition to generating the combined interrupt (IRQ29), USS820core is also capable of generating a separate interrupt (IRQ30) when USB_SUSP pin is asserted.

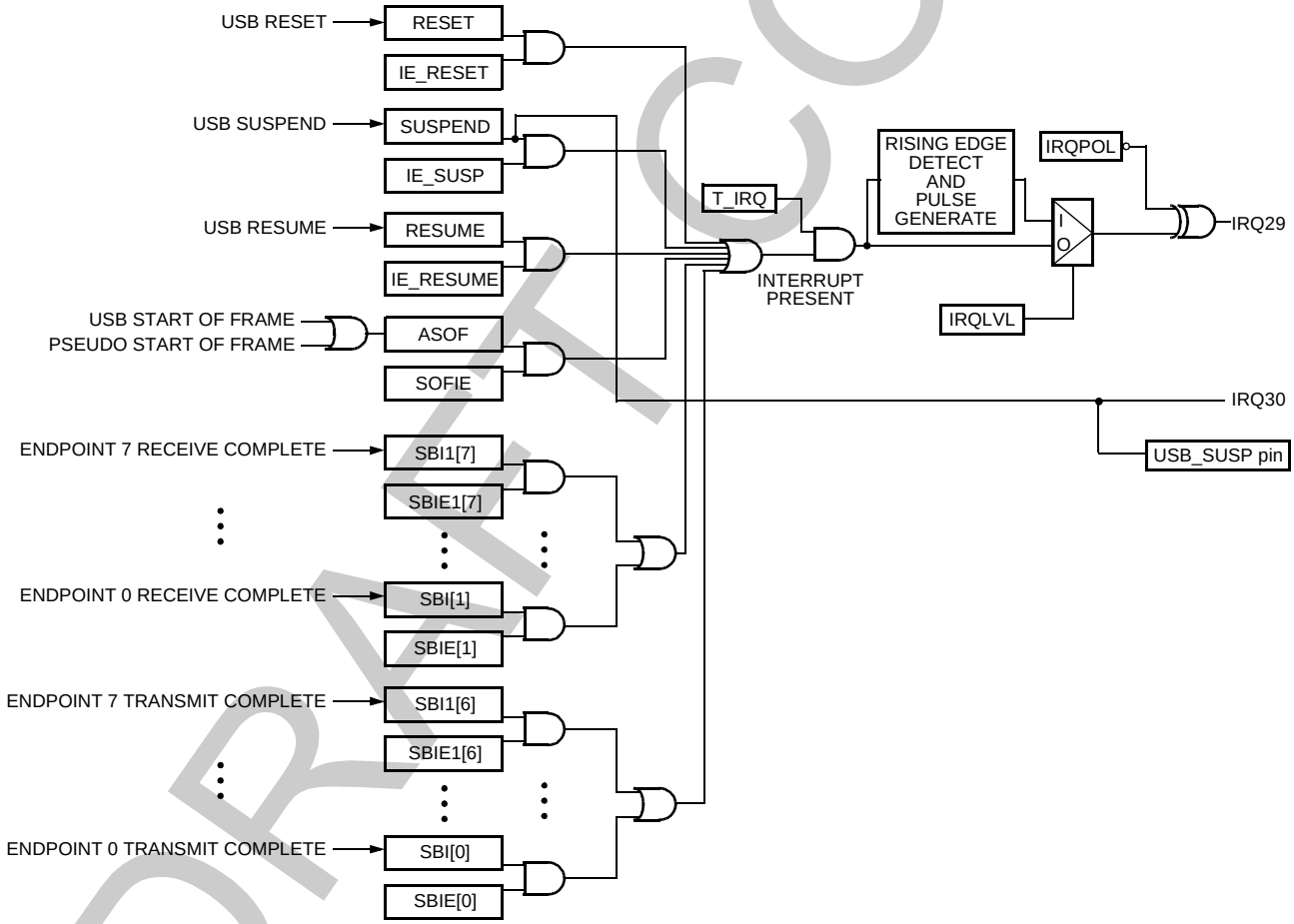


Figure 7.14-9 USS820core Interrupts

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

7.14.6 Firmware Responsibilities for USB SETUP Commands

All SETUP commands are passed through from the USB host to the corresponding receive FIFO (assuming no data transfer errors). Firmware must interpret and execute each command according to its USB definition.

Reception of a new SETUP command can be identified by the RXSETUP bit being set when a receive interrupt is generated. Any old data in the receive FIFO is over-written by a new SETUP command. The STOVW register bit is set by hardware when a new SETUP packet is detected. When the complete SETUP packet has been written, hardware resets the STOVW bit and sets the EDOVW bit. If either the STOVW or EDOVW bit is set, the effect of any firmware actions on the FIFO pointers is blocked. This prevents the FIFO from under-flowing as a result of firmware attempting to read the FIFO while hardware is writing a new setup packet. Firmware must reset the EDOVW bit, read the SETUP command from the FIFO, and then check the STOVW and EDOVW bits. If either is set, the SETUP that was just read out is old and should be discarded. Firmware must then proceed with reading the new SETUP command.

Firmware responsibilities for interpreting and executing USB standard commands are defined in Table 7.14-2.

Table 7.14-2 Firmware Responsibilities for USB SETUP Commands

USB Command	Firmware Responsibility
GET_STATUS	For device status, firmware should write two data bytes to transmit FIFO 0, where bit 0 of byte 0 indicates if the device is self-powered, and bit 1 indicates if the remote wake-up feature is supported (which should equal the value stored in the RWUPE register bit). For interface status, firmware should write two data bytes of zeros. For endpoint status, firmware should write two data bytes to transmit FIFO 0, where bit 0 of byte 0 is the RXSTL or TXSTL bit of the endpoint indicated by the SETUP command.
SET/CLEAR_FEATURE	For the DEVICE_REMOTE_WAKEUP feature, firmware should set/reset the RWUPE register bit. For the ENDPOINT_STALL feature, firmware should set/clear the RXSTL or TXSTL register bit indicated by the SETUP command. Firmware must also handle all side effects of these commands as documented in the USB specification, such as zeroing an endpoint's data toggle bit on CLEAR_FEATURE[stall].
SET_ADDRESS	Firmware should write the FADDR register with the device address indicated by the SETUP command. This write must not occur until after the status stage of the control transfer has completed successfully.
GET_CONFIGURATION, SET_CONFIGURATION, GET_INTERFACE, SET_INTERFACE	Firmware must maintain all information regarding which endpoints, interfaces, alternate settings, and configurations are supported and/or currently enabled. The enabled status of a particular endpoint direction, as specified by the current configuration, interface, and alternate setting, must be indicated in the corresponding RXEPEN or TXEPEN register bit. Firmware must also handle any side effects of these commands as documented in the USB specification, such as zeroing an endpoint's stall and data toggle bits on SET_INTERFACE or SET_CONFIGURATION.
GET_DESCRIPTOR, SET_DESCRIPTOR	Firmware must maintain all information regarding all types of descriptors and write the appropriate descriptor information to transmit FIFO 0 upon receiving GET_DESCRIPTOR, or read the appropriate descriptor information from receive FIFO 0 upon receiving SET_DESCRIPTOR.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Firmware must keep track of the direction of data flow during a control transfer, and detect the start of the status stage by a change in that direction. For control OUT transfers, the status stage will be an IN, and the firmware should write a zero-byte data packet to the transmit FIFO, assuming the command completed successfully. For control IN transfers, the status stage will be an OUT, and the firmware should read the data packet and set the RXFFRC register bit (like any other OUT transfer), again assuming the command completed successfully. This will cause an ACK to be sent to the host, indicating a successful completion.

Firmware should stall endpoint 0 if it receives a standard command that does not match any of the defined commands or a valid command that contains a parameter with a bad value (e.g., GET_STATUS[Endpoint x] when endpoint x is not enabled). Firmware should also stall if the data stage of a control transaction attempts to transfer more bytes than were indicated by the SETUP stage.

Firmware must interpret any vendor or class commands as defined by the application.

7.14.7 Other Firmware Responsibilities

Table 7.14-3 Other Firmware Responsibilities

USB Event	Firmware Responsibility
USB Reset	USB reset can be detected by reading a 1 from the RESET bit of the SSR register. If the USB interrupt is enabled (IE_RESET), this will be indicated by the IRQ29 output. At that time, firmware must reset any information it maintains regarding endpoints, interfaces, alternate settings, and configurations. All RXEPEN and TXEPEN endpoints should be set to 0, except for endpoint 0, which should be set to 1. The function address register FADDR should be set to 0. The data toggle bits for all endpoints should be set to 0 as well. If MCSR.FEAT = 1, FADDR is automatically cleared to 0 when USB reset is detected.
USB Suspend and Resume	Firmware must manage the SUSPEND and RESUME register bits, as documented in Section 7.14.8, in order to meet the USB specifications for bus-powered devices.

7.14.8 Frame Timer Behavior

The USS820core contains an internal frame timer that allows the device to lock to the USB host frame timer, and to synthesize lost SOF packets, as required by the USB specification. The frame timer requires three valid SOF packets from the host in order to lock to the host frame timer. This locked status is indicated by the FTLOCK status bit in SOFH. In order to achieve this lock, the interval between each SOF must be within 45 clocks of the nominal 12,000 clocks, and the successive intervals must be within two clocks of each other. Both of these conditions will be true in a correctly functioning system with no bus errors. While the frame timer is locked, it will synthesize SOFs by setting ASOF and generating an SOF interrupt (if SOFIE = 1) for up to three consecutive frames if SOF packets are no longer received from the host. The frame timer will become unlocked under any of the following conditions:

- Hard or soft reset.
- USB reset.
- The device goes suspended.
- No SOF packets are received from the host for three frames.
- An SOF is received that violates the USB specification for frame interval or previous frame length comparison.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

7.14.9 Suspend and Resume Behavior

Note: In the following sections describing suspend and resume behavior, the following terminology is used:

- Device—The entire T8307 IC that contains the USS820core.
- Application—All electronic components of the device other than the USS820core, such as a microcontroller, RAM, power control logic, reset logic, or crystal.
- Firmware—Code running on the microcontroller, which is part of the application.
- Controller—That intelligent part of the application that uses the USS820core address, data and read/write signals to access its internal registers.
- Powered-off components—Those parts of the application that are connected to the USS820core and powered off during suspend, for example, a microcontroller or RAM.
- Hardware—Logic inside the USS820core.

During a suspend/resume sequence, the following sequence of events occurs:

1. Hardware Suspend Detect: The USS820core detects a suspend request from the host on USB and notifies firmware.
2. Firmware Suspend Initiate: Firmware reacts to the pending suspend request and suspends the device.
3. Hardware Resume Detect/Initiate: Some time later a resume is initiated, either by the host or the application.
4. Hardware Resume Sequence: When the resume is complete, the USS820core notifies firmware.
5. Firmware Resume Sequence: Firmware reacts to the resume and completes any required actions.

The following sections describe each of these steps in more detail.

7.14.9.1 Hardware Suspend Detect

The USS820core detects a USB suspend condition if a J state persists on the bus for at least 3 ms. When this suspend condition is detected, hardware sets the SSR.SUSPEND register status bit and, if IE_SUSP = 1, causes an interrupt.

Suspend detection may be blocked by firmware by setting the SSR.SUSPDIS register bit to 1. SSR.SUSPDIS should only be set for test purposes, never in a running system.

7.14.9.2 Firmware Suspend Initiate

When firmware detects that a suspend request from the host has been detected, it must prepare itself, and any other application components for which it is responsible, for suspend mode. For bus-powered devices, this will normally require turning off power to application components or placing them in low-power mode. When firmware is finished preparing for a device suspend, it should check the SSR.SUSPEND register status bit once more. If this status bit has reset, firmware should abort the suspend sequence, since the host has already awakened the device. This will only happen if firmware is too slow in responding to the suspend detect. If the status bit is still set, firmware should proceed with the suspend sequence. This second check of the status bit guarantees that the device will see wake-up signaling of sufficient length from the host.

To suspend the USS820core, firmware must set the SSR.SUSPEND register control bit to 1, and then reset the bit to 0. This action causes to the USS820core to immediately enter suspend mode.

In order to guarantee correct behavior when resuming, firmware must not attempt any register reads until at least three tRDREC periods have elapsed since resetting the SSR.SUSPEND register control bit.

Since firmware must have the PEND register bit set when modifying the SSR.SUSPEND register bit, and since registers cannot be written while the USS820core is suspended, firmware must remember to reset the PEND register bit after the USS820core resumes.

Since the SSR.SUSPEND register status bit will remain set while the USS820core is suspended, a pending SUSPEND interrupt will remain until the USS820core resumes. For this reason, firmware may wish to reset the SCR.IE_SUSP bit before suspending the USS820core.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

In order to meet the USB specification's current draw limit for suspended devices, the USS820core must turn off its internal clocks. This occurs when the SSR.SUSPEND register control bit is reset by firmware as described above and is indicated by the USS820core USB_SUSP output pin being asserted. While in suspend mode, the USS820core must remain powered, but the USS820core's power consumption will be reduced to almost zero and will remain in this state until a wake-up is signaled.

Self-powered devices will most likely not need to turn off power to other application components during suspend. This is indicated to the USS820core by the SSR.SUSPPO register bit = 0, which should be written by firmware at device initialization time. In such an environment, during suspend, the USS820core outputs and inputs continue to be driven by the USS820core and the application, respectively. In addition, the USS820core bidirectional pins are 3-stated in the USS820core and driven to 0 or 1 by the application.

Bus-powered devices will most likely need to turn off power to other application components during suspend. This is indicated to the USS820core by the SSR.SUSPPO register bit = 1, which should be written by firmware at device initialization time. Such devices can be implemented so that the USS820core USB_SUSP output pin controls power to other application components.

While the USS820core is suspended, its internal registers may still be read, presumably only in self-powered devices. The interface timing for such reads is different from register reads during operational mode. Register writes must not be attempted while the USS820core is suspended, with the possible exception of the SCR.SRESET bit (see the SCR.SRESET description for details). Certain register reads during the nonsuspended state can cause USS820core device register states to change. These reads are described in the Register Reads with Side Effects section. These register reads must not be attempted while the USS820core is suspended.

7.14.10 Hardware Resume Detect/Initiate

Wake-up can be initiated by either the host or the application. A host-signaled wake-up (global resume) is indicated when the host drives a K state on the USB bus. A remote wake-up is initiated by the application by asserting the USS820core RWUPN input signal. The USS820core can also be awakened by firmware writing a 1 to SCR.SRESET if MCSR.FEAT = 1 (see SCR.SRESET description for details). In these cases, the USS820core will initiate a wake-up sequence as described in Section 7.14.10.1.

7.14.10.1 Hardware Resume Sequence

The USS820core starts a wake-up sequence by asynchronously re-enabling its internal clock and deasserting the USB_SUSP output pin. Once the internally generated clocks are stable (a period of 6 ms to 15 ms), then it enables clocks to the entire core and sets the SSR.RESUME register bit, which causes an interrupt if SCRATCH.IE_RESUME register bit = 1. The USS820core will require up to 15 ms to resume functionality after a wake-up sequence is initiated. If the wake-up was a remote wake-up, the USS820cor will then drive wake-up signaling (K) on the USB for 12 ms.

The USS820core requires a minimum of 7 ms from the time a remote wake-up is initiated to the time it can begin transmitting resume signaling upstream. This guarantees adherence to the USB specification for tWTRSM of 5 ms.

7.14.10.2 Firmware Resume Sequence

The USS820core indicates that the resume sequence is complete by setting the SSR.RESUME register bit, and possibly causing an interrupt. When firmware is prepared for the application to return to normal operation, it must reset the SSR.RESUME register bit to allow detection of any subsequent suspend events.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

7.14.11 USB Initialization Sequence

USB core has been designed for low power consumption. It is capable of handling asynchronous wake up sequence originated from the external transceiver through the USB data bus input pins. In order to start up USB core, a proper initialization sequence should be followed. The following code is an example of setting up the USB core prior to any register read or write operations to the core. Refer to Section 7.2.4.15 for the definition of USBFWC set and clear register.

```
//*****************************//
// Setting up USB core          //
//*****************************//
*USBFWC_CLEAR_REG = 0x40;   // clear the firmware transceiver suspend
                            // bit (USBFWC bit 6) to enable USB transceiver
*USBFWC_SET_REG = 0x5; // enable and reset USB core first for several cycles
Tempvar0 = *USBFWC_SET_REG; // Delay to allow uss820core
                            // to synchronously reset flip flops
Tempvar0 = *USBFWC_SET_REG;
*USBFWC_CLEAR_REG = 0x4;      //clear the USB core reset
```

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

7.14.12 Registers

The USB contains registers inside the USS820core and also in the wrapper around the core. The addresses of the USS820core and wrapper registers are listed in Table 6.2-1.

The USS820core registers are described in Section 7.14.12.6—Section 7.14.12.8. The USS820core is controlled through an asynchronous, read/write register interface. Reserved bits of registers must always be written with 0. Writing 1 to these bits may produce undefined results. These bits return undefined values when read.

Table 7.4-13—Table 7.4-24 provide details for each of the USS820core registers. Some of these registers are replicated for each endpoint. The individual, endpoint-specific register is selected by the EPINDEX register.

USS820core registers are clocked in the 48 MHz/12 MHz clock domains. Accesses to the USS820core registers can be lengthened by adding wait-states. The value in the wait code field of the GC2 wrapper register determines the number of cycles involved in USS820core accesses. It is anticipated that a minimum of three AHB bus cycles will be necessary to access the USS820core registers (wait code = 1).

The wrapper registers are described in Section 7.14.12.1—Section 7.14.12.5. All wrapper register names start with GC (for general control). Wrapper registers are clocked by the AHB bus clock (13 MHz—91 MHz). The AHB bus clock is asynchronous to the 48 MHz and 12 MHz clocks.

Some of the wrapper registers are associated with three addresses. The main register address (e.g., GC2) is the normal address of the register and can be used to access all bits in the register in the normal fashion. The set address (e.g., GC2_SET) can be used to set the bits that contain a 1 in the write data and to leave the other bits undisturbed. For example, a write of data 0x3 to a set address would set bits 0 and 1 of the register to 1 and would leave the other bits undisturbed. The clear address (e.g., GC2_CLR) can be used to clear the bits that contain a 1 in the write data and to leave the other bits undisturbed. For example, a write of data 0x3 to a clear address would clear bits 0 and 1 of the register to 0 and would leave the other bits undisturbed.

7.14.12.1 GC1 Register (GC1)

GC1 register is mapped at CP block memory address 0x64018100.

Any write to this register (regardless of data) causes a low-going pulse on the USS820core's RWUPN input port. The pulse lasts one AHB bus clock period. This register always reads as 0s.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

7.14.12.2 GC2 Register (GC2)

Table 7.14-4 GC2 Register, Addresses (0x64018104, Set 0x64018108/Clear 0x6401810C)

Bit #	15	14—6	5—2	1—0
Name	force_ucore_rwup	RSVD	Wait Code	XCVRTYPE

Bit #	Name	R/W	Reset Value	Description
15	force_ucore_rwup	R/W	0	When 1, causes the USB to execute a remote wakeup sequence by asserting the RWUPN input to USS820core. This bit should be pulsed high and then low, (i.e., it should not remain high).
14—6	RSVD	R/W	0	Not used in T8307.
5—2	Wait Code[3:0]	R/W	0	Bit 2 is LSB. This field controls the number of *ARM*-side system bus cycles (13 MHz—91 MHz) used for an access of the USS820core registers. The number of bus cycles is the wait code value +2, consisting of one regular cycle plus (wait code value +1) wait cycles. For example, if Wait Code[3 0] = 0010, then the access takes four cycles, consisting of one regular cycle plus three wait-state cycles. It is anticipated that the wait code will be set to a value of 1 (3-cycle access) when the AHB clock runs at 13 MHz. **Note:** The wait code value does not affect accesses to the GC registers, which are outside the USS820core and do not require additional wait-states. All accesses to the GC registers take [TBD] system bus cycles.
1—0	XCVRTYPE	R/W	0	Transceiver type configuration bits. Default to single-ended type transceiver after reset. **XCVRTYPE** — **Transceiver Type** 01 — Differential type transceiver 10 — Bidirectional differential type transceiver 00 or 11 — Single-ended type transceiver

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

7.14.12.3 USB PLL Control Register (GC3)

This register controls the frequency and trim of the USB PLL. The desired frequency multiplier is written in the M field. This value may range from 3—63. The PLL will multiply its reference clock frequency by M + 1. For standard 48 MHz USB core system operation, this field should be set to 47 (0x2F hex). In general, the USB postdivided PLL output clock frequency is determined by:

$$fPLLout = fssCKI \times \frac{(M + 1)}{(N + 1)\,(P + 1)\,13}, \text{ where } N = 0,\ P = 0.$$

The TRMRST bit controls the PLL trim function, which is used to minimize clock jitter. In automatic switchover mode (MS in the USB clock management register is 0), writing a 0 to TRMRST causes the trim feature to operate automatically upon writing the GC3. If a 1 is written to TRMRST, the trim feature is defeated, which may allow a significantly faster PLL lock time, but with increased clock jitter. In manual switchover mode, it is necessary for software to explicitly activate the PLL trim feature. Two writes are required: first the M field value is written with TRMRST set to 1, then the same M value is rewritten with TRMRST reset to 0. For the trim feature to be disabled in manual mode, only one write is required with TRMRST set to 1 and the M field set to the desired value.

WARNING: Never update the GC3 while the USB PLL is the active USB core system clock.

Note: The USB PLL uses the input clock CKI, therefore, this clock will be maintained while USB is operational. To meet the USB 1.1 specification for full-speed data rate tolerance, the accuracy of the CKI clock frequency must be within the tolerance of ±2500 ppm (±0.25%). See Section 7.1.11 in the USB 1.1 specification.

Note: When the PLL is powered down and subsequently powered up again, the time for the PLL to lock is typically less than 50 ∝s.

Table 7.14-5 USB PLL Control Register (GC3), Addresses (0x64018110, Set 0x64018114/Clear 0x64018118)

Bit #	31—16	15	14—6	5—0
Name	RSVD	TRMRST	RSVD	M

Bit #	Name	R/W	Reset Value	Description
31—16	RSVD	R	—	Reserved. Read as 0. Always write with 0.
15	TRMRST	R/W	0	1: Reset PLL autotrim logic. Autotrim is inactive. 0: Enable PLL autotrim feature. **Note:** In manual switch mode, software must set then clear this bit whenever the M field is modified if PLL autotrim is desired.
14—6	RSVD	R	—	Reserved. Read as 0. Always write with 0.
5—0	M	R/W	101111	PLL frequency multiplier. Values range from 3—63 (0x3—0x3F).

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

7.14.12.4 GC4 Register (GC4)

This read-only register is reserved for test purpose.

Table 7.14-6 GC4 Register, Address (0x6401811C)

Bit #	15—0
Name	RSVD

Bit #	Name	R/W	Reset Value	Description
15—0	RSVD	R	NA	Reserved for test purpose.

7.14.12.5 USB Clock Control Register (GC5)

This register controls the USB core system clock modes and selection, and displays the USB PLL lock detect and USB core system clock source status. The USB core system clock source may be switched to the postdivided USB PLL clock by setting USBCLKC to 1. If the manual mode selection bit (MS) is set to a 1, the clock source will switch immediately to the PLL upon setting the USBCLKC bit regardless of the USB PLL lock status. When MS is 0, selection of the PLL via USBCLKC will be automatically delayed until after the PLL is locked to the desired frequency.

The MS bit also affects the power sequencing of the PLL. When MS is 1, the USB PLL is not powered down automatically when deselected via the USBCLKC bit, rather the power management of the PLL is explicitly controlled by writing the UPLLPWR bit. Also in manual mode, modification of the USB PLL frequency multiplier in the USB PLL control register (GC3) requires software to toggle the TRMRST bit in that register as well for proper autotrim operation.

The postdivided USB PLL output clock may also be made visible on the CPTSTSTOP_CKO external pin via the UPLL2CKO and ALTPINC[11] bits. When ALTPINC[11] is set to 1, CPTSTSTOP_CKO pin is configured to be a test clock output pin. In such a case UPLL2CKO bit further determines whether 1/2 of *ARM* system clock or postdivided USB PLL clock is presented on CPTSTSTOP_CKO output. See Figure 5.2-2 for further detail.

The PLL frequency postdivide circuit is controlled by the DIVBYP and DIVRSTn bits. DIVRSTn allows software to reset the divider at any time, while DIVBYP allows the PLL output clock to be made available to the USB module with no frequency division.

Notes: The value of neither DIVBYP nor DIVRSTn should be modified while the USB PLL is selected as the USB core system clock. Unpredictable behavior of the USB module may result.

CSC shows the current USB core system clock. The USB PLL lock detect status is shown in the LCKDET field.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Table 7.14-7 USB Clock Control Register (GC5), Addresses (0x64018120, Set 0x64018124/Clear 0x64018128)

Bit #	31—10	9	8	7	6	5	4	3—2	1—0
Name	RSVD	USBCLKC	UPLL2CKO	DIVBYP	DIVRSTn	MS	UPLLPWR	LCKDET	CSC

Bit #	Name	R/W	Reset Value	Description
31—10	RSVD	R	—	Reserved. Read as 0. Always write with 0.
9	USBCLKC	R/W	0	Selects USB PLL as USB core system clock source. In manual switch-over mode, the PLL is selected regardless of its lock status. In automatic switch-over mode, clock source selection will be delayed until the PLL is locked. 1: Select the postdivided USB PLL output as the USB core system clock. 0: Deselect the USB PLL source.
8	UPLL2CKO	RW	0	Enables postdivided USB PLL clock output or 1/2 *ARM* system clock onto CPTSTSTOP_CKO. CPTSTSTOP_CKO is a test clock output only when ALTPIN control register bit 11 is set to 1. 1: USB PLL clock output is presented to the CKO. 0: 1/2 of *ARM* system clock output is presented to the CKO, if CKOEN bit is set to 1 (i.e., *ARM* test clock output enabled). Setting UPLL2CKO bits to 1 overrides the setting of CKOEN bits.
7	DIVBYP	R/W	0	Controls frequency division of USBDIVCLK. 1: USBDIVCLK frequency is USB PLL output frequency. 0: USBDIVCLK frequency is USB PLL output frequency divided by 13.
6	DIVRSTn	R/W	0	Resets the USBDIVCLK frequency divider. 1: USB clock divider is out of reset. 0: USB clock divider is in reset. Clock output is held high unless DIVBYP is set.
5	MS	R/W	0	Selects manual switch-over mode for USB system clock. 1: Manual switch-over mode selected. 0: Automatic switch-over mode selected.
4	UPLLPWR	R/W	0	Powers the USB PLL block. In manual switch-over mode, software must set this bit prior to selecting PLL as USB system clock. In automatic switch-over mode, this is automatically set upon selection of PLL. 1: Power is applied to USB PLL block. 0: PLL block is powered down in quiescent state.
3—2	LCKDET	R	00	USB PLL lock detect flags. Bit 3 = 0: Coarse lock not achieved yet. Bit 3 = 1: Coarse lock detected. Bit 2 = 0: Fine lock not achieved yet. Bit 2 = 1: Fine lock detected.
1—0	CSC	R	00	Current USB system clock indicator from clock control block. 00: USB system clock is frozen. 01: USB PLL is current system clock source. 10 and 11: Reserved.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

7.14.12.6 Special Firmware Action for Shared USS820core Register Bits

Since the USS820core registers are not bit-addressable and contain several bits that may be written by either firmware or hardware (shared bits), special care must be taken to avoid incorrect behavior. In particular, firmware must be careful not to write a bit after hardware has updated the bit, but before firmware has recognized the hardware update of the bit.

There are two general cases where this may occur:

1. Direct collision—Firmware does a read-modify-write sequence to update a register bit, but between the firmware read and firmware write, hardware updates the bit. For example, in dual-packet mode, hardware could update an SBI/SBI1 bit while firmware is simultaneously resetting the same SBI/SBI1 bit. This would cause firmware to miss the fact that a new transfer has completed.
2. Indirect collision—Firmware does a read-modify-write sequence to update a register bit, but between the firmware read and firmware write, hardware updates a different bit in the same register. For example, firmware could do a read-modify-write to update the SOFIE bit of the SOFH register, but at the same time, hardware could be updating the ASOF status bit. Firmware would inadvertently reset the ASOF bit without being aware of the hardware update.

These problems can be avoided through the use of the PEND register, which can only be written by firmware. Firmware must ensure that the PEND register bit is set before writing any registers that contain shared bits.

All shared register bits have two copies: a standard copy and a pended copy. The manner in which these register bits are updated varies depending on the value of the PEND register bit, as described in Table 7.14-8. The standard copy is the bit that is read and written during normal operation (PEND = 0). While PEND = 1, hardware updates only affect the pended copy, and firmware updates only affect the standard copy. When firmware resets the PEND bit, the pended copies of the shared bits are used to update the standard copies of the shared bits as described in Table 7.14-9. Through these means, hardware updates during a firmware read-modify-write sequence will not be missed.

Table 7.14-8 Shared Register Bit Update Behavior (ASOF Example)

Bit	Update Behavior While PEND = 0	Update Behavior While PEND = 1	Update Behavior When Firmware Resets PEND to 0
ASOF (standard copy)	Updated by hardware (firmware must not write this register)	Updated by firmware	Updated as documented in Table 7.14-9
ASOF (pended copy)	Not used	Updated by hardware	No longer used

Firmware must execute the following sequence when processing a shared bit (to avoid the direct collision case), or when writing a bit that resides in a register that contains shared bits (to avoid the indirect collision case):

- Set the PEND bit.
- Read the register with the shared bit [Read].
- If processing a shared bit, respond to the shared bit. For example, for an SBI/SBI1 bit, process any data sets present for that endpoint.
- Update the bit [Modify].
- Write the register with the shared bit with the modified data [Write].
- Reset the PEND bit.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

When a data set is written to a receive FIFO, that FIFO's SBI/SBI1 register bit will set. Firmware must process the indicated receive data set and, in doing so, manage that FIFO's SBI/SBI1 bit according to the sequence described in this section. In dual-packet mode, it is possible that a second data set will be written to a receive FIFO before firmware has completed processing of the initial data set. This second data set could have been written either before or after firmware set the PEND bit to 1. Therefore, firmware cannot determine whether or not this second receive done indication was saved in the pended copy of the SBI/SBI1 bit. Because of this uncertainty, firmware must process all receive data sets that are present in the indicated FIFO before resetting the PEND bit to 0. If the receive done indication of the second data set was in fact saved in the pended SBI/SBI1 register, then the standard copy of the SBI/SBI1 bit will be set when firmware resets the PEND bit to 0.

In this case, the SBI/SBI1 bit will be set even though there is no corresponding data set present in the receive FIFO. Therefore, firmware must be prepared to service a receive done interrupt where no data sets are present in the indicated FIFO.

Table 7.14-9 shows the values loaded into each of the standard copies of the shared register bits when firmware resets the PEND register bit.

Table 7.14-9 Shared Register Update Values When Firmware Resets PEND

Register	Bit(s)	Update Value
SBI	All bits	Set to 1 if standard copy = 1 or pended copy = 1.
SBI1	All bits	Set to 1 if standard copy = 1 or pended copy = 1.
RXSTAT	RXSETUP	Loaded with pended copy if USB action updated RXSETUP while PEND was set.
RXSTAT	EDOVW	Set to 1 if standard copy = 1 or pended copy = 1.
EPCON	RXSTL	Set to 1 if standard copy = 1 or pended copy = 1.
SOFH	ASOF	Set to 1 if standard copy = 1 or pended copy = 1.
SOFH	TS	Loaded with pended copy if USB SOF was received while PEND was set.

Table 7.14-9 Shared Register Update Values When Firmware Resets PEND (continued)

Register	Bit(s)	Update Value
SOFL	All bits	Loaded with pended copy if USB SOF was received while PEND was set.
SSR	RESET	Set to 1 if standard copy = 1 or pended copy = 1.

The register bits that are only updated by firmware, but reside in registers with shared bits and must, therefore, be updated only while PEND is set, are shown in Table 7.14-10.

Table 7.14-10 Register Bits Only Updated While PEND Is Set

Register	Bit(s)
RXSTAT	RXSEQ
EPCON	All bits except RXSTL
SOFH	SOFIE
SSR	SUSPPO, SUSPDIS, RESUME, SUSPEND

Firmware should attempt to minimize the period during which PEND is set in order to minimize the distortion of the detection of hardware events.

7.14.12.7 USS820core Register Reads with Side Effects

In general, USS820core register reads do not have side effects—they do not cause any device state to change. The following are exceptions to this rule:

- RXDAT reads cause the internal Rx FIFO read pointer to change and possibly cause the RXFLG.RXURF register bit to set.
- RXCNTH/RXCNTL reads while RXFLG.RXFIF = 00 cause the RXFLG.RXURF register bit to set.
- LOCK reads restart the register unlock sequence after suspend (described in *Special Action Required by USS820D/USS-825 After Suspend*—AP97-058CMPR-04).
- Any register reads during a register unlock sequence after suspend, other than the LOCK register, cause the unlock sequence to fail and require the sequence to be restarted.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

7.14.12.8 USS820core Register Descriptions

Serial Bus Interrupt Enable Register (SBIE)—Address: 0x64018058; Default: 0000 0000B

This register enables and disables the receive and transmit done interrupts for function endpoints 0 through 3.

Table 7.14-11 Serial Bus Interrupt Enable Register (SBIE)—Address: 0x64018058; Default: 0000 0000B

Bit	7	6	5	4	3	2	1	0
Name	FRXIE3	FTXIE3	FRXIE2	FTXIE2	FRXIE1	FTXIE1	FRXIE0	FTXIE0
Read/Write	R/W							

Bit*	Name	Function/Description
7	FRXIE3	**Function Receive Interrupt Enable 3.** Enables receive done interrupt for endpoint 3 (FRXD3).
6	FTXIE3	**Function Transmit Interrupt Enable 3.** Enables transmit done interrupt for endpoint 3 (FTXD3).
5	FRXIE2	**Function Receive Interrupt Enable 2.** Enables receive done interrupt for endpoint 2 (FRXD2).
4	FTXIE2	**Function Transmit Interrupt Enable 2.** Enables transmit done interrupt for endpoint 2 (FTXD2).
3	FRXIE1	**Function Receive Interrupt Enable 1.** Enables receive done interrupt for endpoint 1 (FRXD1).
2	FTXIE1	**Function Transmit Interrupt Enable 1.** Enables transmit done interrupt for endpoint 1 (FTXD1).
1	FRXIE0	**Function Receive Interrupt Enable 0.** Enables receive done interrupt for endpoint 0 (FRXD0).
0	FTXIE0	**Function Transmit Interrupt Enable 0.** Enables transmit done interrupt for endpoint 0 (FTXD0).

* For all bits, a 1 indicates the interrupt is enabled and causes an interrupt to be signaled to the microcontroller. A 0 indicates the associated interrupt source is disabled and cannot cause an interrupt. However, the interrupt bit's value is still reflected in the SBI/SBI1 register. All of these bits can be read/written by firmware.

Serial Bus Interrupt Enable Register 1 (SBIE1)—Address: 0x6401805C; Default: 0000 0000B

This register enables and disables the receive and transmit done interrupts for function endpoints 4 through 7.

Table 7.14-12 Serial Bus Interrupt Enable Register 1 (SBIE1)—Address: 0x6401805C; Default: 0000 0000B

Bit	7	6	5	4	3	2	1	0
Name	FRXIE7	FTXIE7	FRXIE6	FTXIE6	FRXIE5	FTXIE5	FRXIE4	FTXIE4
Read/Write	R/W							

Bit*	Name	Function/Description
7	FRXIE7	**Function Receive Interrupt Enable 7.** Enables receive done interrupt for endpoint 7 (FRXD7).
6	FTXIE7	**Function Transmit Interrupt Enable 7.** Enables transmit done interrupt for endpoint 7 (FTXD7).
5	FRXIE6	**Function Receive Interrupt Enable 6.** Enables receive done interrupt for endpoint 6 (FRXD6).
4	FTXIE6	**Function Transmit Interrupt Enable 6.** Enables transmit done interrupt for endpoint 6 (FTXD6).
3	FRXIE5	**Function Receive Interrupt Enable 5.** Enables receive done interrupt for endpoint 5 (FRXD5).
2	FTXIE5	**Function Transmit Interrupt Enable 5.** Enables transmit done interrupt for endpoint 5 (FTXD5).
1	FRXIE4	**Function Receive Interrupt Enable 4.** Enables receive done interrupt for endpoint 4 (FRXD4).
0	FTXIE4	**Function Transmit Interrupt Enable 4.** Enables transmit done interrupt for endpoint 4 (FTXD4).

* For all bits, a 1 indicates the interrupt is enabled and causes an interrupt to be signaled to the microcontroller. A 0 indicates the associated interrupt source is disabled and cannot cause an interrupt. However, the interrupt bit's value is still reflected in the SBI/SBI1 register. All of these bits can be read/written by firmware.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Serial Bus Interrupt Register (SBI)—Address: 0x64018050; Default: 0000 0000B

This register contains the USB function's transmit and receive done interrupt flags for nonisochronous endpoints. These bits are never set for isochronous endpoints.

Table 7.14-13 Serial Bus Interrupt Register (SBI)—Address: 0x64018050; Default: 0000 0000B

Bit	7	6	5	4	3	2	1	0
Name	FRXD3	FTXD3	FRXD2	FTXD2	FRXD1	FTXD1	FRXD0	FTXD0
Read/Write	R/W (S*)							

Bit	Name	Function/Description
7	FRXD3	**Function Receive Done Flag, Endpoint 3.**
6	FTXD3	**Function Transmit Done Flag, Endpoint 3.**
5	FRXD2	**Function Receive Done Flag, Endpoint 2.**
4	FTXD2	**Function Transmit Done Flag, Endpoint 2.**
3	FRXD1	**Function Receive Done Flag, Endpoint 1.**
2	FTXD1	**Function Transmit Done Flag, Endpoint 1.**
1	FRXD0	**Function Receive Done Flag, Endpoint 0.**
0	FTXD0	**Function Transmit Done Flag, Endpoint 0.**

* S = shared bit. See Section 7.14.12.6.

For all bits in the interrupt flag register, a 1 indicates that an interrupt is actively pending; a 0 indicates that the interrupt is not active. The interrupt status is shown regardless of the state of the corresponding interrupt enable bit in the SBIE/SBIE1.

Hardware can only set bits to 1. In normal operation, firmware should only clear bits to 0. Firmware can also set the bits to 1 for test purposes. This allows the interrupt to be generated in firmware.

A set receive bit indicates either that valid data is waiting to be serviced in the Rx FIFO for the indicated endpoint and that the data was received without error and has been acknowledged, or that data was received with a receive data error requiring firmware intervention to be cleared.

A set transmit bit indicates either that data has been transmitted from the Tx FIFO for the indicated endpoint and has been acknowledged by the host, or that data was transmitted with an error requiring firmware intervention to be cleared.

If TXNAKE = 1, this also may indicate that a NAK was sent to the host in response to an IN packet that was received when TXFIF = 00. This condition also sets TXVOID. This SBI/SBI1 setting will persist until firmware clears TXVOID (or clears TXNAKE).

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Serial Bus Interrupt 1 Register (SBI1)—Address: 0x64018054; Default: 0000 0000B

This register contains the USB function's transmit and receive done interrupt flags for nonisochronous endpoints. These bits are never set for isochronous endpoints.

Table 7.14-14 Serial Bus Interrupt 1 Register (SBI1)—Address: 0x64018054; Default: 0000 0000B

Bit	7	6	5	4	3	2	1	0
Name	FRXD7	FTXD7	FRXD6	FTXD6	FRXD5	FTXD5	FRXD4	FTXD4
Read/Write	R/W (S*)							

Bit	Name	Function/Description
7	FRXD7	**Function Receive Done Flag, Endpoint 7.**
6	FTXD7	**Function Transmit Done Flag, Endpoint 7.**
5	FRXD6	**Function Receive Done Flag, Endpoint 6.**
4	FTXD6	**Function Transmit Done Flag, Endpoint 6.**
3	FRXD5	**Function Receive Done Flag, Endpoint 5.**
2	FTXD5	**Function Transmit Done Flag, Endpoint 5.**
1	FRXD4	**Function Receive Done Flag, Endpoint 4.**
0	FTXD4	**Function Transmit Done Flag, Endpoint 4.**

* S = shared bit. See Section 7.14.12.6.

For all bits in the interrupt flag register, a 1 indicates that an interrupt is actively pending; a 0 indicates that the interrupt is not active. The interrupt status is shown regardless of the state of the corresponding interrupt enable bit in the SBIE/SBIE1.

Hardware can only set bits to 1. In normal operation, firmware should only clear bits to 0. Firmware can also set the bits to 1 for test purposes. This allows the interrupt to be generated in firmware.

A set receive bit indicates either that valid data is waiting to be serviced in the Rx FIFO for the indicated endpoint and that the data was received without error and has been acknowledged, or that data was received with a receive data error requiring firmware intervention to be cleared.

A set transmit bit indicates either that data has been transmitted from the Tx FIFO for the indicated endpoint and has been acknowledged by the host, or that data was transmitted with an error requiring firmware intervention to be cleared.

If TXNAKE = 1, this also may indicate that a NAK was sent to the host in response to an IN packet that was received when TXFIF = 00. This condition also sets TXVOID. This SBI/SBI1 setting will persist until firmware clears TXVOID (or clears TXNAKE).

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Start of Frame High Register (SOFH)—Address: 0x6401803C; Default: 0000 0000B

This register contains isochronous data transfer enable and interrupt bits and the upper 3 bits of the 11-bit time stamp received from the host.

Table 7.14-15 Start of Frame High Register (SOFH)—Address: 0x6401803C; Default: 0000 0000B

Bit	7	6	5	4	3	2	1	0
Name	SOFACK	ASOF	SOFIE	FTLOCK	RSVD	TS10	TS9	TS8
Read/Write	R	R/W (S*)	R/W (P*)	R	R/W (P*)	R/W (S*)		

Bit	Name	Function/Description
7	SOFACK	**SOF Token Received Without Error (Read Only)**. When set, this bit signifies that the 11-bit time stamp stored in SOFL and SOFH is valid. This bit is updated every time an SOF token is received from the USB bus, and it is cleared when an artificial SOF is generated by the frame timer. This bit is set and cleared by hardware.
6	ASOF	**Any Start of Frame.** This bit is set by hardware to signify that a new frame has begun. The interrupt can result either from the reception of an actual SOF packet or from an artificially generated SOF from the frame timer. This interrupt is asserted in hardware even if the frame timer is not locked to the USB bus frame timing. When set, this bit indicates that either the actual SOF packet was received or an artificial SOF was generated by the frame timer. Setting this bit to 1 by firmware has the same effect as when it is set by hardware. This bit must be cleared to 0 by firmware if MCSR.FEAT = 1. If MCSR.FEAT = 0, this bit clears itself after one t_{CLK}. This bit also serves as the SOF interrupt flag. This interrupt is only asserted in hardware if the SOF interrupt is enabled (SOFIE set) and the interrupt channel is enabled.
5	SOFIE	**SOF Interrupt Enable**. When set, setting the ASOF bit causes an interrupt request to be generated if the interrupt channel is enabled. Hardware reads this bit but does not write to it.
4	FTLOCK	**Frame Timer Lock (Read Only)**. When set, this bit signifies that the frame timer is presently locked to the USB bus frame time. When cleared, this bit indicates that the frame timer is attempting to synchronize the frame time.
3	RSVD	**Reserved**. Read as 0. Always write with a zero.
2:0	TS[10:8]	**Time Stamp Received from Host**. TS[10:8] are the upper 3 bits of the 11-bit frame number issued with an SOF token. This time stamp is valid only if the SOFACK bit is set.

* S = shared bit. P = PEND must be set when writing this bit. See Section 7.14.12.6.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Start of Frame Low Register (SOFL)—Address: 0x64018038; Default: 0000 0000B

This register contains the lower 8 bits of the 11-bit time stamp received from the host.

Table 7.14-16 Start of Frame Low Register (SOFL)—Address: 0x64018038; Default: 0000 0000B

Bit	7	6	5	4	3	2	1	0
Name	TS7	TS6	TS5	TS4	TS3	TS2	TS1	TS0
Read/Write	R/W (S*)							

Bit	Name	Function/Description
7—0	TS[7:0]	**Time Stamp Received from Host**. This time stamp is valid only if the SOFACK bit in the SOFH register is set. TS[7:0] are the lower 8 bits of the 11-bit frame number issued with an SOF token. The time stamp remains at its previous value if an artificial SOF is generated, and it is up to firmware to update it. These bits are set and cleared by hardware.

* S = shared bit. See Section 7.14.12.6.

Endpoint Index Register (EPINDEX)—Address: 0x64018028; Default: 0000 0000B

This register identifies the endpoint pair. The register's contents select the transmit and receive FIFO pair and serve as an index to endpoint-specific special function registers (SFRs).

Table 7.14-17 Endpoint Index Register (EPINDEX)—Address: 0x64018028; Default: 0000 0000B

Bit	7—3	2	1	0
Name	RSVD	EPINX2	EPINX1	EPINX0
Read/Write	—	R/W		

Bit	Name	Function/Description
7—3	RSVD	**Reserved.** Write 0s to these bits. Reads always return 0s.
2—0	EPINX[2:0]	**Endpoint Index.** **EPINDEX*** — **Function Endpoint** 0000 0000 — Function endpoint 0 0000 0001 — Function endpoint 1 0000 0010 — Function endpoint 2 0000 0011 — Function endpoint 3 0000 0100 — Function endpoint 4 0000 0101 — Function endpoint 5 0000 0110 — Function endpoint 6 0000 0111 — Function endpoint 7 The EPINDEX register must not be changed during a sequence of RXDAT reads of a particular data set.

* The EPINDEX register identifies the endpoint pair and selects the associated transmit and receive FIFO pair. The value in this register plus SFR addresses select the associated band of endpoint-indexed SFRs (TXDAT, TXCON, TXFLG, TXCNTH/L, RXDAT, RXCON, RXFLG, RXCNTH/L, EPCON, TXSTAT, and RXSTAT).

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Endpoint Control Register (EPCON)—Address: 0x6401802C; Default: Endpoint 0 = 0011 0101B; Others = 0001 0000B

This SFR configures the operation of the endpoint specified by EPINDEX. This register is endpoint indexed.

Table 7.14-18 Endpoint Control Register (EPCON)—Address: 0x6401802C; Default: Endpoint 0 = 0011 0101B; Others = 0001 0000B

Bit	7	6	5	4	3	2	1	0
Name	RXSTL	TXSTL	CTLEP	RXSPM	RXIE	RXEPEN	TXOE	TXEPEN
Read/Write	R/W (S*)	R/W(P*)						

Bit	Name	Function/Description
7	RXSTL	**Stall Receive Endpoint**. When set, this bit stalls the receive endpoint. Firmware must clear this bit only after the host has intervened through commands sent down endpoint 0. When this bit is set and RXSETUP is clear, the receive endpoint responds with a STALL handshake to a valid OUT token. When this bit is set and RXSETUP is set, the receive endpoint will NACK. This bit does not affect the reception of SETUP tokens by a control endpoint. This bit is set by the hardware if the data phase of the status stage of a control transfer does not use the correct data PID (DATA1) or has more than 0 data bytes.
6	TXSTL	**Stall Transmit Endpoint**. When set, this bit stalls the transmit endpoint. Firmware must clear this bit only after the host has intervened through commands sent down endpoint 0. When this bit is set and RXSETUP is clear, the transmit endpoint responds with a STALL handshake to a valid IN token. When this bit is set and RXSETUP is set, the receive endpoint will NACK.
5	CTLEP	**Control Endpoint.** When set, this bit configures the endpoint as a control endpoint. Only control endpoints are capable of receiving SETUP tokens.
4	RXSPM	**Receive Single-Packet Mode**. When set, this bit configures the receive endpoint for single data packet operation. When enabled, only a single data packet is allowed to reside in the receive FIFO. **Note:** For control endpoints (CTLEP = 1), this bit should be set for single-packet mode operation as the recommended firmware model. However, it is possible to have a control endpoint configured in dual-packet mode as long as the firmware handles the endpoint correctly.
3	RXIE	**Receive Input Enable**. When set, this bit enables data from the USB to be written into the receive FIFO. If cleared, the endpoint responds to an OUT token by ignoring the data and returning a NACK handshake to the host (unless RXSTL is set, in which case a STALL is returned). This bit does not affect a valid SETUP token.
2	RXEPEN	**Receive Endpoint Enable**. When set, this bit enables the receive endpoint. When disabled, the endpoint does not respond to a valid OUT or SETUP token. This bit is hardware read only and has the highest priority among RXIE and RXSTL. **Note:** Endpoint 0 is enabled for reception upon reset.
1	TXOE	**Transmit Output Enable**. When set, this bit enables the data in TXDAT to be transmitted. If cleared, the endpoint returns a NACK handshake to a valid IN token if the TXSTL bit is not set.
0	TXEPEN	**Transmit Endpoint Enable**. When set, this bit enables the transmit endpoint. When disabled, the endpoint does not respond to a valid IN token. This bit is hardware read only. **Note:** Endpoint 0 is enabled for transmission upon reset.

* S = shared bit. P = PEND must be set when writing this bit. See Section 7.14.12.6.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Endpoint Transmit Status Register (TXSTAT)—Address: 0x64018030; Default: 0000 0000B

This register contains the current endpoint status of the transmit FIFO specified by EPINDEX. This register is endpoint indexed.

Table 7.14-19 Endpoint Transmit Status Register (TXSTAT)—Address: 0x64018030; Default: 0000 0000B

Bit	7	6	5	4	3	2	1	0
Name	TXSEQ	TXDSAM	TXNAKE	TXFLUSH	TXSOVW	TXVOID	TXERR	TXACK
Read/Write	R/W*	R/W	R/W	R	W*	R/W†	R	

Bit	Name	Function/Description
7	TXSEQ	**Transmitter Current Sequence Bit (Read, Conditional Write).*** This bit is transmitted in the next PID and toggled on a valid ACK handshake. This bit is toggled by hardware on a valid SETUP token. This bit can be written by firmware if the TXSOVW bit is set when written together with the next TXSEQ value.
6	TXDSAM	**Transmit Data-Set-Available Mode.** If set, a NAK response to an IN token causes the corresponding RXAV/TXAV bit in the DSAV register to set, and the DSA output pin to assert (if enabled by MCSR.BDFEAT), rather than the standard condition (transmit data set empty). This only occurs on NAKs caused by TXFIF = 00. This bit must not be set for isochronous endpoints. When reset to 0 (along with MCSR.FEAT, MCSR.BDFEAT, and TXSTAT.TXNAKE), USS820core will behave like revision B.
5	TXNAKE	**Transmit NAK Mode Enable.** If set, a NAK response to an IN token causes the TXVOID bit and the corresponding bits in the SBI/SBI1 register to set, causing an IRQ29 interrupt (if enabled). This only occurs on NAKs caused by TXFIF = 00. This bit must not be set for isochronous endpoints. When set, this bit also changes the meaning and usage of the TXSTAT.TXVOID bit. When reset to 0 (along with MCSR.FEAT, MCSR.BDFEAT, and TXSTAT.TXDSAM), USS820core will behave like revision B.
4	TXFLUSH	**Transmit FIFO Packet Flushed (Read Only).** Updated at each SOF. When set, this bit indicates that hardware flushed a stale isochronous data packet from the transmit FIFO at SOF. Behavior when MCSR.FEAT = 0: To guard against a missed IN token in isochronous mode, if, with TXFIF[1:0] = 11, no IN token is received for the current endpoint, hardware automatically flushes the oldest packet and decrements the TXFIF[1:0] value. This flush does not occur if there is only one data set present (TXFIF = 01/10). Behavior when MCSR.FEAT = 1: A firmware data set write causes a TXFIF bit to set. For isochronous endpoints, this data set does not become visible to the host until the next SOF. The data set is intended to be read out during that frame. If that read does not occur (possibly due to a lost IN packet), that data set is flushed at the next SOF, setting TXFLUSH. If firmware writes two data sets during a single frame (TXFIF must have equalled 00 at the start of that frame), the first, older data set written is flushed at the subsequent SOF, setting TXFLUSH.

* For normal operation, this bit should not be modified by the user except as required by the implementation of USB standard commands, such as SET_CONFIGURATION, SET_INTERFACE, and CLEAR_FEATURE [stall]. The SIE handles all sequence bit tracking required by normal USB traffic, as documented in the USB specification, Section 8.6.

† Only writable if TXNAKE = 1.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Table 7.14-19 Endpoint Transmit Status Register (TXSTAT)—Address: 0x64018030; Default: 0000 0000B (continued)

Bit	Name	Function/Description
3	TXSOVW	**Transmit Data Sequence Overwrite Bit.*** Writing a 1 to this bit allows the value of the TXSEQ bit to be overwritten. Writing a 0 to this bit has no effect on TXSEQ. This bit always returns 0 when read.
2	TXVOID	**Transmit Void.†** Behavior when TXNAKE = 0: This bit is read only if TXNAKE = 0. Indicates a void condition has occurred in response to a valid IN token. Transmit void is closely associated with the NACK/STALL handshake returned by the function after a valid IN token. This void condition occurs when the endpoint output is disabled (TXOE = 0) or stalled (TXSTL = 1), the corresponding receive FIFO contains a setup packet (RXSETUP = 1), the FIFO contains no valid data sets (TXFIF = 00), or there is an existing FIFO error (TXURF = 1 or TXOVF = 1). This bit is used to check any NACK/STALL handshake returned by the function. This bit does not affect the FTXDx, TXERR, or TXACK bits. This bit is updated by hardware at the end of a nonisochronous transaction in response to a valid IN token. For isochronous transactions, this bit is not updated until the next SOF. This bit is not updated at SOF if TXFLUSH is performed. Behavior when TXNAKE = 1: When TXNAKE = 1, this bit becomes writable by firmware. The meaning of the bit is also changed, to indicate only that a NAK was sent to the host in response to an IN when TXFIF = 00. Hardware setting of this bit always takes priority over firmware writes. Hardware setting of this bit also causes the corresponding SBI/SBI1 bit to set, possibly causing an interrupt. That setting will persist until TXVOID is cleared by firmware.
1	TXERR	**Transmit Error (Read Only).** Indicates an error condition has occurred with the transmission. Complete or partial data has been transmitted. The error can be one of the following: 1. Data transmitted successfully but no handshake received. 2. Transmit FIFO goes into underrun condition while transmitting. These conditions also cause the corresponding transmit done bit, FTXDx in SBI or SBI1, to be set. For nonisochronous transactions, TXERR is updated by hardware along with the TXACK bit at the end of data transmission. TEXERR and TXACK are updated at the same time—one bit is set to 1, and the other is reset to 0. For isochronous transactions, TXERR is not updated until the next SOF. This bit is not updated at SOF if TXFLUSH is performed.
0	TXACK	**Transmit Acknowledge (Read Only).** Indicates data transmission completed and acknowledged successfully. This condition also causes the corresponding transmit done bit, FTXDx in SBI or SBI1, to be set. For nonisochronous transactions, TXACK is updated by hardware along with the TXERR bit at the end of data transmission. TEXERR and TXACK are updated at the same time—one bit is set to 1, and the other is reset to 0. For isochronous transactions, TXACK is not updated until the next SOF. This bit is not updated at SOF if TXFLUSH is performed.

* For normal operation, this bit should not be modified by the user except as required by the implementation of USB standard commands, such as SET_CONFIGURATION, SET_INTERFACE, and CLEAR_FEATURE [stall]. The SIE handles all sequence bit tracking required by normal USB traffic, as documented in the USB specification, Section 8.6.

† Only writable if TXNAKE = 1.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Endpoint Receive Status Register (RXSTAT)—Address: 0x64018034; Default: 0000 0000B

This register contains the current endpoint status of the receive FIFO specified by EPINDEX. This register is an endpoint-indexed SFR.

Table 7.14-20 Endpoint Receive Status Register (RXSTAT)—Address: 0x64018034; Default: 0000 0000B

Bit	7	6	5	4	3	2	1	0
Name	RXSEQ	RXSETUP	STOVW	EDOVW	RXSOVW	RXVOID	RXERR	RXACK
Read/Write	R/W* (P†)	R/W (S†)	R	R/W (S†)	W (P†)	R		

Bit	Name	Function/Description
7	RXSEQ	**Receiver Endpoint Sequence Bit (Read, Conditional Write).*** This bit is toggled on completion of an ACK handshake in response to an OUT token. This bit is set (or cleared) by hardware after reception of a SETUP token. If the RXSOVW bit is set, this bit can be written by firmware when written along with the new RXSEQ value. **Note:** Always verify this bit after writing to ensure that there is no conflict with hardware, which may occur if a new SETUP token is received.
6	RXSETUP	**Received SETUP Token**. This bit is set by hardware when a valid SETUP token has been received. When set, this bit causes received IN or OUT tokens to be NACKed until the bit is cleared to allow proper data management for the transmit and receive FIFOs from the previous transaction. IN or OUT tokens are NACKed even if the endpoint is stalled (RXSTL or TXSTL) to allow a control transaction to clear a stalled endpoint. Firmware must clear this bit after it has finished reading out the SETUP packet and is prepared for the next stage of the control transaction (data or status). For a stalled control endpoint, this bit should not be cleared until the RXSTL/TXSTL bits have been cleared.
5	STOVW	**Start Overwrite Flag (Read Only).** This bit is set by hardware upon receipt of a SETUP token for any control endpoint to indicate that the receive FIFO is being overwritten with new SETUP data. When set, the FIFO state (RXFIF and read pointer) resets and is locked for this endpoint until EDOVW is set. This prevents a prior, ongoing firmware read from corrupting the read pointer as the receive FIFO is being cleared and new data is being written into it. This bit is cleared by hardware at the end of handshake phase transmission of the SETUP stage. This bit is used only for control endpoints.
4	EDOVW	**End Overwrite Flag.** This flag is set by hardware during the handshake phase of a SETUP stage. It is set after every SETUP packet is received and must be cleared prior to reading the contents of the FIFO. When set, the FIFO state (RXFIF and read pointer) remains locked for this endpoint until this bit is cleared. This prevents a prior, ongoing firmware read from corrupting the read pointer after the new data has been written into the receive FIFO. This bit is used only for control endpoints.
3	RXSOVW	**Receive Data Sequence Overwrite Bit.*** Writing a 1 to this bit allows the value of the RXSEQ bit to be overwritten. Writing a 0 to this bit has no effect on RXSEQ. This bit always returns 0 when read.

* For normal operation, this bit should not be modified by the user except as required by the implementation of USB standard commands, such as SET_CONFIGURATION, SET_INTERFACE, and CLEAR_FEATURE [stall]. The SIE handles all sequence bit tracking required by normal USB traffic, as documented in the USB specification, Section 8.6.

† S = shared bit. P = PEND must be set when writing this bit. See Section 7.14.12.6.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Table 7.14-20 Endpoint Receive Status Register (RXSTAT)—Address: 0x64018034; Default: 0000 0000B (continued)

Bit	Name	Function/Description
2	RXVOID	**Receive Void (Read Only).** Indicates a void condition has occurred in response to a valid OUT token. Receive void is closely associated with the NACK/STALL handshake returned by the function after a valid OUT token. This void condition occurs when the endpoint input is disabled (RXIE = 0) or stalled (RXSTL = 1), the FIFO contains a setup packet (RXSETUP = 1), the FIFO has no available data sets (RXFIF = 11, or RXFIF = 01/10 and RXSPM = 1), or there is an existing FIFO error (RXURF = 1 or RXOVF = 1). This bit is set and cleared by hardware. For nonisochronous transactions, this bit is updated by hardware at the end of the transaction in response to a valid OUT token. For isochronous transactions, it is not updated until the next SOF.
1	RXERR	**Receive Error (Read Only).** Set when an error condition has occurred with the reception of a SETUP or OUT transaction. Complete or partial data has been written into the receive FIFO. No handshake is returned. The error can be one of the following: ■ Data failed CRC check. ■ Bit stuffing error. ■ A receive FIFO goes into overrun or underrun condition while receiving. This bit is updated by hardware at the end of a valid SETUP or OUT token transaction (nonisochronous) or at the next SOF on each valid OUT token transaction (isochronous). These conditions also cause the corresponding FRXDx bit of SBI or SBI1 to be set. RXERR is updated with the RXACK bit at the end of data reception. RXERR and RXACK are updated at the same time—one bit is set to 1, and the other is reset to 0.
0	RXACK	**Receive Acknowledge (Read Only).** This bit is set when an ACK handshake is sent in response to data being written to the receive FIFO. This read-only bit is updated by hardware at the end of a valid SETUP or OUT token transaction (nonisochronous) or at the next SOF on each valid OUT token transaction (isochronous). This condition also causes the corresponding FRXDx bit of SBI or SBI1 to be set. RXACK is updated with the RXERR bit at the end of data reception. RXERR and RXACK are updated at the same time—one bit is set to 1, and the other is reset to 0.

* For normal operation, this bit should not be modified by the user except as required by the implementation of USB standard commands, such as SET_CONFIGURATION, SET_INTERFACE, and CLEAR_FEATURE [stall]. The SIE handles all sequence bit tracking required by normal USB traffic, as documented in the USB specification, Section 8.6.

† S = shared bit. P = PEND must be set when writing this bit. See Section 7.14.12.6.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Function Address Register (FADDR)—Address: 0x64018040; Default: 0000 0000B

This SFR holds the address for the USB function. During bus enumeration, it is written by firmware with a unique value assigned by the host. If MCSR.FEAT = 1, this register is reset to 0 if a USB reset is detected.

Table 7.14-21 Function Address Register (FADDR)—Address: 0x64018040; Default: 0000 0000B

Bit	7	6	5	4	3	2	1	0
Name	RSVD	A6	A5	A4	A3	A2	A1	A0
Read/Write	—	R/W						

Bit	Name	Function/Description
7	RSVD	**Reserved.** Write 0 to this bit. Reads always return 0.
6—0	A[6:0]	**7-Bit Programmable Function Address.** This register is written by firmware as a result of commands received via endpoint 0.

Transmit FIFO Data Register (TXDAT)—Address: 0x64018000; Default: 0000 0000B

Data to be transmitted by the FIFO specified by EPINDEX is first written to this register. This register is endpoint indexed. TXDAT must not be written if TXFIF = 11.

Table 7.14-22 Transmit FIFO Data Register (TXDAT)—Address: 0x64018000; Default: 0000 0000B

Bit	7	6	5	4	3	2	1	0
Name	TXDAT7	TXDAT6	TXDAT5	TXDAT4	TXDAT3	TXDAT2	TXDAT1	TXDAT0
Read/Write	W							

Bit	Name	Function/Description
7—0	TXDAT[7:0]	**Transmit Data Byte (Write Only).** To write data to the transmit FIFO, write to this register. The write pointer is incremented automatically after a write.

Transmit FIFO Byte-Count High and Low Registers (TXCNTH, TXCNTL)—Address: TXCNTH = 0x64018008, TXCNTL = 0x64018004; Default: TXCNTH = 0000 0000B; TXCNTL = 0000 0000B

Written by firmware to indicate the number of bytes just written to the transmit FIFO specified by EPINDEX. This register is endpoint indexed. TXCNTL should be written after TXCNTH. TXCNTL write increments TXFIF, validating the data set just written.

Table 7.14-23 Transmit FIFO Byte-Count High and Low Registers (TXCNTH, TXCNTL)—Address: TXCNTH = 0x64018008, TXCNTL = 0x64018004; Default: TXCNTH = 0000 0000B; TXCNTL = 0000 0000B

Bit	15—10	9	8
Name	RSVD	BC9	BC8
Read/Write	—	R/W	

Bit	7	6	5	4	3	2	1	0
Name	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
Read/Write	R/W							

Bit	Name	Function/Description
15—10	RSVD	**Reserved.** Write 0s to these bits. Reads always return 0s.
9—0	BC[9:0]	**Transmit Byte Count (Write, Conditional Read).** 10-bit, ring buffer. These bits store transmit byte count (TXCNT).

Note: To send a status stage after a control write, no data control command, or a null packet, write 0 to TXCNT.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

USB Transmit FIFO Control Register (TXCON)—Address: 0x6401800C; Default: 0000 0100B

This register controls the transmit FIFO specified by EPINDEX. This register is endpoint indexed.

Table 7.14-24 USB Transmit FIFO Control Register (TXCON)—Address: 0x6401800C; Default: 0000 0100B

Bit	7	6	5	4	3	2	1	0
Name	TXCLR	FFSZ1	FFSZ0	RSVD	TXISO	ATM	ADVRM	REVRP
Read/Write	R/W			—	R/W			

Bit	Name	Function/Description
7	TXCLR	**Transmit FIFO Clear.** Setting this bit flushes the transmit FIFO, resets all the read/write pointers and markers, resets the TXCNTH and TXCNTL registers, resets the TXFLUSH, TXVOID, TXERR, and TXACK bits of the TXSTAT register, sets the TXEMP bit in TXFLG, and clears all other bits in TXFLG. Hardware clears this bit after the flush. Setting this bit does not affect the TXSEQ bit in the TXSTAT register. This bit should only be set when the endpoint is known to be inactive or there is a FIFO error present.
6—5	FFSZ[1:0]	**FIFO Size.** These bits select the size of the transmit FIFO.
4	RSVD	**Reserved.** Write 0 to this bit. Reads always return 0.
3	TXISO	**Transmit Isochronous Data.** Firmware sets this bit to indicate that the transmit FIFO contains isochronous data. The SIE uses this bit to determine if a handshake is required at the end of a transmission.
2	ATM	**Automatic Transmit Management.*** Setting this bit (the default value) causes the read pointer and read marker to be adjusted automatically as indicated:.
1	ADVRM	**Advance Read Marker Control (Non-ATM Mode Only).†** Setting this bit prepares for the next packet transmission by advancing the read marker to the origin of the next data packet (the position of the read pointer). Hardware clears this bit after the read marker is advanced. This bit is effective only when the REVRP, ATM, and TXCLR bits are clear.
0	REVRP	**Reverse Read Pointer (Non-ATM Mode Only).†** In the case of a bad transmission, the same data stack may need to be available for retransmit. Setting this bit reverses the read pointer to point to the origin of the last data set (the position of the read marker) so that the SIE can reread the last set for retransmission. Hardware clears this bit after the read pointer is reversed. This bit is effective only when the ADVRM, ATM, and TXCLR bits are all clear.

FFSZ[1:0]	Nonisochronous Size	Isochronous Size
00	16	64
01	64	256
10	8*	512
11	32*	1024

Status	Read Pointer	Read Marker
ACK	Unchanged	Advanced (1)
NACK	Reversed (2)	Unchanged

1. To origin of next data set.
2. To origin of the data set last read.

This bit should always be set, except for test purposes. Setting this bit disables ADVRM and REVRP. This bit can be set and cleared by firmware. Hardware neither clears nor sets this bit. This bit must always be set for isochronous endpoints (TXISO = 1).

* ATM mode is recommended for normal operation. ADVRM and REVRP, which control the read marker and read pointer when ATM = 0, are used for test purposes.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Transmit FIFO Flag Register (TXFLG)—Address: 0x64018010; Default: 0000 1000B

These flags indicate the status of data packets in the transmit FIFO specified by EPINDEX. This register is end-point indexed.

Table 7.14-25 Transmit FIFO Flag Register (TXFLG)—Address: 0x64018010; Default: 0000 1000B

Bit	7	6	5—4	3	2	1	0
Name	TXFIF1	TXFIF0	RSVD	TXEMP	TXFULL	TXURF	TXOVF
Read/Write	R		—	R		R/W	

Bit	Name	Function/Description
7—6	TXFIF[1:0]	**Transmit FIFO Index Flags (Read Only).** These flags indicate that data sets are present in the transmit FIFO (see below).

Data Sets Present

TXFIF[1:0]	ds1	ds0	Status
00	No	No	Empty
01	No	Yes	1 set
10	Yes	No	1 set
11	Yes	Yes	2 sets

The TXFIF bits are set in sequence after each write to TXCNT to reflect the addition of a data set. Likewise, the TXFIF1 and TFIF0 are cleared in sequence after each advance of the read marker to indicate that the set is effectively discarded. The bit is cleared whether the read marker is advanced by firmware (setting ADVRM) or automatically by hardware (ATM = 1). The next-state table for the TXFIF bits is shown below:

Data Sets Present

TXFIF[1:0]	Operation	Next TXFIF[1:0]
00	Write TXCNT	01
01	Write TXCNT	11
10	Write TXCNT	11
11	Write TXCNT	11 (TXOVF = 1)
00	Advance Read Marker	00
01	Advance Read Marker	00
11	Advance Read Marker	10/01
10	Advance Read Marker	00
XX	Reverse Read Pointer	Unchanged

In isochronous mode, TXOVF, TXURF, and TXFIF are handled using the following rule: firmware events cause status change immediately, while USB events cause status change only at SOF. TXFIF is incremented by firmware and decremented by the USB. Therefore, writes to TXCNT increment TXFIF immediately. However, a successful USB transaction any time within a frame decrements TXFIF only at SOF.

The TXFIF flags must be checked before and after writes to the transmit FIFO and TXCNT for traceability. See the TXFLUSH bit in TXSTAT.

If MCSR.FEAT = 0:

TXFIF bits are immediately visible to the host after a firmware write—the device will send the indicated data set(s) to the host in response to an IN.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Table 7.14-25 Transmit FIFO Flag Register (TXFLG)—Address: 0x64018010; Default: 0000 1000B (continued)

Bit	Name	Function/Description
7—6	TXFIF[1:0]	**Transmit FIFO Index Flags (Read Only)** (continued). If MCSR.FEAT = 1: TXFIF bits are not visible to the host until the first SOF is written, which occurs after the data set. Prior to that SOF, the device will return a zero-length data set in response to an IN (unless there is another, older data set present from the prior frame). This ensures that a given data set may only be sent during the subsequent frame, as required by the USB specification. This behavior also allows firmware to occasionally be late in writing a data set (write complete after SOF), without losing frame/data synchronization with the host. The late data set write will cause a zero-length data set to be sent to the host during the intended frame. The late set will be flushed at the end of the next frame, assuming firmware also writes the correct data set during that frame (see TXSTAT.TXFLUSH description). Firmware must not be late on consecutive frames (this will cause a loss of frame/data synchronization with the host), data sets may be sent during the wrong frame. **Note:** Firmware can enforce single-packet mode by only writing a new data set to the transmit FIFO if there are currently no data sets present in the FIFO (TXFIF = 00). To simplify firmware development, configure control endpoints in single-packet mode.
5—4	RSVD	**Reserved.** Write 0s to these bits. Reads always return 0s.
3	TXEMP	**Transmit FIFO Empty Flag (Read Only).** Hardware sets this bit when firmware has not yet written any data bytes to the current FIFO data set being written. Hardware clears this bit when the empty condition no longer exists. This bit always tracks the current transmit FIFO status regardless of isochronous or nonisochronous mode.
2	TXFULL	**Transmit FIFO Full Flag (Read Only).** Hardware sets this bit when the number of bytes that firmware writes to the current transmit FIFO data set equals the FIFO size. Hardware clears this bit when the full condition no longer exists. This bit always tracks the current transmit FIFO status regardless of isochronous or nonisochronous mode. Check this bit to avoid causing a TXOVF condition.
1	TXURF	**Transmit FIFO Underrun Flag (Read, Clear Only).** Hardware sets this flag when a read is attempted from an empty transmit FIFO. (This is caused when the value written to TXCNT is greater than the number of bytes written to TXDAT.) This bit must be cleared by firmware through TXCLR. When this flag is set, the FIFO is in an unknown state; therefore, it is recommended that the FIFO is reset in the error management routine using the TXCLR bit in TXCON. When the transmit FIFO underruns, the read pointer does not advance; it remains locked in the empty position. When this bit is set, all transmissions are NACKed. In isochronous mode, TXOVF, TXURF, and TXFIF are handled using the following rule: firmware events cause status change immediately, while USB events cause status change only at SOF. Since underrun can only be caused by USB, TXURF is updated at the next SOF regardless of where the underrun occurs in the frame.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Table 7.14-25 Transmit FIFO Flag Register (TXFLG)—Address: 0x64018010; Default: 0000 1000B (continued)

Bit	Name	Function/Description
0	TXOVF	**Transmit FIFO Overrun Flag (Read, Clear Only).** This bit is set when an additional byte is written to a full FIFO, or TXCNT is written while TXFIF[1:0] = 11. This bit must be cleared by firmware through TXCLR. When this bit is set, the FIFO is in an unknown state; thus, it is recommended that the FIFO is reset in the error management routine using the TXCLR bit in TXCON. When the transmit FIFO overruns, the write pointer does not advance; it remains locked in the full position. Check this bit after loading the FIFO prior to writing the byte count register. When this bit is set, all transmissions are NACKed. In isochronous mode, TXOVF, TXURF, and TXFIF are handled using the following rule: firmware events cause status change immediately, while USB events cause status change only at SOF. Since overrun can only be caused by firmware, TXOVF is updated immediately. Check the TXOVF flag after writing to the transmit FIFO before writing to TXCNT.

Receive FIFO Data Register (RXDAT)—Address: 0x64018014; Default: 0000 0000B

Receive FIFO data specified by EPINDEX is stored and read from this register. This register is endpoint indexed.

Table 7.14-26 Receive FIFO Data Register (RXDAT)—Address: 0x64018014; Default: 0000 0000B

Bit	7	6	5	4	3	2	1	0
Name	RXDAT[7:0]							
Read/Write	R							

Bit	Name	Function/Description
7—0	RXDAT[7:0]	**Receive FIFO Data Register (Read Only).** To write to the receive FIFO, the SIE writes to this register. To read data from the receive FIFO, the CPU reads from this register. The write pointer and read pointer are incremented automatically after a write and read, respectively. The EPINDEX register must not be changed during a sequence of RXDAT reads of a particular data set.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Receive FIFO Byte-Count High and Low Registers (RXCNTH, RXCNTL)—Address: RXCNTH = 0x6401801C, RXCNTL = 0x64018018; Default: RXCNTH = 0000 0000B, RXCNTL = 0000 0000B

High and low registers are in a two-register ring buffer that is used to store the byte count for the data packets received in the receive FIFO specified by EPINDEX. These registers are endpoint indexed.

Table 7.14-27 Receive FIFO Byte-Count High and Low Registers (RXCNTH, RXCNTL)—Address: RXCNTH = 0x6401801C, RXCNTL = 0x64018018; Default: RXCNTH = 0000 0000B, RXCNTL = 0000 0000B

Bit	15—10	9	8
Name	RSVD	BC9	BC8
Read/Write	—	R	

Bit	7	6	5	4	3	2	1	0
Name	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
Read/Write	R							

Bit	Name	Function/Description
15—10	RSVD	**Reserved.** Write 0s to these bits. Reads always return 0s.
9—0	BC[9:0]	**Receive Byte Count (Read Only).** 10-bit, ring buffer byte. Stores receive byte count (RXCNT).

Receive FIFO Control Register (RXCON)—Address: 0x64018020; Default: 0000 0100B

Controls the receive FIFO specified by EPINDEX. This register is endpoint indexed.

Table 7.14-28 Receive FIFO Control Register (RXCON)—Address: 0x64018020; Default: 0000 0100B

Bit	7	6	5	4	3	2	1	0
Name	RXCLR	FFSZ1	FFSZ0	RXFFRC	RXISO	ARM	ADVWM	REVWP
Read/Write	R/W							

Bit	Name	Function/Description
7	RXCLR	**Receive FIFO Clear.** Setting this bit flushes the receive FIFO, resets all the read/write pointers and markers, resets the RXSETUP, STOVW, EDOVW, RXVOID, RXERR, and RXACK bits of the RXSTAT register, sets the RXEMP bit in RXFLG register, and clears all other bits in RXFLG register. Hardware clears this bit when the flush operation is completed. Setting this bit does not affect the RXSEQ bit of RXSTAT. This bit should only be set when the endpoint is disabled or there is a FIFO error present. Firmware should never set this bit to clear a SETUP packet. The next SETUP packet will automatically clear the receive FIFO.
6—5	FFSZ[1:0]	**FIFO Size.** These bits select the size of the receive FIFO. **FFSZ[1:0] — Nonisochronous Size — Isochronous Size** 00 — 16 — 64 01 — 64 — 256 10 — 8* — 512 11 — 32* — 1024

* Assumes MCSR.FEAT = 1. If MCSR.FEAT = 0, these FFSZ settings indicate 64 bytes.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Table 7.14-28 Receive FIFO Control Register (RXCON)—Address: 0x64018020; Default: 0000 0100B (continued)

<table>
<tr><th>Bit</th><th>Name</th><th>Function/Description</th></tr>
<tr><td>4</td><td>RXFFRC</td><td>FIFO Read Complete. When set, the receive FIFO is released when a data set read is complete. Setting this bit clears the RXFIF bit (in the RXFLG register), corresponding to the data set that was just read. Hardware clears this bit after the RXFIF bit is cleared. All data from this data set must have been read. For isochronous endpoints, firmware must check RXFLUSH before setting RXFFRC, and the act of setting RXFFRC clears RXFLUSH. See RXFLUSH description for details.

Note: FIFO read complete only works if the STOVW and EDOVW bits are both cleared.</td></tr>
<tr><td>3</td><td>RXISO</td><td>Receive Isochronous Data. When set, this indicates that the receive FIFO is programmed to receive isochronous data and to set up the USB interface to handle an isochronous data transfer.</td></tr>
<tr><td>2</td><td>ARM</td><td>Auto Receive Management.* When set, the write pointer and write marker are adjusted automatically based on the following conditions:.

<table>
<tr><th>Rx Status</th><th>Write Pointer</th><th>Write Marker</th></tr>
<tr><td>ACK</td><td>Unchanged</td><td>Advanced</td></tr>
<tr><td>NACK</td><td>Reversed</td><td>Unchanged</td></tr>
</table>

This bit should always be set, except for test purposes. When this bit is set, setting REVWP or ADVWM has no effect. Hardware neither clears nor sets this bit. This bit can be set and cleared by firmware. This bit must always be set for isochronous endpoints (RXISO = 1).</td></tr>
<tr><td>1</td><td>ADVWM</td><td>Advance Write Marker (Non-ARM Mode Only).* When set, the write marker is advanced to the origin of the next data set. Advancing the write marker is used for back-to-back receptions. Hardware clears this bit after the write marker is advanced. Setting this bit is effective only when the REVWP, ARM, and RXCLR bits are clear.</td></tr>
<tr><td>0</td><td>REVWP</td><td>Reverse Write Pointer (Non-ARM Mode Only).* When set, the write pointer is returned to the origin of the last data set received, as identified by the write marker. The SIE can then reread the last data packet and write to the receive FIFO starting from the same origin when the host resends the same data packet. Hardware clears this bit after the write pointer is reversed. Setting this bit is effective only when the ADVWM, ARM, and RXCLR bits are clear.

REVWP is used when a data packet is bad. When the function interface receives the data packet again, the write starts at the origin of the previous (bad) data set.</td></tr>
</table>

* *ARM* mode is recommended for normal operation. ADVWM and REVWP, which control the write marker and write pointer when *ARM* = 0, are used for test purposes.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Receive FIFO Flag Register (RXFLG)—Address: 0x64018024; Default: 0000 1000B

These flags indicate the status of the data packets in the receive FIFO specified by EPINDEX. This register is endpoint indexed.

Table 7.14-29 Receive FIFO Flag Register (RXFLG)—Address: 0x64018024; Default: 0000 1000B

Bit	7	6	5	4	3	2	1	0
Name	RXFIF1	RXFIF0	RSVD	RXFLUSH	RXEMP	RXFULL	RXURF	RXOVF
Read/Write	R		—	R	R		R/W	

Bit	Name	Function/Description
7—6	RXFIF[1:0]	**Receive FIFO Index Flags (Read Only).** These read-only flags indicate that data packets are present in the receive FIFO (see below).

Data Sets Present

RXFIF[1:0]	ds1	ds0	Status
00	No	No	Empty
01	No	Yes	1 set
10	Yes	No	1 set
11	Yes	Yes	2 sets

The RXFIF bits are updated after each write to RXCNT to reflect the addition of a data packet. Likewise, the RXFIF bits are cleared in sequence after each setting of the RXFFRC bit. The next-state table for RXFIF bits is shown below for operation in dual-packet mode.

RXFIF[1:0]	Operation	Next RXFIF[1:0]
00	Advance Write Marker	01
01	Advance Write Marker	11
10	Advance Write Marker	11
11	Advance Write Marker Not Possible—Device will NACK any OUT.	11
00	Set RXFFRC	00
01	Set RXFFRC	00
11	Set RXFFRC	10/01
10	Set RXFFRC	00
00	Reverse Write Pointer	Unchanged

When the receive FIFO is programmed to operate in single-packet mode (RXSPM set in EPCON), valid RXFIF states are 00 and 01 only.

In isochronous mode, RXOVF, RXURF, and RXFIF are handled using the following rule: firmware events cause status change immediately, while USB events cause status change only at SOF. RXFIF is incremented by the USB and decremented by firmware. Therefore, setting RXFFRC decrements RFIF immediately. However, a successful USB transaction within a frame increments RXFIF only at SOF.

If MCSR.FEAT = 1:

An old data set is flushed from an isochronous FIFO if it is not read out by firmware during the intended frame (see RXFLG.RXFLUSH description). This flush occurs at SOF, sets RXFLG.RXFLUSH, and causes RXFIF to decrement without firmware intervention.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Table 7.14-29 Receive FIFO Flag Register (RXFLG)—Address: 0x64018024; Default: 0000 1000B (continued)

Bit	Name	Function/Description
7—6	RXFIF[1:0]	**Receive FIFO Index Flags (Read Only)** (continued). For traceability, the RXFIF flags must be checked before and after reads from the receive FIFO and the setting of RXFFRC in RXCON. **Note:** To simplify firmware development, it is recommended that control endpoints are used in single-packet mode only.
5	RSVD	**Reserved.** Write 0s to these bits. Reads always return 0s.
4	RXFLUSH	**Receive FIFO Flush (Read Only).** Only available if MCSR.FEAT = 1. Updated at every SOF, and only used for isochronous endpoints. RXFIF bits are set when valid data sets are received from the host. For isochronous endpoints, this RXFIF increment does not occur until the next SOF. During that subsequent frame, it is the responsibility of firmware to read out the data set. If that read is not completed (RXFFRC set by firmware) by the time the next SOF is received, that data set is flushed from the receive FIFO—RXFIF is decremented by hardware. This flush is indicated by hardware by setting the RXFLUSH bit. While this bit is set, the affect of firmware receive FIFO data (RXDAT) reads is blocked, in order to stop potential corruption of a new data set. Before firmware sets RXFFRC (for isochronous endpoints only), it must first check RXFLUSH. If RXFLUSH is set, firmware must discard the data set that it just read, because it is potentially corrupted. This situation should only occur if firmware is late in reading out a data set (read not completed before SOF). Firmware must not be late on consecutive frames—this will cause a loss of frame/data synchronization with the host—data sets may be visible to firmware during the wrong frame. Firmware must always set RXFFRC at the end of a data set read, even if RXFLUSH = 1. RXFLUSH is reset to 0 by the setting of RXFFRC to 1.
3	RXEMP	**Receive FIFO Empty Flag (Read Only).** Hardware sets this flag when there are no data bytes present in the data set currently being read. Hardware clears the bit when the empty condition no longer exists. This bit always tracks the current status of the receive FIFO, regardless of isochronous or nonisochronous mode.
2	RXFULL	**Receive FIFO Full Flag (Read Only).** Hardware sets this flag when the data set currently being read contains the same number of data bytes as the size of the FIFO. Hardware clears the bit when the full condition no longer exists. This bit always tracks the current status of the receive FIFO regardless of isochronous or nonisochronous mode.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Table 7.14-29 Receive FIFO Flag Register (RXFLG)—Address: 0x64018024; Default: 0000 1000B (continued)

Bit	Name	Function/Description
1	RXURF	**Receive FIFO Underrun Flag (Read, Clear Only).** Hardware sets this bit when an additional byte is read from an empty receive FIFO or when RXCNTH or RXCNTL is read while RXFIF[1:0] = 00. Hardware does not clear this bit, so it must be cleared by firmware through RXCLR. When the receive FIFO underruns, the read pointer does not advance. It remains locked in the empty position. When this bit is set, all transmissions are NACKed. In isochronous mode, RXOVF, RXURF, and RXFIF are handled using the following rule: firmware events cause status change immediately, while USB events cause status change only at SOF. Since underrun can only be caused by firmware, RXURF is updated immediately. The RXURF flag must be checked after reads from the receive FIFO before setting the RXFFRC bit in RXCON. **Note:** When this bit is set, the FIFO is in an unknown state. It is recommended that the FIFO is reset in the error management routine using the RXCLR bit in the RXCON register.
0	RXOVF	**Receive FIFO Overrun Flag (Read, Clear Only).** This bit is set when the SIE writes an additional byte to a full receive FIFO or writes a byte count to RXCNT with RXFIF[1:0] = 11. This bit must be cleared by firmware through RXCLR, although it can be cleared by hardware if a SETUP packet is received after an RXOVF error has already occurred. When this bit is set, all transmissions are NACKed. In isochronous mode, RXOVF, RXURF, and RXFIF are handled using the following rule: firmware events cause status change immediately, while USB events cause status change only at SOF. Since overrrun can only be caused by the USB, RXOVF is updated only at the next SOF regardless of where the overrun occurred during the current frame. **Note:** When this bit is set, the FIFO is in an unknown state. It is recommended that the FIFO is reset in the error management routine using the RXCLR bit in the RXCON register. When the receive FIFO overruns, the write pointer does not advance. It remains locked in the full position.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

System Control Register (SCR)—Address: 0x64018044; Default: 0000 0000B

This register controls the FIFO mode, IRQ mask, and IRQ mode selection.

Table 7.14-30 System Control Register (SCR)—Address: 0x64018044; Default: 0000 0000B

Bit	7	6	5	4	3	2	1	0
Name	IRQPOL	RWUPE	IE_SUSP	IE_RESET	SRESET	IRQLVL	T_IRQ	RSVD
Read/Write	R/W							—

Bit	Name	Function/Description
7	IRQPOL	**IRQ Polarity.** Determines the polarity of the IRQ29 output. When asserted, the IRQ29 output is active-high (default is active-low). Firmware must be careful to ensure that setting this bit does not cause a false interrupt to be detected and processed.
6	RWUPE	**Enable Remote Wake-Up Feature.** When set, remote wake-up is enabled.
5	IE_SUSP	**Enable Suspend Interrupt.** When set, the SUSPEND interrupt is enabled.
4	IE_RESET	**Enable Reset Interrupt.** When set, the RESET interrupt is enabled.
3	SRESET	**Software Reset.** Setting this bit to 1 in software places the USS820core in the RESET state. This is equivalent to asserting the hardware RESET pin, except that this feature is not available if the device is suspended. Setting this bit back to 0 leaves the USS820core in an unconfigured state that follows a hardware reset. If MCSR.FEAT = 1, SSR.SUPPO = 0 and MCSR.SUSPLOE = 0: This bit may also be set to 1 while the device is suspended. The effect of this write is to wake up the device as if a remote wake-up had been performed, with the following exceptions: 1) Resume signaling is not transmitted to the host, 2) The feature is enabled regardless of the SCR.RWUPE setting, and 3) The MCSR.RWUPR register bit does not set. The actual setting of the SCR.SRESET register bit does not occur until the device is resumed and internal clocks are enabled, but the wake-up is initiated immediately. Once the wake-up is complete, the SRESET bit sets, and the behavior is the same as if SRESET had been set while the device was awake.
2	IRQLVL	**Interrupt Mode.** Level mode interrupt is selected when this bit is cleared. Pulse mode interrupt is selected when this bit is set. In pulse mode, IRQ29 signal is driven (high or low, depending on the IRQPOL setting) by USS820core for two t_{CLK} periods.
1	T_IRQ	**Global Interrupt Enable.** When this bit is set, it enables hardware interrupt to be generated on IRQ29 signal when any of Tx/Rx bits, ASOF bit, RESET bit, or SUSPEND bit is set.
0	RSVD	**Reserved.** Write 0 to this bit. Reads always return 0.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

System Status Register (SSR)—Address: 0x64018048; Default: 0000 0000B

This register allows control and monitoring of the USB suspend and reset events.

Table 7.14-31 System Status Register (SSR)—Address: 0x64018048; Default: 0000 0000B

Bit	7—5	4	3	2	1		0
Name	RSVD	SUSPPO	SUSPDIS	RESUME	SUSPEND		RESET
Read/Write	—	R/W (P*)			R	W (P*)	R/W (S*)

Bit	Name	Function/Description
7—5	RSVD	**Reserved.** Write 0s to these bits. Reads always return 0s.
4	SUSPPO	**Suspend Power Off.** This bit must be set by firmware if externally connected devices will be powered off during a suspend. The correct value of this bit must be established before firmware suspends the USS820core and should only need to be done once at device initialization time. Since T8307 is a self-powered USB device, this bit should be set to 0.
3	SUSPDIS	**Suspend Disable.** When asserted, this bit disables the detection of a USB suspend event. This bit is for test purposes and should not be set during normal system operation.
2	RESUME	**Resume Detected.** For a complete description of the use of this bit, see Section 7.14.9 will When set, the USS820core has detected and responded to a wake-up condition, either global or remote. A global resume is indicated when the host asserts a non-IDLE state on the USB bus. A remote wake-up is indicated when the RWUPN input is asserted (if that feature is enabled by the RWUPE bit). This bit should be reset by firmware as soon as possible after resuming to allow the next suspend event to be detected.
1	SUSPEND	**Suspend Detected (Read Only)/Suspend Control (Write Only).** For a complete description of the use of this bit, see Section 7.14.9. This bit serves as both a read-only status bit and a write-only control bit. For this reason, firmware cannot do a simple read/modify/write sequence to update this register. Firmware must always explicitly specify the correct value of this SUSPEND control bit when writing SSR. The read-only status bit is set by hardware when a SUSPEND condition is detected on the USB bus, and clears itself after the SUSPEND condition ceases and the device resumes. The bit will remain set during device wake-up. The value of this read-only bit is not affected by firmware writes. The write-only control bit is only updated by firmware, and is used to suspend the device by setting the bit to 1, and then setting the bit to 0. This write sequence will cause the device to suspend regardless of the initial value of the bit, which cannot be read.
0	RESET	**USB Reset Detected.** When set, a RESET condition is detected on the USB bus. If interrupt is enabled (T_IRQ and IE_RESET set), an interrupt is generated to the controller. Firmware clears this bit.

* S = shared bit. P = PEND must be set when writing this bit. See Section 7.14.12.6.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Hardware Revision Register (REV)—Address: 0x64018060; Default: 0001 0100B

This register contains the hardware revision number, which will be incremented for each version of the USS820core. This will allow firmware to query the hardware status and determine which functions or features are supported. For USS820core in T8307, this hardware revision number is 0x14.

Table 7.14-32 Hardware Revision Register (REV)—Address: 0x64018060; Default: 0001 0100B

Bit	7	6	5	4	3	2	1	0
Name	Main Hardware Revision Number				Sub Hardware Revision Number			
Read/Write	R							

Bit	Name	Function/Description
7—4	—	**Main Hardware Revision Number.**
3—0	—	**Sub Hardware Revision Number.**

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Suspend Power-Off Locking Register (LOCK)—Address: 0x64018064; Default: 0000 0001B

This register contains the control and status which enables the USS820core locking mechanism. This feature protects the internal register set from being corrupted during and immediately after a suspend where the external controller is powered off. The feature is enabled by the SUSPLOE bit, and its proper usage is documented in the *Special Action Required by USS-820D/USS-825 After Suspend* Application Note (AP97-058CMPR-04).

Table 7.14-33 Suspend Power-Off Locking Register (LOCK)—Address: 0x64018064; Default: 0000 0001B

Bit	7—1	0
Name	RSVD	UNLOCKED
Read/Write	—	R/W

Bit	Name	Function/Description
7—1	RSVD	**Reserved.**
0	UNLOCKED	**Locking Control/Status.** Use of this bit is described in the *Special Action Required by USS-820D/USS-825 After Suspend* Application Note (AP97-058CMPR-04).

Pend Hardware Status Update Register (PEND)—Address: 0x64018068; Default: 0000 0000B

This register contains the PEND bit.

Table 7.14-34 Pend Hardware Status Update Register (PEND)—Address: 0x64018068; Default: 0000 0000B

Bit	7—1	0
Name	RSVD	PEND
Read/Write	—	R/W

Bit	Name	Function/Description
7—1	RSVD	**Reserved.**
0	PEND	**Pend.** When set, this bit modifies the behavior of other shared register bits. See Section 7.14.12.6 for a detailed explanation.

Scratch Firmware Information Register (SCRATCH)—Address: 0x6401806C; Default: 0000 0000B

This register contains a 6-bit scratch field that can be used by firmware to save and restore information. One possible use would be to save the device's USB state (e.g., DEFAULT, ADDRESSED) during suspend power off. The register also contains the resume interrupt enable bit.

Table 7.14-35 Scratch Firmware Information Register (SCRATCH)—Address: 0x6401806C; Default: 0000 0000B

Bit	7	6	5	4	3	2	1	0
Name	IE_RESUME	RSVD	SCRATCH					
Read/ Write	R/W	R/W	R/W					

Bit	Name	Function/Description
7	IE_RESUME	**Enable Resume Interrupt.** When set, the RESUME interrupt is enabled.
6	RSVD	**Reserved for Product Test.** Always write 0 to this bit.
5—0	SCRATCH	**Scratch Information.**

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Miscellaneous Control/Status Register (MCSR)—Address: 0x64018070; Default: 0000 0000B

This register contains miscellaneous control and status bits.

Table 7.14-36 Miscellaneous Control/Status Register (MCSR)—Address: 0x64018070; Default: 0000 0000B

Bit	7	6	5	4	3	2	1	0
Name	RWUPR	INIT	SUSPS	RSVD	FEAT	BDFEAT	SUSPLOE	RSVD
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W

Bit	Name	Function/Description
7	RWUPR	**Remote Wake-Up Remember.** This bit is only available if MCSR.FEAT = 1; otherwise, it always reads 0. Updated by hardware on each wake-up from a suspended state. This bit is set to 1 if the wake-up was caused by a remote wake-up event (RWUPN input signal asserted). Otherwise, it is reset to 0 (on a global resume or USB reset). If RWUPN is asserted simultaneously with a global wake-up, the bit is reset to 0 (global wake-up wins). When set, this bit indicates that resume signaling will be transmitted upstream.
6	INIT	**Device Initialized.** This bit will read 0 until internal clocks are turned on after a hardware reset. This bit is not affected by software reset. This bit can be used by firmware to determine when the device is operational after a hardware reset.
5	SUSPS	**Suspend Status.** Indicates the current suspended status of the device. This bit will be set when the device goes suspended and will remain set until internal clocks are turned back on at the end of a resume sequence.
4	RSVD	**Reserved.** Read as zero.
3	FEAT	**Feature Enable.** When set, this bit enables various features introduced in revision C of the USS820C. This bit controls those features that do not impact existing circuit boards using the USS820 revision B (i.e., those features not enabled by MCSR.BDFEAT). These features are explained in detail in the Appendix C of the USS820D data sheet. When reset to 0 (along with MCSR.BDFEAT, TXSTAT.TXDSAM and TXSTAT.TXNAKE), the device will behave like revision B.
2	BDFEAT	**Board Feature Enable.** When set, this bit enables various features introduced in revision C of the USS820C. This bit controls those features that could be incompatible with existing circuit boards using the USS820 revision B. These features are explained in detail in the Appendix C of the USS820D data sheet. When reset to 0 (along with MCSR.FEAT, TXSTAT.TXDSAM, and TXSTAT.TXNAKE), the USS820core will behave like revision B.
1	SUSPLOE	**Suspend Lock Out Enable.** Enables the device locking mechanism, which will then engage on every device resume. The correct value of this bit must be established before firmware suspends the device.
0	RSVD	**Reserved.** Read as zero. Always write 0 when writing to this register bit.

7 Call Processor (CP) Block (continued)

7.14 USB Device Controller (USBDC) (continued)

Data Set Available (DSAV)—Address: 0x64018074; Default: 0000 0000B

This register contains receive/transmit data set available bits.

Table 7.14-37 Data Set Available (DSAV)—Address: 0x64018074; Default: 0000 0000B

Bit	7	6	5	4	3	2	1	0
Name	RXAV3	TXAV3	RXAV2	TXAV2	RXAV1	TXAV1	RXAV0	TXAV0
Read/Write	R	R	R	R	R	R	R	R

Bit	Name	Function/Description
7	RXAV3	**Receive/Transmit Data Set Available.** This feature is only available if MCSR.FEAT = 1 or TXDSAM = 1; otherwise, reads 0. May be used to improve firmware efficiency when polling endpoints. For receive FIFOs, this register indicates that one or more data sets are available to be read. For transmit FIFOs, this register indicates that one or more data sets are available to be written. Bits always read 0 for endpoints that are not enabled (RXEPEN/TXEPEN = 0). If a transmit endpoint has TXDSAM = 1, the corresponding RXAV/TXAV bit of the DSAV register indicates instead that the TXVOID bit is set (a NAK has been sent to the host). This usage when TXDSAM = 1 does not require MCSR.FEAT = 1.
6	TXAV3	
5	RXAV2	
4	TXAV2	
3	RXAV1	
2	TXAV1	
1	RXAV0	
0	TXAV0	

Data Set Available (DSAV1)—Address: 0x64018078; Default: 0000 0000B

This register contains receive/transmit data set available bits.

Table 7.14-38 Data Set Available (DSAV1)—Address: 0x64018078; Default: 0000 0000B

Bit	7	6	5	4	3	2	1	0
Name	RXAV7	TXAV7	RXAV6	TXAV6	RXAV5	TXAV5	RXAV4	TXAV4
Read/Write	R	R	R	R	R	R	R	R

Bit	Name	Function/Description
7	RXAV7	**Receive/Transmit Data Set Available.** This feature is only available if MCSR.FEAT = 1 or TXDSAM = 1; otherwise, reads 0. May be used to improve firmware efficiency when polling endpoints. For receive FIFOs, this register indicates that one or more data sets are available to be read. For transmit FIFOs, this register indicates that one or more data sets are available to be written. Bits always read 0 for endpoints that are not enabled (RXEPEN/TXEPEN = 0). If a transmit endpoint has TXDSAM = 1, the corresponding RXAV/TXAV bit of the DSAV register indicates instead that the TXVOID bit is set (a NAK has been sent to the host). This usage when TXDSAM = 1 does not require MCSR.FEAT = 1.
6	TXAV7	
5	RXAV6	
4	TXAV6	
3	RXAV5	
2	TXAV5	
1	RXAV4	
0	TXAV4	

7 Call Processor (CP) Block (continued)

7.15 Pin Multiplexer (PMUX) Module

The PMUX module controls the multiplexing of multi-functional pins and the enabling of pull-up resistors on various T8307 pins. The PMUX module also contains the *ARM* ID register.

7.15.1 ALTPIN Control Clear Register (ALTPINC Clear)

The ALTPIN control clear register (see Table 7.15-1) provides an easy way to clear individual bits in the ALTPIN control register.

Table 7.15-1 ALTPIN Control Clear Register (ALTPINC Clear), Address (0x700CF000)

Bit	31—14	13—0
Name	RSVD	ALTPIN_CNTL_CLR[13:0]

Bit	Name	Description
31—14	RSVD	Reserved—write with all ones.
13—0	ALTPIN_ CNTL_CLR [13:0]	ALTPIN control clear. For each bit when written: 1—The corresponding bit in the ALTPIN control register is cleared. 0—The corresponding bit in the ALTPIN control register is unchanged. When read: The current value of the ALTPIN control register is returned.

7.15.2 ALTPIN Control Set Register (ALTPINC Set)

The ALTPIN control set register (see Table 7.15-2) provides an easy way to set individual bits in the ALTPIN control register.

Table 7.15-2 ALTPIN Control Set Register (ALTPINC Set), Address (0x700CF004)

Bit	31—14	13—0
Name	RSVD	ALTPIN_CNTL_SET[11:0]

Bit	Name	Description
31—14	RSVD	Reserved—write all 0s.
13—0	ALTPIN_ CNTL_SET [11:0]	ALTPIN control set. For each bit when written: 1—The corresponding bit in the ALTPIN control register is set. 0—The corresponding bit in the ALTPIN control register is unchanged. When read: The current value of the ALTPIN control register is returned.

7 Call Processor (CP) Block (continued)

7.15 Pin Multiplexer (PMUX) Module (continued)

7.15.3 ALTPIN Control Register (ALTPINC)

The ALTPIN control register (see Table 7.15-3) allows the software to control the function of T8307 pins that have multiple functions multiplexed onto them.

Table 7.15-3 ALTPIN Control Register (ALTPINC), Address (0x700CF008)

Bit	31—14	13—0
Name	RSVD	ALTPIN_CNTL[13:0]

Bit	Name	Description
31—14	RSVD	Reserved—write with 0.
13—0	ALTPIN_ CNTL[13:0]	ALTPIN control. For each bit when written: 1—The alternate functions for the pin(s) are enabled. 0—The default functions for the pin(s) are enabled. When read: The current value is returned. See Table 7.15-4 for the description of which bit controls which T8307 pin(s).

Table 7.15-4 ALTPIN (MUX Control) Blocks*

Block (Default/Alternate)	Pin†	Direction (Default/Alternate)	Control
UART0/PPI	RI0_PIO01	O/IO	ALTPIN_CNTL[0]
	DSR0_PIO02		
	DTR0_PIO03	I/IO	
	RTS0_PIO04		
	CTS0_PIO29	IO/IO	
	DCD0_PIO44		
DSP Bit IO/PPI	IOBIT0_PIO05	IO/IO	ALTPIN_CNTL[1]
	IOBIT1_PIO06		
PPI/USB (5 pins)	PIO07_USB_SUSP	IO/O	ALTPIN_CNTL[2]
	PIO36_USB_OEN		
	PIO12_USB_VPO	IO/IO	
	PIO13_USB_VMO		
	PIO37_USB_DATA	IO/I	

* (TBR) ALTPIN_CNTL[8] and ALTPIN_CNTL[13] are used for MMC/SD I/F fix. Set both to 1 (or 0 if can't pass the test)
† PIO00_IRQ5 and PIO31_IRQ6 are not multiplexed pins. Their input signals are always connected to both the PPI input and the IRQ input.

7 Call Processor (CP) Block (continued)

7.15 Pin Multiplexer (PMUX) Module (continued)

Table 7.15-4 ALTPIN (MUX Control) Blocks (continued)

Block (Default/Alternate)	Pin*	Direction (Default/Alternate)	Control
PPI/SD Card Interface	PIO08_MCI_CMD	IO/IO	ALTPIN_CNTL[3]
	PIO21_MCI_DAT0		
	PIO22_MCI_DAT1		
	PIO23_MCI_DAT2		
	PIO24_MCI_DAT3		
	PIO38_MCI_CLK	IO/O	
	PIO39_MCI_CMD_EN		
	PIO40_MCI_DAT_EN		
	PIO43_MCI_DAT0_EN		
DSP-Side SSPI^{2}S/PPI	SPCLK1_PIO18	IO/IO	ALTPIN_CNTL[4]
	SPRXD1_PIO17		
	SPTXD1_I2SD_PIO16		
	SPFS1_PIO15		
PPI/Reset	PIO20_SYSCLKREQ	IO/O	ALTPIN_CNTL[5]
PPI/IrDA	PIO41_IRDATX	IO/O	ALTPIN_CNTL[6]
	PIO42_IRDARX	IO/I	
PPI/SMC	PIO30_WAITN	IO/I	ALTPIN_CNTL[7]
PPI/SMC	PIO35_A_A25_BOOTSEL	IO/O	ALTPIN_CNTL[9]
PWM/PPI	PWM1_PIO46	O/IO	ALTPIN_CNTL[10]
Clock/CKO	CPTSTSTOP_CKO	I/O	ALTPIN_CNTL[11]
PPI/USB (2 pins)	PIO09_USB_VPI	IO/I	ALTPIN_CNTL[12]
	PIO11_USB_VMI		
ARM JTAG/PPI, UART1 Flow Control, and PWM2	ATMS_PIO45	I/IO	TEST1—TEST3 = 111 for *ARM* JTAG; TEST1—TEST3 = 011 or 010 for alternative functions
	ATCK_CTS1		
	ATDI_RTS1		
	ATDO_PWM2	O/O	

* PIO00_IRQ5 and PIO31_IRQ6 are not multiplexed pins. Their input signals are always connected to both the PPI input and the IRQ input.

7 Call Processor (CP) Block (continued)

7.15 Pin Multiplexer (PMUX) Module (continued)

7.15.4 *ARM* ID Register (ARMID)

The *ARM* ID register (see Table 7.15-5) allows the software to identify different mask versions of T8307.

Table 7.15-5 *ARM* ID Register (ARMID), Address (0x700CF018)

Bit	31—18	17—16	15—0
Name	TEST	VERSION ID	ROMCODE

Bit	Name	Value	Features
31—18	TEST	0x0	For test purposes only. Read as all 0s.
17—16	VERSION ID	0x0	Version identification, where the version values are listed below. Bit [17:16] / Version: 00 — T8307.; 01 — Reserved.; 10 — Reserved.; 11 — Reserved.
15—0	ROMCODE	0x0190	User's ROMCODE ID: the ROMCODE ID is the 16-bit binary value of the following expression for the initial version: *400 + (10 x value of the first letter) + (value of the second letter),* or *(20 x value of the first letter) + (value of the second letter)* for all following versions. The values of the letters are shown in the following table. T8307's ROMCODE field contains AA = 0x190, PB=0x105 and so on.

ROMCODE Letter	A	B	C	D	E	F	G	H	J	K	L	M	N	P	R	S	T	U	W	Y
Value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

7.15.5 PMUX Feature Control Register (PMUXFC)

Table 7.15-6 Feature Control Register (PMUXFC), Address (0x700CF01C)

Bit	31—8	7—2	1	0
Name	RSVD	RSVD	ACC1LWE	ACC0LWE

Bit	Name	Features
31—8	RSVD	Reserved—write with 0.
7—2	RSVD	Reserved.
1	ACC1LWE	1: Enable ACC1 Rx line wake-up interrupt generation. 0: Disable ACC1 Rx line wake-up interrupt generation.
0	ACC0LWE	1: Enable ACC0 Rx line wake-up interrupt generation. 0: Disable ACC0 Rx line wake-up interrupt generation.

7 Call Processor (CP) Block (continued)

7.15 Pin Multiplexer (PMUX) Module (continued)

7.15.6 Pull-Up Resistor Enable Control Registers (PURESEN1—3)

The three pull-up resistor enable control registers (see Table 7.15-7—Table 7.15-12) control the enabling of the pull-up resistors on various T8307 pins.

Table 7.15-7 Pull-Up Resistor Enable Control 1 Register (PURESEN1), Address (0x700CF020)

Bit	31—0
Name	PUE1[31:0]

Bit	Name	Description
31—0	PUE1[31:0]	Pull-up resistor enable 1. For each bit when written: 1—The pull-up resistor is enabled on the pin controlled by the bit. 0—The pull-up resistor is disabled on the pin controlled by the bit. When read: The current value is returned. Table 7.15-8 shows the mapping between each register bit and the T8307 pin controlled.

Table 7.15-8 Pull-Up Resistor Enable Control 1 Register to Pin Mapping

Bit	Pin	Bit	Pin
31	PIO21_MCI_DAT0	15	ATMS_PIO45
30	PIO08_MCI_CMD	14	ATCK_CTS1
29	PIO38_MCI_CLK	13	ATDI_RTS1
28	KEYBRD11	12	IOBIT1_PIO06
27	KEYBRD10	11	IOBIT0_PIO05
26	KEYBRD9	10	RSVD
25	KEYBRD8	9	RTS0_PIO04
24	KEYBRD7	8	DTR0_PIO03
23	KEYBRD6	7	D_D[15:0]*
22	KEYBRD5(PSW1_BUF)	6	CPTSTSTOP_CKO†
21	KEYBRD4	5	A_D[7:0]*
20	KEYBRD3	4	A_D[15:8]*
19	KEYBRD2	3	RSVD
18	KEYBRD1	2	RSVD
17	KEYBRD0	1	RSVD
16	DCD0_PIO44	0	RSVD

* For these pins, bus keepers (loopback circuit) are controlled by these control bits. When enabled, the last data value driven on the bus is maintained.
† This bit controls the pull-down resistor on CPTSTSTOP_CKO pin.

7 Call Processor (CP) Block (continued)

7.15 Pin Multiplexer (PMUX) Module (continued)

Table 7.15-9 Pull-Up Resistor Enable Control 2 Register (PURESEN2), Address (0x700CF024)

Bit	31—0
Name	PUE2[31:0]

Bit	Name	Description
31—0	PUE2[31:0]	Pull-up resistor enable 2. For each bit when written: 1—The pull-up resistor is enabled on the pin controlled by the bit. 0—The pull-up resistor is disabled on the pin controlled by the bit. When read: The current value is returned. Table 7.15-10 shows the mapping between each register bit and the T8307 pin controlled.

Table 7.15-10 Pull-Up Resistor Enable Control 2 Register to Pin Mapping

Bit	Pin	Bit	Pin
31	PIO33	15	PIO37_USB_DATA
30	PIO32	14	PIO36_USB_OEN
29	PIO31_IRQ6	13	PIO13_USB_VMO
28	PIO30_WAITN	12	PIO12_USB_VPO
27	CTS0_PIO29	11	PIO11_USB_VMI
26	PIO28	10	PIO10
25	PIO27	9	PIO09_USB_VPI
24	PIO26	8	PIO07_USB_SUSP
23	PIO25	7	DSR0_PIO02
22	IRQ1	6	RI0_PIO01
21	IRQ2	5	PIO43_MCI_DAT0_EN
20	IRQ3	4	PIO40_MCI_DAT_EN
19	IRQ4	3	PIO39_MCI_CMD_EN
18	PIO20_SYSCLKREQ	2	PIO24_MCI_DAT3
17	PIO19 (PWRKEEP)	1	PIO23_MCI_DAT2
16	PIO14 (SIMRST)	0	PIO22_MCI_DAT1

7 Call Processor (CP) Block (continued)

7.15 Pin Multiplexer (PMUX) Module (continued)

Table 7.15-11 Pull-Up Resistor Enable Control 3 Register (PURESEN3), Address (0x700CF028)

Bit	31—24	23—0
Name	RSVD	PUE3[23:0]

Bit	Name	Description
31—24	RSVD	Reserved—write with 0.
23—0	PUE3[23:0]	Pull-up resistor enable 3. For each bit when written: 1—The pull-up resistor is enabled on the pin controlled by the bit. 0—The pull-up resistor is disabled on the pin controlled by the bit. When read: The current value is returned. Table 7.15-12 shows the mapping between each register bit and the T8307 pin controlled.

Table 7.15-12 Pull-Up Resistor Enable Control 3 Register to Pin Mapping

Bit	Pin	Bit	Pin
23	PIO47	11	SPFS0
22	RSVD	10	SPCLK1_PIO18
21	TX1	9	SPCLK0
20	TX0	8	SIMIO
19	TEST3	7	RX1_IRQ28
18	TEST2	6	RX0_IRQ28
17	TEST1	5	PWM1_PIO46
16	SPFS1_PIO15	4	PIO00_IRQ5
15	SPTXD1_I2SD_PIO16	3	PIO42_IRDARX
14	SPTXD0_I2SD	2	PIO41_IRDATX
13	SPRXD1_PIO17	1	PIO35_A_A25_BOOTSEL
12	SPRXD0	0	PIO34

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface

SD/MMC interface module is implemented using the *ARM PrimeCell* Multimedia Card Interface (MCI, PL180). It conforms to the following standards:

- *Multimedia Card Specification v2.11*.
- *Secure Digital Memory Card Physical Layer Specification v0.96*.

SD/MMC interface module acts as either a multimedia card bus host or a secure digital memory card bus host. When acting as a multimedia card bus host.

7.16.1 Functional Description

Figure 7.16-1 shows the block diagram of SD/MMC interface module (*PrimeCell* MCI) in T8307. The *PrimeCell* MCI consists of two parts:

- The MCI adapter block provides all functions specific to the multimedia/secure digital memory card. These include the clock generation unit, power management control, command and data transfer.
- The APB interface provides access to the *PrimeCell* MCI registers, and generates interrupt and DMA request signals.

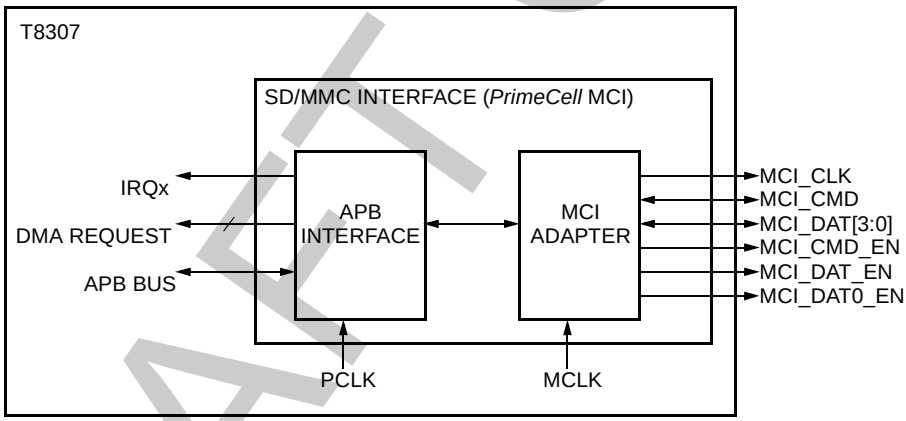


Figure 7.16-1 SD/MMC Interface (*PrimeCell* MCI) Block Diagram

The *PrimeCell* MCI uses two clock signals: *PrimeCell* MCI adapter clock (MCLK) and APB bus clock (PCLK). In T8307, MCLK is derived from PCLK, thus they have the same frequency.

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

7.16.1.1 Multimedia Card System

The multimedia card system (see Figure 7.16-2) transfers commands and data using three signal lines on a single physical bus:

- CLK: One bit is transferred on both command and data lines with each clock cycle. The clock frequency varies between 0 MHz and 20 MHz for a multimedia card.
- CMD: Bidirectional command channel that initializes a card and transfers commands. CMD has two operational modes: open-drain for initialization and push-pull for command transfer.
- DAT: Bidirectional data channel, operating in push-pull mode.

T8307 I/O pins operates at 1.8 V. An external level shifter is required for interfacing with the 3 V MMC cards.

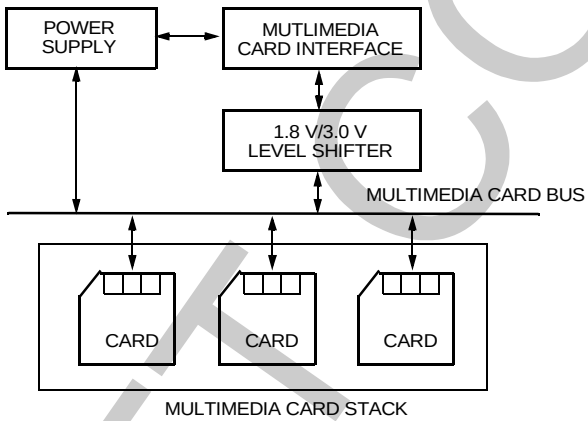


Figure 7.16-2 Multimedia Card System

7.16.1.2 Secure Digital Memory Card System

The secure digital memory card system (see Figure 7.16-3) consists of the host and cards connected in a star topology. Multimedia cards and secure digital memory cards can be used in the same system. The system host contains the secure digital card controller and a power supply. The power supply is not described in this document. T8307 I/O pins operates at 1.8 V. An external level shifter is required for interfacing with the 3 V SD cards.

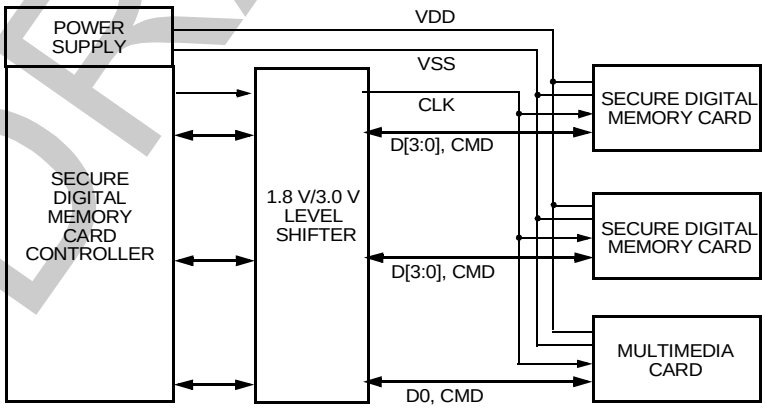


Figure 7.16-3 Secure Digital Memory Card System

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

The secure digital memory card bus is implemented using multiplexing logic, as shown in Figure 7.16-4. T8307 controls the output demultiplexers (1 to N) and the input multiplexers (N to 1) through GPIO pins.

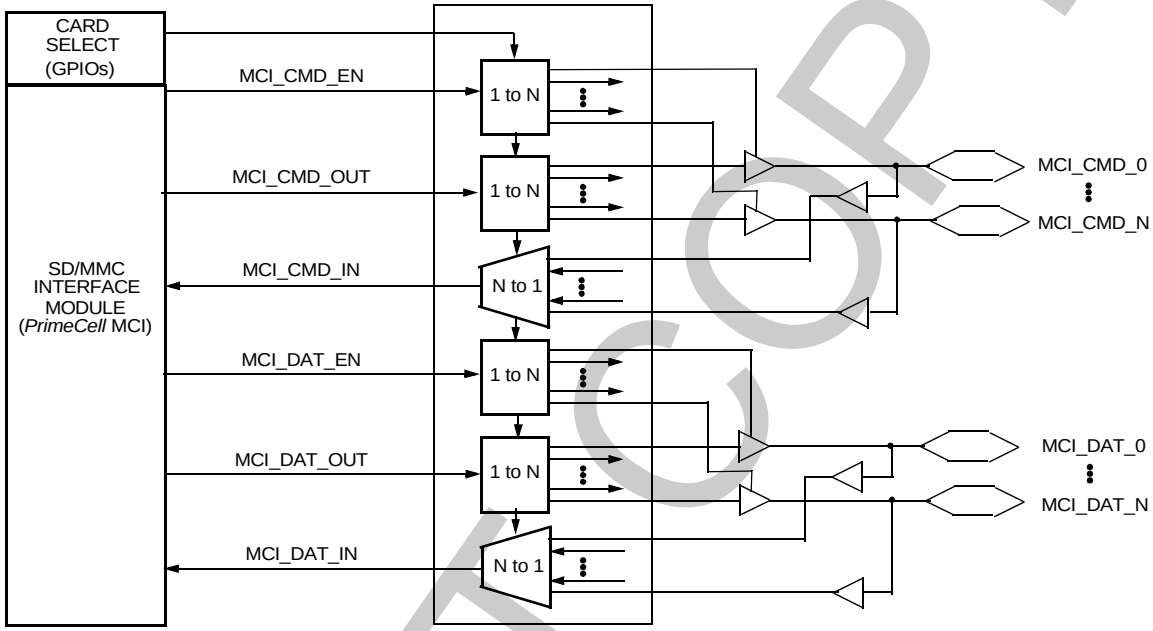


Figure 7.16-4 Secure Digital Memory Card Bus Implementation

The maximum number of cards that can be installed in a secure digital memory card system depends on the number of data ports on the secure digital card controller. The clock (CLK), power (V_{DD}), and ground (Vss) are common to all cards, while the command and data (DAT[3:0]) signals are dedicated to each card. After powerup, the secure digital cards only use DAT0 for data transfer. After initialization, the host can change the data bus width. If a multimedia card is connected to the secure digital card controller, only DAT0 is used for data transfer.

The following signals are used on the secure digital memory card bus:

- CLK: Host to card clock signal.
- CMD: Bidirectional command/response signal (one per card).
- DAT[3:0]: Bidirectional data signals (one per card).
- V_{DD}, Vss1, Vss2: Power and ground signals.

The *PrimeCell* MCI does not contain the bus multiplexing logic. If more than one secure digital memory cards needs to be supported, the user must implement this bus multiplexing logic with external components.

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

7.16.2 *PrimeCell* MCI Adapter

Figure 7.16-5 shows a simplified block diagram of the *PrimeCell* MCI adapter. In this figure, the card select and power connections are not shown for clarity.

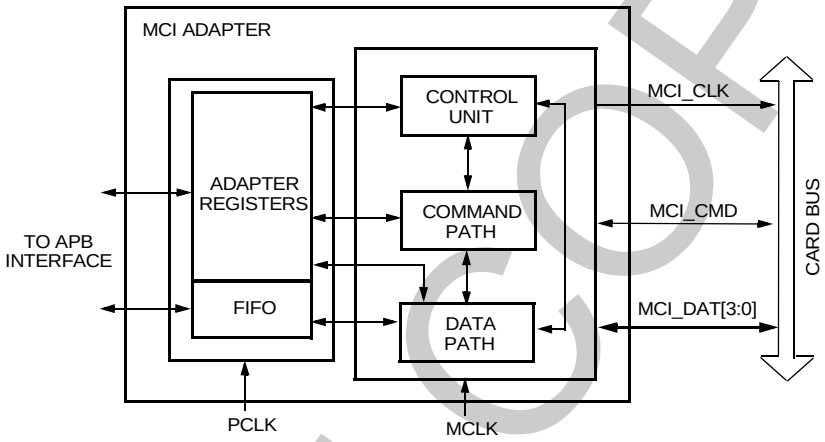


Figure 7.16-5 *PrimeCell* MCI Adapter

The *PrimeCell* MCI adapter is a multimedia/secure digital memory card bus master that provides an interface to the multimedia card stack or to the secure digital memory cards. It consists of five subunits:

- Adapter register block.
- Control unit.
- Command path.
- Data path.
- Data FIFO.

The adapter registers and FIFO use the APB bus clock domain. The control unit, command path, and data path use the *PrimeCell* MCI adapter clock domain.

7.16.2.1 Adapter Register Block

The adapter register block contains all system registers. This block also generates the signals that clear the static flags in the multimedia card. The clear signals are generated when 1 is written into the corresponding bit location of the MCIClear register. The clear signal for flags generated in the MCLK domain is synchronized to that domain.

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

7.16.2.2 Control Unit

The control unit contains the power management functions and the card bus clock divider. Figure 7.16-6 shows a block diagram of the control unit.

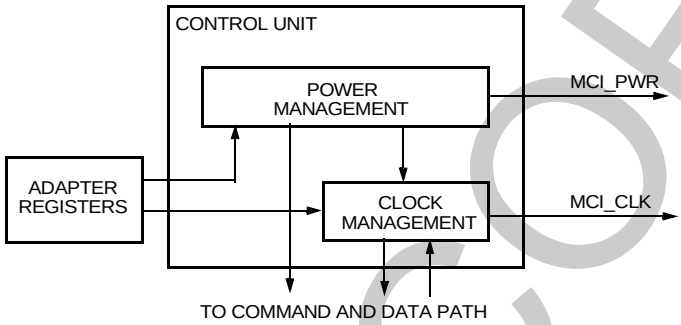


Figure 7.16-6 Control Unit

There are two power phases for *PrimeCell* MCI logic:

- Power-off.
- Power-on.

T8307 controls the external power supply unit through GPIO pins. During the power-off phase, *PrimeCell* MCI logic and the card bus output signals are disabled by the MCIPower register. After T8307 switches on the external power supply unit through GPIO pins, it should wait for a short period of time before switching on *PrimeCell* MCI through the MCIPower register so that the external power supply can reach the card bus operating voltage.

The clock management logic generates and controls the MCI_CLK signal. The MCI_CLK output can use either a clock divide or clock bypass mode. The clock output is inactive:

- After the *PrimeCell* MCI is reset.
- During the power-off phases.
- If the power-saving mode is enabled and the card bus is in the IDLE state (eight clock periods after both the command and data path subunits enter the IDLE phase).

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

7.16.2.3 Command Path

The command path subunit sends commands to and receives responses from the cards. Figure 7.16-7 shows a block diagram of the command path.

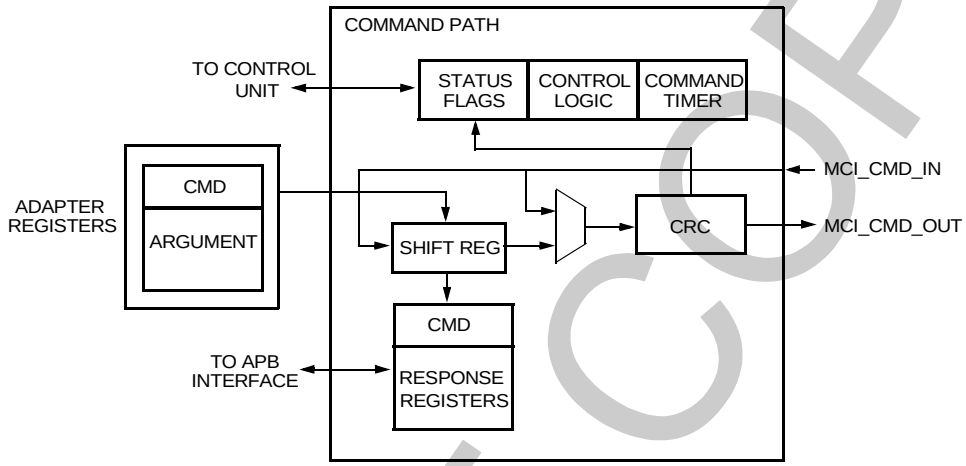


Figure 7.16-7 Command Path

Command Path State Machine

When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent, the command path state machine (CPSM) sets the status flags and enters the IDLE state if a response is not required. If a response is required, it waits for the response (see Figure 7.16-8). When the response is received, the received CRC code and the internally generated code are compared, and the appropriate status flags are set.

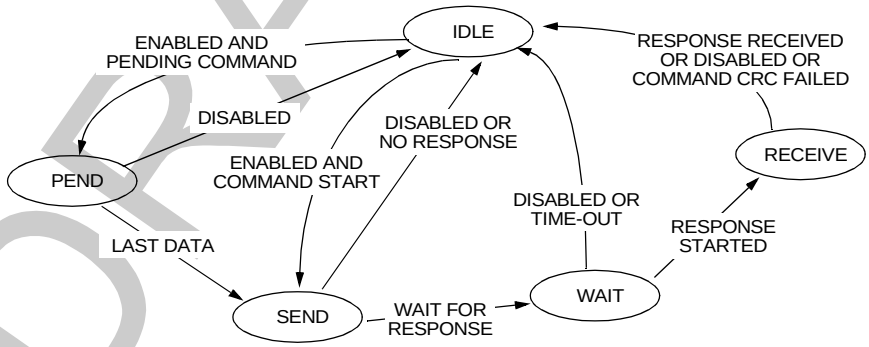


Figure 7.16-8 Command Path State Machine

When the wait state is entered, the command timer starts running. If the time-out is reached before the CPSM moves to the receive state, the time-out flag is set and the idle state is entered. The time-out period has a fixed value of 64 MCI_CLK clock periods.

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

If the interrupt bit is set in the command register, the timer is disabled and the CPSM waits for an interrupt request from one of the cards. If a pending bit is set in the command register, the CPSM enters the PEND state, and waits for a CmdPend signal from the data path subunit. When CmdPend is detected, the CPSM moves to the SEND state. This enables the data counter to trigger the stop command transmission. Note that the CPSM remains in the IDLE state for at least eight MCI_CLK periods to meet Ncc and Nrc timing constraints.

Figure 7.16-9 shows the *PrimeCell* MCI command transfer.

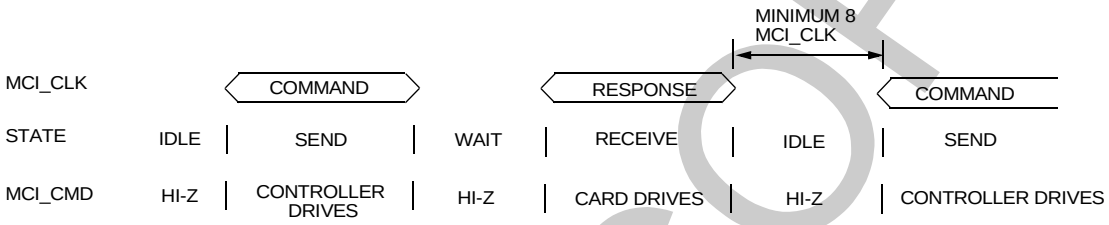


Figure 7.16-9 *PrimeCell* MCI Command Transfer

Command Format

The command path operates in a half-duplex mode so that commands and responses can either be sent or received. If the CPSM is not in the SEND state, the MCI_CMD output is in HI-Z state, as shown in Figure 7.16-9. Data on MCI_CMD is synchronous to the rising MCI_CLK edge. All commands have a fixed length of 48 bits. Table 7.16-1 shows the command format.

Table 7.16-1 Command Format

Bit Position	Width	Value	Description
47	1	0	Start bit.
46	1	1	Transmission bit.
45—40	6	—	Command index.
39—8	32	—	Argument.
7—1	7	—	CRC7.
0	1	1	End bit.

The *PrimeCell* MCI adapter supports two response types. Both use CRC error checking:

- 48-bit short response (see Table 7.16-2).
- 136-bit long response (see Table 7.16-3).

If the response does not contain CRC (CMD1 response), the device driver must ignore the CRC failed status.

Table 7.16-2 Short Response Format

Bit Position	Width	Value	Description
47	1	0	Start bit.
46	1	0	Transmission bit.
45—40	6	—	Command index.
39—8	32	—	Argument.
7—1	7	—	CRC7 (or 1111111).
0	1	1	End bit.

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

Table 7.16-3 Long Response Format

Bit Position	Width	Value	Description
135	1	0	Start bit.
134	1	1	Transmission bit.
133—128	6	111111	Reserved.
127—1	127	—	CID or CSD (including internal CRC7).
0	1	1	End bit.

The command register contains the command index (six bits sent to a card) and the command type. These determine whether the command requires a response, and whether the response is 48 or 136 bits long (see command register, MCICommand for more information). The command path implements the status flags shown in Table 7.16-4 (see Status register, MCIStatus for more information).

Table 7.16-4 Command Path Status Flags

Flag	Description
CmdRespEnd	Set if response CRC is OK.
CmdCrcFail	Set if response CRC fails.
CmdSent	Set when command (that does not require response) is sent.
CmdTimeOut	Response time-out.
CmdActive	Command transfer in progress.

The CRC generator calculates the CRC checksum for all bits before the CRC code. This includes the start bit, transmitter bit, command index, and command argument (or card status). The CRC checksum is calculated for the first 120 bits of CID or CSD for the long response format. Note that the start bit, transmitter bit, and the six reserved bits are not used in the long response format CRC calculation. The CRC checksum is a 7-bit value:

$$CRC[6:0] = Remainder\ [(M(x) * x^7)/G(x)]$$

$$G(x) = x^7 + x^3 + 1$$

$$M(x) = (start\ bit) * x^{39} + \ldots + (last\ bit\ before\ CRC) * x^0\ , or$$

$$M(x) = (start\ bit) * x^{119} + \ldots + (last\ bit\ before\ CRC) * x^0$$

7.16.2.4 Data Path

The data path subunit transfers data to and from cards. Figure 7.16-10 shows a block diagram of the data path.

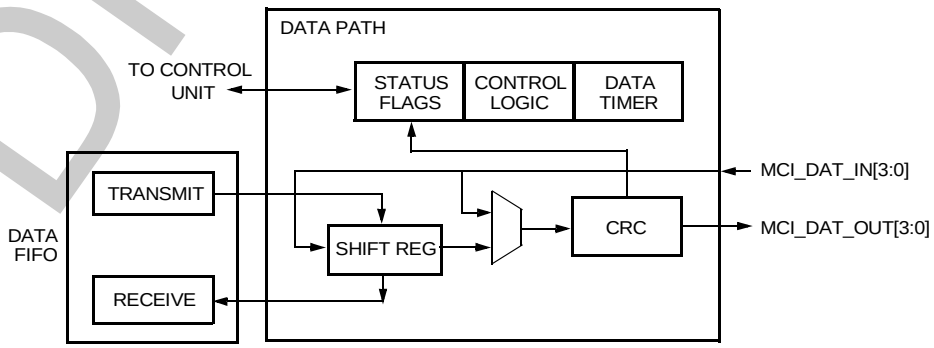


Figure 7.16-10 Data Path

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

The user can program the card data bus width using the clock control register. If the wide bus mode is enabled, data is transferred at four bits per clock cycle over all four data signals (MCI_DAT[3:0]). If the wide bus mode is not enabled, only one bit per clock cycle is transferred over MCI_DAT0.

Depending on the transfer direction (send or receive), the Data Path State Machine (DPSM) moves to the WAIT_S or WAIT_R state when it is enabled:

- Send: The DPSM moves to the WAIT_S state. If there is data in the send FIFO, the DPSM moves to the SEND state, and the data path subunit starts sending data to a card.
- Receive: The DPSM moves to the WAIT_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the RECEIVE state, and the data path subunit starts receiving data from a card.

Data Path State Machine

The DPSM operates at MCI_CLK frequency. Data on the card bus signals is synchronous to the rising edge of MCI_CLK. The DPSM has six states, as shown in Figure 7.16-11.

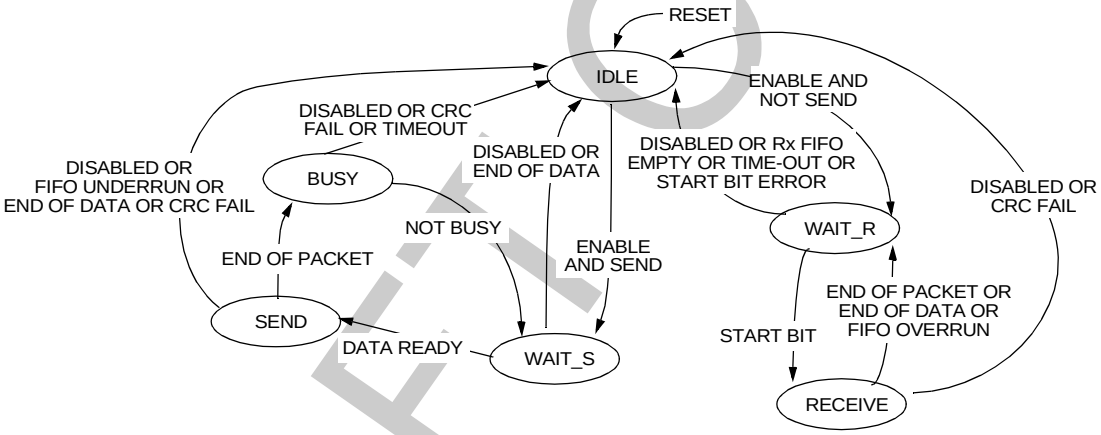


Figure 7.16-11 Data Path State Machine

- IDLE: The data path is inactive, and the MCI_DAT[3:0] outputs are in HI-Z. When the data control register is written and the enable bit is set, the DPSM loads the data counter with a new value and, depending on the data direction bit, moves to either the WAIT_S or WAIT_R state.
- WAIT_R: If the data counter equals zero, the DPSM moves to the IDLE state when the receive FIFO is empty. If the data counter is not zero, the DPSM waits for a start bit on MCI_DAT. The DPSM moves to the receive state if it receives a start bit before a time-out, and loads the data block counter. If it reaches a time-out before it detects a start bit, or a start bit error occurs, it moves to the idle state, and sets the time-out status flag.
- RECEIVE: Serial data received from a card is packed in bytes and written to the data FIFO. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block or stream:
  - — In block mode, when the data block counter reaches zero, the DPSM waits until it receives the CRC code. If the received code matches the internally generated CRC code, the DPSM moves to the WAIT_R state. If not, the CRC fail status flag is set and the DPSM moves to the idle state.
  - — In stream mode, the DPSM receives data while the data counter is not zero. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the WAIT_R state.

  If a FIFO overrun error occurs, the DPSM sets the FIFO error flag and moves to the WAIT_R state.

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

- WAIT_S: The DPSM moves to the idle state if the data counter is zero. If not, it waits until the data FIFO empty flag is deasserted, and moves to the send state. Note that the DPSM remains in the WAIT_S state for at least two clock periods to meet Nwr timing constraints.
- SEND: The DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block or stream:
  - — In block mode, when the data block counter reaches zero, the DPSM sends an internally generated CRC code and end bit, and moves to the busy state.
  - — In stream mode, the DPSM sends data to a card while the enable bit is high and the data counter is not zero. It then moves to the idle state.

  If a FIFO underrun error occurs, the DPSM sets the FIFO error flag and moves to the idle state.
- BUSY: The DPSM waits for the CRC status flag:
  - — If it does not receive a positive CRC status, it moves to the IDLE state and sets the CRC fail status flag.
  - — If it receives a positive CRC status, it moves to the WAIT_S state if MCI_DAT0 is not LOW (the card is not busy).

  If a time-out occurs while the DPSM is in the busy state, it sets the data time-out flag and moves to the IDLE state.

  The data timer is enabled when the DPSM is in the WAIT_R or BUSY state, and generates the data time-out error:
  - — When transmitting data, the time-out occurs if the DPSM stays in the BUSY state for longer than the programmed time-out period.
  - — When receiving data, the time-out occurs if the end of the data is not true, and if the DPSM stays in the WAIT_R state for longer than the programmed time-out period.

Data Counter

The data counter has two functions:

- To stop a data transfer when it reaches zero. This is the end of the data condition.
- To start transferring a pending command (see Figure 7.16-12). This is used to send the stop command for a stream data transfer.

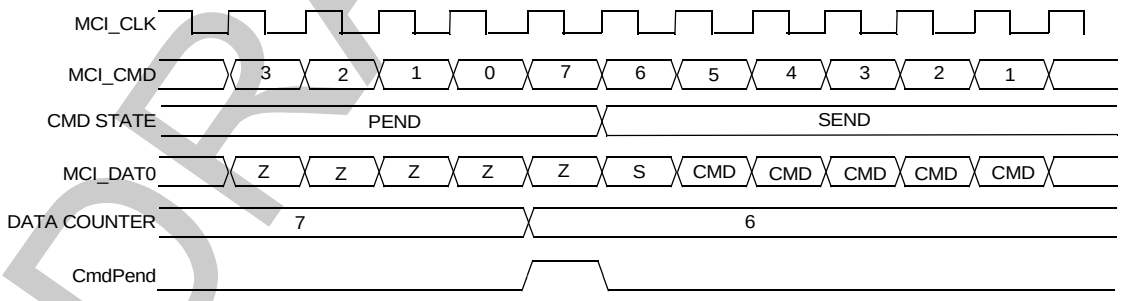


Figure 7.16-12 Pending Command Start

The data block counter determines the end of a data block. If the counter is zero, the end-of-data condition is TRUE (see Data control register, MCIDataCtrl for more information).

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

Bus Mode

In wide bus mode, all four data signals (MCI_DAT[3:0]) are used to transfer data, and the CRC code is calculated separately for each data signal. While transmitting data blocks to a card, only MCI_DAT0 is used for the CRC token and busy signaling. The start bit must be transmitted on all four data signals at the same time (during the same clock period). If the start bit is not detected on all data signals on the same clock edge while receiving data, the DPSM sets the start bit error flag and moves to the IDLE state.

The data path also operates in half-duplex mode, where data is either sent to a card or received from a card. While not being transferred, MCI_DAT[3:0] are in the HI-Z state. Data on these signals is synchronous to the rising edge of the clock period.

If the user selects wide mode, both MCI_DAT0_EN and MCI_DAT_EN outputs are driven low at the same time. If not, the MCI_DAT[3:1] outputs are always in HI-Z state (MCI_DAT_EN) is driven HIGH), and only the MCI_DAT0 output is driven LOW when data is transmitted.

CRC Token Status

The CRC token status follows each write data block and determines whether a card has received the data block correctly. When the token has been received, the card asserts a busy signal by driving MCI_DAT0 LOW. Table 7.16-5 shows the CRC token status values.

Table 7.16-5 CRC Token Status

Token	Description
010	Card has received error-free data block.
101	Card has detected a CRC error.

Status Flags

Table 7.16-6 lists the data path status flags (see status register, MCIStatus for more information).

Table 7.16-6 Data Path Status Flags

Flag	Description
TxFifoFull	Transmit FIFO is full.
TxFifoEmpty	Transmit FIFO is empty.
TxFifoHalfEmpty	Transmit FIFO is half full.
TxDataAvlbl	Transmit FIFO data available.
TxUnderrun	Transmit FIFO underrun error.
RxFifoFull	Receive FIFO is full.
RxFifoEmpty	Receive FIFO is empty.
RxFifoHalfFull	Receive FIFO is half full.
RxDataAvlbl	Receive FIFO data available.
RxOverrun	Receive FIFO overrun error.
DataBlockEnd	Data block sent/received.
StartBitErr	Start bit not detected on all data signals in wide bus mode.
DataCrcFail	Data packet CRC failed.
DataEnd	Data end (data counter is zero).
DataTimeOut	Data time-out.
TxActive	Data transmission in progress.
RxActive	Data reception in progress.

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

CRC Generator

The CRC generator calculates the CRC checksum only for the data bits in a single block, and is bypassed in data stream mode. The checksum is a 16-bit value:

$$CRC[15:0] = Remainder\ [(M(x) * x^{15})/G(x)]$$

$$G(x) = x^{16} + x^{12} + x^5 + 1$$

$$M(x) = (first\ data\ bit) * x^n + \ldots + (last\ data\ bit) * x^0$$

7.16.2.5 Data FIFO

The data FIFO (first-in-first-out) subunit contains a 32-bit wide, 16-word deep data buffer, and transmit and receive logic. Because the data FIFO operates in the APB clock domain (PCLK), all signals from the subunits in the *PrimeCell* MCI clock domain (MCLK) are resynchronized.

Depending on TxActive and RxActive, the FIFO can be disabled, transmit enabled, or receive enabled. TxActive and RxActive are driven by the data path subunit and are mutually exclusive:

- The transmit FIFO refers to the transmit logic and data buffer when TxActive is asserted.
- The receive FIFO refers to the receive logic and data buffer when RxActive is asserted.

Transmit FIFO

Data is written to the transmit FIFO through the APB interface once the MCI is enabled for transmission. When the write signal (TxWriteEn) is asserted, data can be written (on the rising edge of PCLK) into the FIFO location specified by the current value of the data pointer. The pointer is incremented after every FIFO write.

The transmit FIFO contains a data output register. This holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, the data path logic asserts TxRdPtrInc. This signal is synchronized with PCLK, and increments the read pointer and drives new data on the TxRdData output.

If the transmit FIFO is disabled, all status flags are deasserted, and the read and write pointers are reset. The data path subunit asserts TxActive when it transmits data. Table 7.16-7 lists the transmit FIFO status flags.

Table 7.16-7 Transmit FIFO Status Flags

Flag	Description
TxFifoFull	Set to high when all 16 transmit FIFO words contain valid data.
TxFifoEmpty	Set to high when the transmit FIFO does not contain valid data.
TxFifoHalfEmpty	Set to high when 8 or more transmit FIFO words are empty. This flag can be used as a DMA request.
TxDataAvlbl	Set to high when the transmit FIFO contains valid data. This flag is the inverse of the TxFifoEmpty flag.
TxUnderrun	Set to high when an underrun error occurs. This flag is cleared by writing to the MCIClear register.

Receive FIFO

When the data path subunit receives a word of data, it drives data on the write data bus and asserts the write enable signal. This signal is synchronized to the PCLK domain. The write pointer is incremented after the write is completed, and the receive FIFO control logic asserts RxWrDone, that then deasserts the write enable signal.

On the read side, the content of the FIFO word pointed to by the current value of the read pointer is driven on the read data bus. The read pointer is incremented when the APB bus interface asserts RxRdPrtInc.

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

If the receive FIFO is disabled, all status flags are deasserted, and the read and write pointers are reset. The data path subunit asserts RxActive when it receives data. Table 7.16-8 lists the receive FIFO status flags.

Table 7.16-8 Receive FIFO Status Flags

Flag	Description
RxFifoFull	Set to high when all 16 receive FIFO words contain valid data.
RxFifoEmpty	Set to high when the receive FIFO does not contain valid data.
RxFifoHalfFull	Set to high when 8 or more receive FIFO words contain valid data. This flag can be used as a DMA request.
RxDataAvlbl	Set to high when the receive FIFO is not empty. This flag is the inverse of the RxFifoEmpty flag.
RxOverrun	Set to high when an overrun error occurs. This flag is cleared by writing to the MCIClear register.

7.16.3 APB Interface

The APB interface (see Figure 7.16-13) generates the interrupt and DMA requests, and accesses the *PrimeCell* MCI adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt/DMA logic.

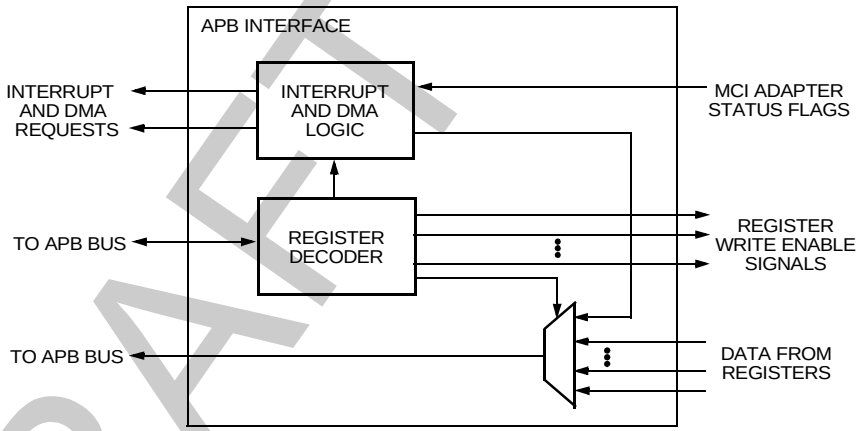


Figure 7.16-13 APB Interface

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

7.16.3.1 Interrupt Logic

The interrupt logic (see Figure 7.16-14) generates two interrupt request signals, that are asserted when at least one of the selected status flags is high. A status flag generates the interrupt request if a corresponding mask flag is set. The interrupt request can be asserted even if PCLK is disabled.

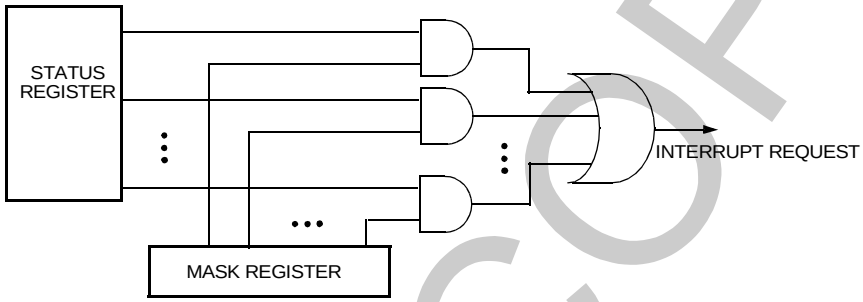


Figure 7.16-14 Interrupt Request Logic

A separate mask register is provided for each interrupt request signal (see interrupt mask registers, MCIMask0—MCIMask1 for more information).

7.16.3.2 DMA

The interface to the DMA controller includes the signals described in Table 7.16-9.

Table 7.16-9 DMA Controller Interface Signals

Signal	Type	Description
DMASREQ	Single-word DMA transfer request, asserted by *PrimeCell* MCI	For receive: Asserted if data counter is zero and receive FIFO contains more than one and fewer than eight words. For transmit: Asserted if fewer than eight and more than one word remain for transfer to FIFO.
DMABREQ	Burst DMA transfer request, asserted by *PrimeCell* MCI	For receive: Asserted if FIFO contains eight words and data counter is not zero, or if FIFO contains more than eight words. For transmit: Asserted if more than eight words remain for transfer to FIFO.
DMALSREQ	Last single-word DMA transfer request, asserted by *PrimeCell* MCI	For receive: Asserted if data counter is zero and FIFO contains only one word. For transmit: Asserted if only one word remains for transfer to FIFO.
DMALBREQ	Last burst DMA transfer request, asserted by *PrimeCell* MCI	For receive: Asserted if data counter is zero and FIFO contains eight words. For transmit: Asserted if only eight words remain for transfer to FIFO.
DMACLR	DMA request clear, asserted by DMA controller to clear request signals	Asserted during transfer of last data in burst if DMA burst transfer is requested.

Because the four request signals are mutually exclusive, only one signal is asserted at a time. The signal remains asserted until DMACLR is asserted. After this, a request signal can be active again, depending on the conditions described in Table 7.16-9. When the enable bit in the data control register is cleared, the data path is disabled and all request signals are deasserted.

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

The DMA signals are synchronous with PCLK. Figure 7.16-15 shows the DMA transfer of the last three words.

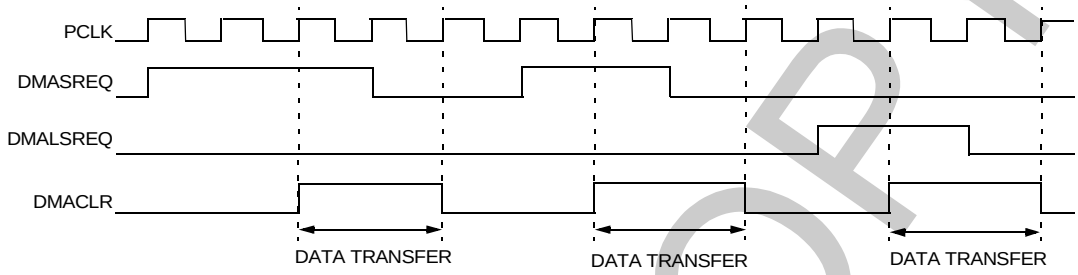


Figure 7.16-15 DMA Interface

7.16.4 Timing Requirements

The clock output is routed back to the *PrimeCell* MCI and is used to clock the output registers, to meet the hold time requirements of MCI_CMD and MCI_DATx. Figure 7.16-16 shows a block diagram of the clock output routing.

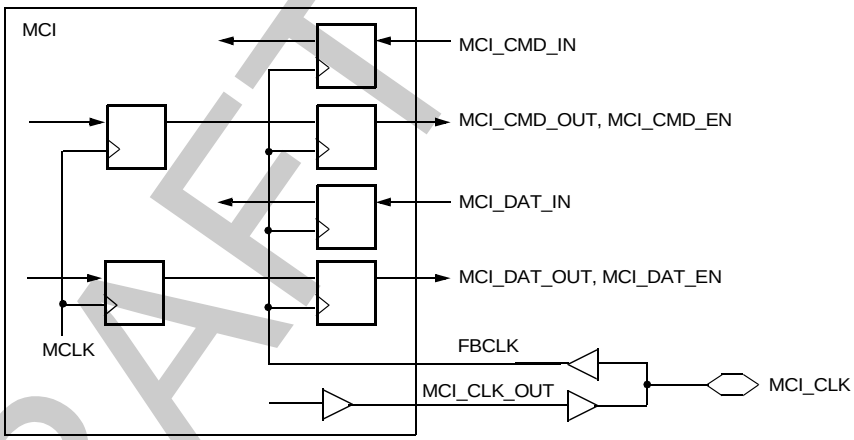


Figure 7.16-16 Clock Output Retiming Logic

Figure 7.16-17 shows the signal timing relationship when the *PrimeCell* MCI is integrated in T8307.

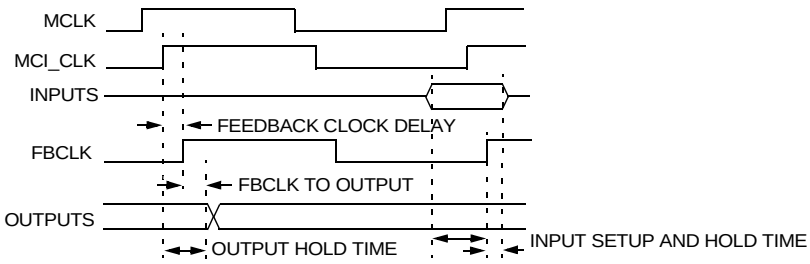


Figure 7.16-17 MCI_CMD and MCI_DAT Timing

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

7.16.5 Registers

7.16.5.1 MCIPower Register (MCIPower)

The MCIPower register controls the activity of *PrimeCell* MCI in accordance to external card power supply. When the external power supply is switched on, the software waits until the supply output is stable before setting *PrimeCell* MCI to the power-on phase.

Note: After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

Table 7.16-10 MCIPower Register, Address (0x700CA000)

Bit	31—2	1—0
Name	RSVD	Ctrl

Bit	Name	Description
31—2	RSVD	Reserved.
1—0	Ctrl	00 = Power off. 01 = Reserved. 10 = Reserved. 11 = Power on.

7.16.5.2 Clock Control Register (MCIClock)

The MCIClock register controls the MCI_CLK output. While the SD/MMC interface module is in identification mode, the MCI_CLK frequency must be less than 400 kHz. The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.

Note: After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

Table 7.16-11 Clock Control Register (MCIClock), Address (0x700CA004)

Bit	31—12	11	10	9	8	7—0
Name	RSVD	WideBus	Bypass	PwrSave	Enable	ClkDiv

Bit	Name	Description
31—12	RSVD	Reserved.
11	WideBus	Enable wide bus mode: 0 = Standard bus mode (only MCI_DAT0 used). 1 = Wide bus mode (MCI_DAT[3:0] used).
10	Bypass	Enable bypass of clock divide logic: 0 = Disable bypass. 1 = Enable bypass.
9	PwrSave	Disable SD/MMC interface clock output when bus is idle: 0 = Always enabled. 1 = Clock enabled when bus is active.
8	Enable	Enable SD/MMC interface bus clock: 0 = Clock disabled. 1 = Clock enabled.
7—0	ClkDiv	SD/MMC interface bus clock period: MCI_CLK frequency = MCLK/(2 x (ClkDiv + 1)).

7.16.5.3 Argument Register (MCIArgument)

The MCIArgument register contains a 32-bit command argument, which is sent to a card as part of a command message. If a command contains an argument, it must be loaded into the argument register before writing a command to the command register.

Table 7.16-12 Argument Register (MCIArgument), Address (0x700CA008)

Bit	31—0
Name	CmdArg

Bit	Name	Description
31—0	CmdArg	Command argument.

7.16.5.4 Command Register (MCICommand)

The MCICommand register contains the command index and command type bits:

- The command index field is sent to a card as part of a command message.
- The other command type fields controls the command path state machine (CPSM).
- Writing 1 to the enable bit starts the command send operation, while clearing the bit disables the CPSM.

Note: After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

Table 7.16-13 Command Register (MCICommand), Address (0x700CA00C)

Bit	31—11	10	9	8	7	6	5—0
Name	RSVD	Enable	Pending	Interrupt	LongRsp	Response	CmdIndex

Bit	Name	Description
31—11	RSVD	Reserved.
10	Enable	If set, CPSM is enabled.
9	Pending	If set, CPSM waits for CmdPend before it starts sending a command.
8	Interrupt	If set, CPSM disables command timer and waits for interrupt request.
7	LongRsp	If set, CPSM receives a 136-bit long response.
6	Response	If set, CPSM waits for a response.
5—0	CmdIndex	Command index.

Table 7.16-14 shows the response types.

Table 7.16-14 Command Response Types

Response	LongRsp	Description
0	0	No response, expect CmdSent flag.
0	1	
1	0	Short response, expect CmdRespEnd or CmdCrcFail flag.
1	1	Long response, expect CmdRespEnd or CmdCrcFail flag.

7.16.5.5 Command Response Register (MCIRespCommand)

The MCIRespCommand register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long response), the RespCmd field is unknown, although it must contain 111111 (the value of the reserved field from the response).

Table 7.16-15 Command Response Register (MCIRespCommand), Address (0x700CA010)

Bit	31—6	5—0
Name	RSVD	RespCmd

Bit	Name	Description
31—6	RSVD	Reserved.
5—0	RespCmd	Response command index.

7.16.5.6 Response Registers (MCIResponse0—MCIResponse3)

The MCIResponse0—MCIResponse3 registers contain the status of a card, which is part of the received response.

Table 7.16-16 Response Registers (MCIResponse0—MCIResponse3), Addresses (0x700CA014—0x700CA020)

Bit	31—0
Name	Status

Bit	Name	Description
31—0	Status	Card status.

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

The card status size can be 32 or 127 bits, depending on the response type (see Table 7.16-17). The most significant bit of the card status is received first. The MCIResponse3 register least significant bit is always 0.

Table 7.16-17 Response Register Type

Description	Short Response	Long Response
MCIResponse0	Card status [31:0]	Card status [127:96]
MCIResponse1	Unused	Card status [95:64]
MCIResponse2	Unused	Card status [63:32]
MCIResponse3	Unused	Card status [31:1]

7.16.5.7 Data Timer Register (MCIDataTimer)

The MCIDataTimer register contains the data time-out period, in card bus clock periods. A counter loads the value from the data timer register, and starts decrementing when the data path state machine (DPSM) enters the WAIT_R or BUSY state. If the timer reaches 0 while the DPSM is in either of these states, the time-out status flag is set.

A data transfer must be written to the data timer register and the data length register before being written to the data control register.

Table 7.16-18 Data Timer Register (MCIDataTimer), Address (0x700CA024)

Bit	31—0
Name	DataTime

Bit	Name	Description
31—0	DataTime	Data time-out period.

7.16.5.8 Data Length Register (MCIDataLength)

The MCIDataLength register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts. For a block data transfer, the value in the data length register must be a multiple of the block size (see data control register, MCIDataCtrl).

A data transfer must be written to the data timer register and the data length register before being written to the data control register.

Table 7.16-19 Data Length Register (MCIDataLength), Address (0x700CA028)

Bit	31—16	15—0
Name	RSVD	DataLength

Bit	Name	Description
31—16	RSVD	Reserved.
15—0	DataLength	Data length value.

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

7.16.5.9 Data Control Register (MCIDataCtrl)

The MCIDataCtrl register controls the data path state machine (DPSM).

Note: After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

Table 7.16-20 Data Control Register (MCIDataCtrl), Address (0x700CA02C)

Bit	31—8	7—4	3	2	1	0
Name	RSVD	BlockSize	DMAEnable	Mode	Direction	Enable

Bit	Name	Description
31—8	RSVD	Reserved.
7—4	BlockSize	Data block length.
3	DMAEnable	Enable DMA: 0 = DMA disabled. 1 = DMA enabled.
2	Mode	Data transfer mode: 0 = Block data transfer. 1 = Stream data transfer.
1	Direction	Data transfer direction: 0 = From controller to card. 1 = From card to controller.
0	Enable	Data transfer enabled.

Data transfer starts if 1 is written to the enable bit. Depending on the direction bit, the DPSM moves to the WAIT_S or WAIT_R state. The enable bit does not need to be cleared after data transfer. Table 7.16-21 shows the data block length if block data transfer mode is selected.

Table 7.16-21 Data Block Length

Block Size	Block Length
0	2^0 = 1 byte
1	2^1 = 2 byte
. . .	. . .
11	2^{11} = 2048 byte
12—15	Reserved

7.16.5.10 Data Counter Register (MCIDataCnt)

The MCIDataCnt register loads the value from the data length register (see data length register, MCIDataLength) when the DPSM moves from the IDLE state to the WAIT_R or WAIT_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then moves to the IDLE state and the data status end flag is set.

Note: This register should be read only when the data transfer is complete.

Table 7.16-22 Data Counter Register (MCIDataCnt), Address (0x700CA030)

Bit	31—16	15—0
Name	RSVD	DataCount

Bit	Name	Description
31—16	RSVD	Reserved.
15—0	DataCount	Remaining data.

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

7.16.5.11 Status Register (MCIStatus)

The MCIStatus register is a read-only register. It contains two types of flag:

- Static [10:0]. These remain asserted until they are cleared by writing to the Clear register (see clear register, MCIClear).
- Dynamic [21:11]. These change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and deasserted as data while written to the FIFO).

Table 7.16-23 Status Register (MCIStatus), Address (0x700CA034)

Bit	31—22	21	20	19	18	17	16	15
Name	RSVD	RxDataAvlbl	TxDataAvlbl	RxFifoEmpty	TxFifoEmpty	RxFifoFull	TxFifoFull	RxFifoHalfFull

Bit	14	13	12	11	10	9	8	7
Name	TxFifoHalfEmpty	RxActive	TxActive	CmdActive	DataBlockEnd	StartBitErr	DataEnd	CmdSent

Bit	6	5	4	3	2	1	0
Name	CmdRespEnd	RxOverrun	TxUnderrun	DataTimeOut	CmdTimeOut	DataCrcFail	CmdCrcFail

Bit	Name	Description
31—22	RSVD	Reserved.
21	RxDataAvlbl	Data available in receive FIFO.
20	TxDataAvlbl	Data available in transmit FIFO.
19	RxFifoEmpty	Receive FIFO empty.
18	TxFifoEmpty	Transmit FIFO empty.
17	RxFifoFull	Receive FIFO full.
16	TxFifoFull	Transmit FIFO full.
15	RxFifoHalfFull	Receive FIFO half full.
14	TxFifoHalfEmpty	Transmit FIFO half empty.
13	RxActive	Data receive in progress.
12	TxActive	Data transmit in progress.
11	CmdActive	Command transfer in progress.
10	DataBlockEnd	Data block sent/received (CRC check passed).
9	StartBitErr	Start bit not detected on all data signals in wide bus mode.
8	DataEnd	Data end (data counter is zero).
7	CmdSent	Command sent (no response required).
6	CmdRespEnd	Command response received (CRC check passed).
5	RxOverrun	Receive FIFO overrun error.
4	TxUnderrun	Transmit FIFO underrun error.
3	DataTimeOut	Data time-out.
2	CmdTimeOut	Command response time-out.
1	DataCrcFail	Data block sent/received (CRC check failed).
0	CmdCrcFail	Command response received (CRC check failed).

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

7.16.5.12 Clear Register (MCIClear)

The MCIClear register is a write-only register. The corresponding static flags can be cleared by writing a 1 to the corresponding bit in the register.

Table 7.16-24 Clear Register (MCIClear), Address (0x700CA038)

Bit	31—11	10	9	8	7	6
Name	RSVD	DataBlockEndClr	StartBitErrClr	DataEndClr	CmdSentClr	CmdRespEndClr

Bit	5	4	3	2	1	0
Name	RxOverrunClr	TxUnderrunClr	DataTimeOutClr	CmdTimeOutClr	DataCrcFailClr	CmdCrcFailClr

Bit	Name	Description
31—11	RSVD	Reserved.
10	DataBlockEndClr	Data block sent/received (CRC check passed).
9	StartBitErrClr	Start bit not detected on all data signals in wide bus mode.
8	DataEndClr	Data end (data counter is zero).
7	CmdSentClr	Command sent (no response required).
6	CmdRespEndClr	Command response received (CRC check passed).
5	RxOverrunClr	Receive FIFO overrun error.
4	TxUnderrunClr	Transmit FIFO underrun error.
3	DataTimeOutClr	Data time-out.
2	CmdTimeOutClr	Command response time-out.
1	DataCrcFailClr	Data block sent/received (CRC check failed).
0	CmdCrcFailClr	Command response received (CRC check failed).

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

7.16.5.13 Interrupt Mask Registers (MCIMask0—MCIMask1)

There are two interrupt mask registers, MCIMask0—MCIMask1, one for each interrupt request signal. The interrupt mask registers determine which status flags generate an interrupt request by setting the corresponding bit to 1.

Table 7.16-25 Interrupt Mask Registers (MCIMask0—MCIMask1), Addresses (0x700CA03C—0x700CA040)

Bit	31—22	21	20	19	18	17	16	15	14	13	12	11
Name	RSVD	Mask21	Mask20	Mask19	Mask18	Mask17	Mask16	Mask15	Mask14	Mask13	Mask12	Mask11

Bit	10	9	8	7	6	5	4	3	2	1	0
Name	Mask10	Mask9	Mask8	Mask7	Mask6	Mask5	Mask4	Mask3	Mask2	Mask1	Mask0

Bit	Name	Description
31—22	RSVD	Reserved.
21	Mask21	Mask RxDataAvlbl flag.
20	Mask20	Mask TxDataAvlbl flag.
19	Mask19	Mask RxFifoEmpty flag.
18	Mask18	Mask TxFifoEmpty flag.
17	Mask17	Mask RxFifoFull flag.
16	Mask16	Mask TxFifoFull flag.
15	Mask15	Mask RxFifoHalfFull flag.
14	Mask14	Mask TxFifoHalfEmpty flag.
13	Mask13	Mask RxActive flag.
12	Mask12	Mask TxActive flag.
11	Mask11	Mask CmdActive flag.
10	Mask10	Mask DataBlockEnd flag.
9	Mask9	Mask StartBitErr flag.
8	Mask8	Mask DataEnd flag.
7	Mask7	Mask CmdSent flag.
6	Mask6	Mask CmdRespEnd flag.
5	Mask5	Mask RxOverrun flag.
4	Mask4	Mask TxUnderrun flag.
3	Mask3	Mask DataTimeOut flag.
2	Mask2	Mask CmdTimeOut flag.
1	Mask1	Mask DataCrcFail flag.
0	Mask0	Mask CmdCrcFail flag.

7 Call Processor (CP) Block (continued)

7.16 SD/MMC Interface (continued)

7.16.5.14 Secure Digital Memory Card Select Register (MCISelect)

This register is reserved.

Table 7.16-26 Secure Digital Memory Card Select Register (MCISelect), Address (0x700CA044)

Bit	31—0
Name	RSVD

Bit	Name	Description
31—0	RSVD	Reserved.

7.16.5.15 FIFO Counter Register (MCIFifoCnt)

The MCIFifoCnt register contains the remaining number of words to be written to or read from the FIFO. The FIFO counter loads the value from the data length register (See Data Length Register, MCIDataLength) when the Enable bit is set in the data control register. If the data length is not word aligned (multiple of 4), the remaining 1 to 3 bytes are regarded as a word.

Table 7.16-27 FIFO Counter Register (MCIFifoCnt), Address (0x700CA048)

Bit	31—15	14—0
Name	RSVD	DataCount

Bit	Name	Description
31—15	RSVD	Reserved.
14—0	DataCount	Remaining data.

7.16.5.16 Data FIFO Register (MCIFIFO)

The receive and transmit FIFOs can be read or written as 32-bit wide registers. The FIFOs contain 16 entries on 16 sequential addresses. This allows the microprocessor to use its load and store multiple operands to read/write to the FIFO.

Table 7.16-28 Data FIFO Register (MCIFIFO), Address (0x700CA080—0x700CA0BC)

Bit	31—0
Name	Data

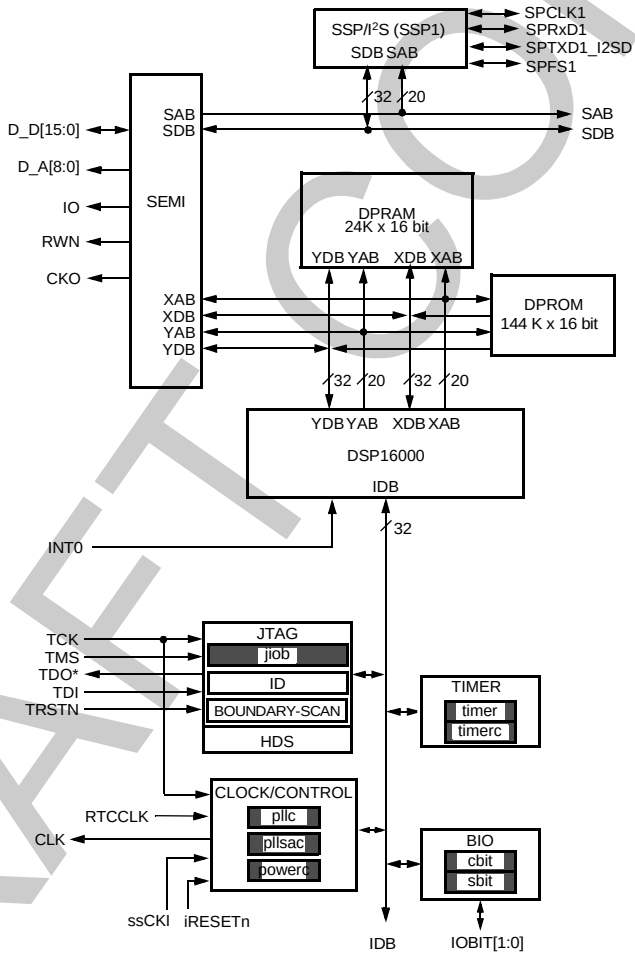
Bit	Name	Description
31—0	Data	FIFO data.

8 Digital Signal Processor (DSP) Block

8.1 DSP Block Architectural Overview

The DSP block of T8307 consists of a DSP16000 core (DSP) together with on-chip memory and peripherals. Advanced architectural features with an expanded instruction set deliver a dramatic increase in performance compared to traditional DSP architectures for signal processing algorithms. This increase in performance, together with an efficient design implementation, results in an extremely cost-efficient and power-efficient solution for wireless and multimedia applications.

Figure 8.1-1 shows the DSP section of the T8307 digital baseband processor.



* Internal daisy chain can be enabled through TEST1-3 pins.

2395 (F).e

Figure 8.1-1 T8307 DSP Section Block Diagram

8 Digital Signal Processor (DSP) Block (continued)

8.1 DSP Block Architectural Overview (continued)

Table 8.1-1 T8307 DSP Block Diagram Legend

Symbol	Description
BIO	Bit I/O unit.
BOUNDARY SCAN	Boundary-scan register.
cbit	16-bit BIO control register.
CLK	Internal clock signal.
DSP	DSP16000 core—system master.
DPROM	Dual-port read-only memory. 144 Kwords.
DPRAM	Dual-port random-access memory. DSP has 24 Kwords. Private code (X) and data (Y).
HDS	Hardware development system.
HDSROM	Internal read-only memory for HDS code.
ID	JTAG port identification register accessible via the JTAG port.
IDB	Internal data bus.
jiob	32-bit JTAG test register.
JTAG	JTAG test port.
pllc	16-bit phase-locked loop control register.
pllsac	16-bit phase-locked loop status register.
powerc	Power control register.
RTCCLK	32 kHz crystal oscillator clock signal.
SAB	20-bit system address bus. address for system bus accesses.
sbit	16-bit BIO status/control register.
SDB	32-bit system data bus. Data for system bus accesses.
SEMI	System and external memory interface.
SSP/I^2S	Synchronous serial port with I^2S feature.
timer	16-bit timer running count register for TIMER.
TIMER	Programmable timer for DSP.
timerc	16-bit timer control register for TIMER.
XAB	20-bit X-memory space address bus.
XDB	32-bit X-memory space data bus.
YAB	20-bit Y-memory space address bus.
YDB	32-bit Y-memory space data bus.

8 Digital Signal Processor (DSP) Block (continued)

8.1 DSP Block Architectural Overview (continued)

8.1.1 DSP16000 Core

The DSP16000 core (DSP) is the signal processing engine of the T8307 digital baseband processor. The DSP16000 is a modified Harvard architecture with separate sets of buses for the instruction/coefficient (X-memory) and data (Y-memory) spaces. Each set of buses has 20 bits of address and 32 bits of data. The core contains data and address arithmetic units and control for on-chip memory and peripherals.

8.1.2 Clock Synthesizer (PLL)

T8307 DSP block powers up with an input clock (CKI) as the source for the processor clock (CLK). An on-chip clock synthesizer (PLL) that runs at a frequency multiple of CKI can also be used to generate CLK. The clock synthesizer is deselected and powered down on reset. The selection of the clock source is under software control of DSP. See Section 8.13 for details.

8.1.3 Dual-Port RAM (DPRAM)

DSP has a private block of DPRAM consisting of 6 banks (banks 0—5) of zero wait-state memory. Each bank consists of 4K x 16-bit words and has two separate address and data ports: one port to the core's instruction/coefficient (X-memory) space, and a second port to the core's data (Y-memory) space.

DPRAM is organized into even and odd interleaved banks for which each even/odd address pair is a 32-bit wide module (see Section 8.10 for details). DPRAM supports single-word, aligned double-word, and misaligned double-word accesses.

8.1.4 Dual-Port ROM (DPROM)

DSP has 18 banks (banks 0—17) of DPROM. Each bank is 8K x 16 bits and is dual-ported (X and Y). DSP has 144 Kwords.

8.1.5 Internal HDS ROM (HDSROM)

DSP has its own internal ROM that contains software to support the Agere hardware development system (HDS).

8.1.6 System and External Memory Interface (SEMI)

The SEMI interfaces the core to external memory and I/O devices. The SEMI also interfaces the core to the internal ICP-shared memory and SSP/I^{2}S block via the internal system bus (SAB and SDB). Additionally, the SEMI supports interface to the CSP8307 device. See Section 8.12 for details.

8.1.7 Bit Input/Output Units (BIO)

T8307 DSP block contains a BIO unit for the DSP16000 core. BIO unit provides convenient and efficient monitoring and control of two individually configurable pins. If configured as outputs, the pins can be individually set, cleared, or toggled. If configured as inputs, individual pins or combinations of pins can be tested for patterns. Flags returned by the BIO can be tested by conditional instructions. See Section 8.5 for details.

8.1.8 Timer Unit (TIMER)

T8307 DSP block contains one timer unit for the DSP16000 core. The timer can be used to provide an interrupt, either single or repetitive, at the expiration of a programmed interval. More than nine orders of magnitude of interval selection are provided. See Section 8.6 for more information.

8.1.9 Synchronous Serial Port with Inter IC Sound Support (SSP/I^{2}S)

T8307 DSP-side SSPI^{2}S port supports all features provided by *ARM PrimeCell* PL022 (e.g., *Motorola* SPI, *Texas Instruments SSI*, and *National Semiconductor MICROWIRE*) and *Philips* I^{2}S formats.

8.1.10 Test Access Ports (JTAG)

T8307 DSP block contains a JTAG unit for the DSP16000 core. See Section 8.9 for details.

8.1.11 Hardware Development System (HDS)

T8307 DSP block contains an HDS unit for the DSP16000 core. The HDS is an on-chip hardware module available for debugging assembly-language programs that execute on the DSP16000 core in real-time. The main capability of the HDS is in allowing controlled visibility into the core's state during program execution. The HDS is enhanced with powerful debugging capabilities such as complex breakpointing conditions, multiple data/address watchpoint registers, and an intelligent trace mechanism for recording discontinuities. For the VoWLAN application, DSP programming tools are used only for Agere firmware development. Customer programming of the DSP is not supported. See Section 8.8 for details.

8 Digital Signal Processor (DSP) Block (continued)

8.2 DSP16000 Core Architectural Overview

T8307 DSP block contains one DSP16000 core (DSP). As shown in Figure 8.2-1, the core consists of four major blocks: system control and cache (SYS), data arithmetic unit (DAU), Y-memory space address arithmetic unit (YAAU), and X-memory space address arithmetic unit (XAAU). Bits within the **auc0** and **auc1** registers configure the DAU mode-controlled operations. See the *DSP16000 Digital Signal Processor Core* Information Manual for a complete description of the DSP16000 core.

8.2.1 System Control and Cache (SYS)

This section consists of the control block and the cache.

The control block provides overall system coordination that is mostly invisible to the user. The control block includes an instruction decoder and sequencer, a pseudorandom sequence generator (PSG), an interrupt and trap handler, a wait-state generator, and low-power standby mode control logic. The interrupt and trap handler are controlled by a user-locatable vector table and three levels of user-assigned interrupt priority.

SYS contains the **alf** register, which is a 16-bit register that contains AWAIT, a power-saving standby mode bit, and peripheral flags. The **inc0** and **inc1** registers are 20-bit interrupt control registers, and **ins** is a 20-bit interrupt status register.

Programs use the instruction cache to store and execute repetitive operations such as those found in an FIR or IIR filter section. The cache can contain up to thirty-one 16-bit and/or 32-bit instructions. The code in the cache can repeat up to $2^{16} - 1$ times without looping overhead. Operations in the cache that require a coefficient access execute at twice the normal rate because the XAAU and its associated bus are not needed for fetching instructions. The cache greatly reduces the need for writing in-line repetitive code and, therefore, reduces instruction/coefficient memory size requirements. In addition, the use of cache reduces power consumption because it eliminates memory accesses for instruction fetches.

The cache provides a convenient, low-overhead looping structure that is interruptible, savable, and restorable. The cache is addressable in both the X and Y memory spaces. An interrupt or trap handling routine can save and restore **cloop**, **cstate**, **csave**, and the contents of the cache. The **cloop** register controls the cache loop count. The **cstate** register contains the current state of the cache. The 32-bit **csave** register holds the opcode of the instruction following the loop instruction in program memory.

8.2.2 Data Arithmetic Unit (DAU)

The DAU is a power-efficient, dual-MAC (multiply/accumulate), parallel-pipelined structure that is tailored to communications applications. It can perform two double-word (32-bit) fetches, two multiplications, and two accumulations in a single instruction cycle. The dual-MAC parallel pipeline begins with two 32-bit registers, **x** and **y**. The pipeline treats the 32-bit registers as four 16-bit signed registers if used as input to two signed 16-bit x 16-bit multipliers. Each multiplier produces a full 32-bit result stored into registers **p0** and **p1**. The DAU can direct the output of each multiplier to a 40-bit ALU or a 40-bit 3-input ADDER. The ALU and ADDER results are each stored in one of eight 40-bit accumulators, **a0** through **a7**. Both the ALU and ADDER include an ACS (add/compare/select) function for Viterbi decoding. The DAU can direct the output of each accumulator to the ALU/ACS, the ADDER/ACS, or a 40-bit BMU (bit manipulation unit).

The ALU implements 2-input addition, subtraction, and various logical operations. The ADDER implements 2-input or 3-input addition and subtraction. To support Viterbi decoding, the ALU and ADDER have a split mode in which two simultaneous 16-bit additions or subtractions are performed. This mode, available in specialized dual-MAC instructions, is used to compute the distance between a received symbol and its estimate.

The ACS provides the add/compare/select function required for Viterbi decoding. This unit provides flags to the traceback encoder for implementing mode-controlled side-effects for ACS operations. The source operands for the ACS are any two accumulators, and results are written back to one of the source accumulators.

The BMU implements barrel-shift, bit-field insertion, bit-field extraction, exponent extraction, normalization, and accumulator shuffling operations. **ar0** through **ar3** are auxiliary registers whose main function is to control BMU operations.

The user can enable overflow saturation to affect the multiplier output and the results of the three arithmetic units. Overflow saturation can also affect an accumulator value as it is transferred to memory or to another register. These features accommodate various speech coding standards such as GSM-FR, GSM-HR, and GSM-EFR. Shifting in the arithmetic pipeline occurs at several stages to accommodate various standards for mixed-precision and double-precision multiplications.

8 Digital Signal Processor (DSP) Block (continued)

8.2 DSP16000 Core Architectural Overview (continued)

The DAU contains control and status registers **auc0**, **auc1**, **psw0**, **psw1**, **vsw**, and **c0**—**c2**.

The arithmetic unit control registers **auc0** and **auc1** select or deselect various modes of DAU operation. These modes include scaling of products, saturation on overflow, feedback to the **x** and **y** registers from accumulators **a6** and **a7**, simultaneous loading of **x** and **y** registers with the same value (used for single-cycle squaring), and clearing the low half of registers when loading the high half to facilitate fixed-point operations.

The processor status word registers **psw0** and **psw1** contain flags set by ALU/ACS, ADDER, or BMU operations. They also include information on the current status of the interrupt controller.

The **vsw** register is the Viterbi support word associated with the traceback encoder. The traceback encoder is a specialized block for accelerating Viterbi decoding. The **vsw** controls side-effects for three compare functions: **cmp0( )**, **cmp1( )**, and **cmp2( )**. These instructions are part of the MAC group that utilizes the traceback encoder. The side-effects allow the DAU to store, with no overhead, state information necessary for traceback decoding. Side-effects use the **c1** counter, the **ar0** and **ar1** auxiliary registers, and bits 1 and 0 of **vsw**.

The **c1** and **c0** counters are 16-bit signed registers used to count events such as the number of times the program has executed a sequence of code. The **c2** register is a holding register for counter **c1**. Conditional instructions control these counters and provide a convenient method of program looping.

8.2.3 Y-Memory Space Address Arithmetic Unit (YAAU)

The YAAU supports high-speed, register-indirect, data memory addressing with postincrement of the address register. Eight 20-bit pointer registers (**r0**—**r7**) store read or write addresses for the data (Y-memory) space. Two sets of 20-bit registers (**rb0** and **re0**; **rb1** and **re1**) define the upper and lower boundaries of two zero-overhead circular buffers for efficient filter implementations. The **j** and **k** registers are two 20-bit signed registers that are used to hold user-defined postincrement values for **r0**—**r7**. Fixed increments of +1, –1, 0, +2, and –2 are also available. (Postincrement options 0 and –2 are not available for some specialized transfers. See the *DSP16000 Digital Signal Processor Core* Information Manual for details.)

The YAAU includes a 20-bit stack pointer (**sp**). The data move group includes a set of stack instructions that consists of push, pop, stack-relative, and pipelined stack-relative operations. The addressing mode used for the stack-relative instructions is register-plus-displacement indirect addressing (the displacement is optional). The displacement is specified as either an immediate value as part of the instruction or a value stored in **j** or **k**. The YAAU computes the address by adding the displacement to **sp** and leaves the contents of **sp** unchanged. The data move group also includes instructions with register-plus-displacement indirect addressing for the pointer registers **r0**—**r6** in addition to **sp**.

The data move group of instructions includes instructions for loading and storing any YAAU register from or to memory or another core register. It also includes instructions for loading any YAAU register with an immediate value stored with the instruction. The pointer arithmetic group of instructions allows adding of an immediate value or the contents of the **j** or **k** register to any YAAU pointer register and storing the result to any YAAU register.

8.2.4 X-Memory Space Address Arithmetic Unit (XAAU)

The XAAU contains registers and an adder that control the sequencing of instructions in the processor. The program counter (**PC**) automatically increments through the instruction space. The interrupt return register **pi**, the subroutine return register **pr**, and the trap return register **ptrap** are automatically loaded with the return address of an interrupt service routine, subroutine, and trap service routine, respectively. High-speed, register-indirect, read-only memory addressing with postincrementing is done with the **pt0** and **pt1** registers. The signed registers **h** and **i** are used to hold a user-defined signed postincrement value. Fixed postincrement values of 0, +1, –1, +2, and –2 are also available. (Postincrement options 0 and –2 are available only if the target of the data transfer is an accumulator. See the *DSP16000 Digital Signal Processor Core* Information Manual for details.)

The data move group includes instructions for loading and storing any XAAU register from or to memory or another core register. It also includes instructions for loading any XAAU register with an immediate value stored with the instruction.

vbase is the 20-bit vector base offset register. The user programs this register with the base address of the interrupt and trap vector table.

8 Digital Signal Processor (DSP) Block (continued)

8.2 DSP16000 Core Architectural Overview (continued)

8.2.5 Core Block Diagram

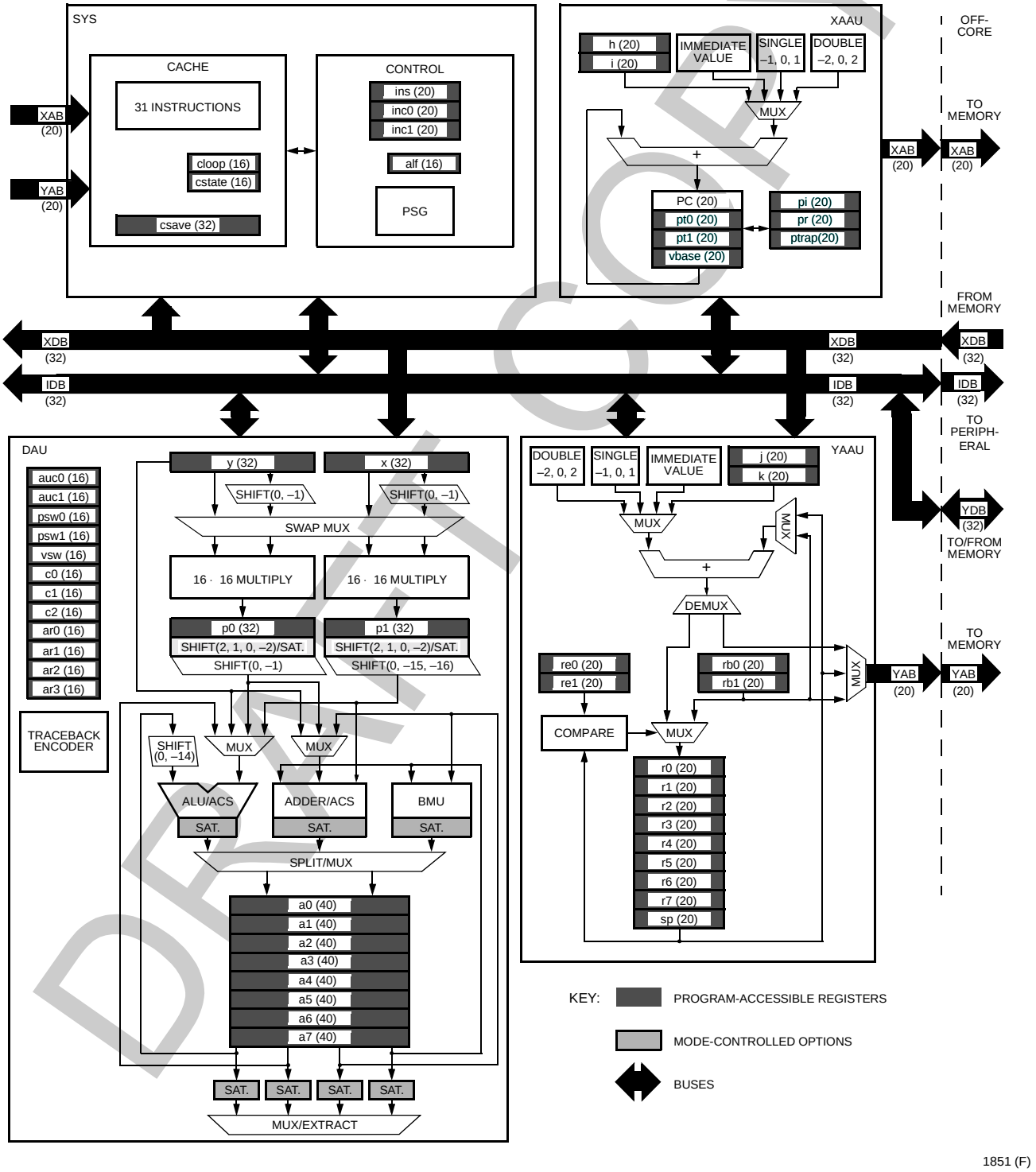


Figure 8.2-1 DSP16000 Core Block Diagram

8 Digital Signal Processor (DSP) Block (continued)

8.2 DSP16000 Core Architectural Overview (continued)

Table 8.2-1 DSP16000 Core Block Diagram Legend

Symbol	Name
16 x 16 MULTIPLY	16-bit x 16-bit multiplier.
a0—**a7**	40-bit accumulators 0—7.
ADDER/ACS	3-input 40-bit adder/subtractor and add/compare/select function. Used in Viterbi decoding.
alf	16-bit AWAIT low-power and flags register.
ALU/ACS	40-bit arithmetic logic unit and add/compare/select function. Used in Viterbi decoding.
ar0—**ar3**	16-bit auxiliary registers 0—3.
auc0, **auc1**	16-bit arithmetic unit control registers.
BMU	40-bit manipulation unit.
c0, **c1**	16-bit counters 0 and 1.
c2	16-bit counter holding register.
cloop	16-bit cache loop count register.
COMPARE	Comparator. Used for circular buffer addressing.
csave	32-bit cache save register.
cstate	16-bit cache state register.
DAU	Data arithmetic unit.
h	20-bit pointer postincrement register for the X-memory space.
i	20-bit pointer postincrement register for the X-memory space.
IDB	32-bit internal data bus.
inc0, **inc1**	20-bit interrupt control registers 0 and 1.
ins	20-bit interrupt status register.
j	20-bit pointer postincrement/offset register for the Y-memory space.
k	20-bit pointer postincrement/offset register for the Y-memory space.
MUX	Multiplexer.
p0, **p1**	32-bit product registers 0 and 1.
PC	20-bit program counter.
pi	20-bit subroutine interrupt return register.
pr	20-bit subroutine return register.
PSG	Pseudorandom Sequence Generator.
psw0, **psw1**	16-bit processor status word registers 0 and 1.
pt0, **pt1**	20-bit pointers 0 and 1 to X-memory space.
ptrap	20-bit subroutine trap return register.
r0—**r7**	20-bit pointers 0—7 to Y-memory space.
rb0, **rb1**	20-bit circular buffer pointers 0 and 1 (begin address).
re0, **re1**	20-bit circular buffer pointers 0 and 1 (end address).
SAT	Saturation.
SHIFT	shifting operation.
sp	20-bit stack pointer.
SPLIT/MUX	Split/multiplexer. Routes the appropriate ALU/ACS, BMU, and ADDER/ACS outputs to the appropriate accumulator.
SWAP MUX	Swap multiplexer. Routes the appropriate data to the appropriate multiplier input.
SYS	System control and cache.

8 Digital Signal Processor (DSP) Block (continued)

8.2 DSP16000 Core Architectural Overview (continued)

Table 8.2-1 DSP16000 Core Block Diagram Legend (continued)

Symbol	Name
vbase	20-bit vector base offset register.
vsw	16-bit viterbi support word. Associated with the traceback encoder.
x	32-bit multiplier input register.
XAAU	X-memory space address arithmetic unit.
XAB	X-memory space address bus.
XDB	X-memory space data bus.
y	32-bit multiplier input register.
YAAU	Y-memory space address arithmetic unit.
YAB	Y-memory space address bus.
YDB	Y-memory space data bus.

8 Digital Signal Processor (DSP) Block (continued)

8.3 DSP Software Architecture

8.3.1 Software Patch Unit

The DSP core in T8307 DSP block has mask-programmable internal ROM. The internal ROM of the DSP contains voice code specific to VoIP application including G.711 ∝-law, G.711 A-law, G.723, G.729A, and G.729B standards. Since the program cannot be changed once the device is manufactured, a software patch unit has been built into each DSP to allow it to patch sections of ROM code with code stored in another segment of memory.

The software patch unit provides the ability to patch up to 16 sections of code. All 16 patches are controlled through the DSP program via the **patchc** register. This register sets, enables, and disables the software patches. The **patchc** register is write only. When reset is applied, the **patchc** register is disabled from detecting addresses. Each location in the register must be reprogrammed and enabled for that address to be detected.

Each patch address will cause a trap in the DSP, and program control to branch to one of 16 corresponding trap vectors. These trap vectors are offsets from the value contained in the **vbase** register.

Table 8.15-17 shows the bit fields in the **patchc** register. A write to this register will either set or clear addresses, depending on the value in bit 31.

8.3.1.1 Programming the Software Patch Unit

Here is an example of how the software patch unit is programmed using the **patchc** register. To simplify some of the notation, preprocessor macros are used.

```
                        //   S=1      |  patch select   |  patch address
   #define  SET(pn, addr)(0x80000000   |  (pn << 27)     |  addr)
   #define  CLR(pn)     (pn << 27)
   // set patches on ROM code addresses
   // identified by assembler labels
   a0 = SET(0, ADDRESS_0)
   patchc = a0
   a0 = SET(1, ADDRESS_1)
   patchc = a0
. . .
   a0 = SET(15, ADDRESS_15)
   patchc = a0
   // a specific patch can be disabled as follows
   a0 = CLR(15)
   patchc = a0
```

After these assembler instructions have executed, the ADDRESS_0 and ADDRESS_1 have been patched. The ADDRESS_15 is no longer patched. When program execution reaches ADDRESS_0 or ADDRESS_1, a trap will be taken, and a jump to the appropriate vector will be executed. The trap handler can then execute the patch code. When execution reaches ADDRESS_15, the code at that address will be executed normally.

8 Digital Signal Processor (DSP) Block (continued)

8.3 DSP Software Architecture (continued)

8.3.1.2 Patch Vectors in the Vectored Interrupt Table

When the address of execution matches the value in one of the 16 patch locations, execution branches to an address corresponding to the sum of the value in the vbase register and the offset associated with that patch location. While vbase is user programmable, these offsets are constant, and they are summarized in Table 8.3-1.

Table 8.3-1 Offset Locations for T8307 DSP Block

Offset from vbase	Source
0x00120	PATCH 0/ICALL 48
0x00124	PATCH 1/ICALL 49
0x00128	PATCH 2/ICALL 50
0x0012C	PATCH 3/ICALL 51
0x00130	PATCH 4/ICALL 52
0x00134	PATCH 5/ICALL 53
0x00138	PATCH 6/ICALL 54
0x0013C	PATCH 7/ICALL 55
0x00140	PATCH 8/ICALL 56
0x00144	PATCH 9/ICALL 57
0x00148	PATCH 10/ICALL 58
0x0014C	PATCH 11/ICALL 59
0x00150	PATCH 12/ICALL 60
0x00154	PATCH 13/ICALL 61
0x00158	PATCH 14/ICALL 62
0x0015C	PATCH 15/ICALL 63

Note: The 16 patch vectors are shared by 16 of the 63 icall (software interrupt) vectors. The programmer must take care not to use the same vector for both features. There are additional precautions to be taken when using the software patch feature and icall in the same program (see Section 8.3.1.4).

8 Digital Signal Processor (DSP) Block (continued)

8.3 DSP Software Architecture (continued)

Below is an example of how the vector interrupt table can be used with the software patch feature:

```
      vbase = interruptVectorTable
                  . . .
interruptVectorTable:
                  . . .
       .=.+0x00120                  // starting address of patch vectors
      ptrap=patchCode0;      nop;     treturn// four words at location 0x00120
      ptrap=patchCode1;      nop;     treturn// four words at location 0x00124
                  . . .
```

In the code above, the value in ptrap (the patched address) is replaced with a label at the start of the code to be executed in place of the original code.

8.3.1.3 The Patching Code

Continuing with the example, the program fragment between patchCode0 and goto returnAddress below would be executed instead of the code at iromLabel0:

```
patchCode0:
      . . .                        // some instructions patching the original code
      far goto returnAddress       // must NOT be the same as the patched address
```

As the comments on the right of the code indicate, it is critical that the return address not be the same as the address that was stored in the software patch unit. This would result in an endless loop.

8.3.1.4 Software Patch, Interrupts, Traps, and the icall Instruction

As mentioned above, the software patch shares its 16 vectors with 16 of the 63 that are devoted to the icall instruction. For this reason, patch locations and the last 16 icall vectors that share the same offset into the vector interrupt table should not be used at the same time.

The software patch has the same priority as a trap; therefore, it takes precedence over all interrupts. The icall (software interrupt) instruction, however, is an exception. Using the software patch on the same address as an icall instruction can cause unpredictable behavior. For this reason, programmers should avoid using these two features together on the same address.

Since icall is an instruction in memory like any other, if a programmer wished to patch this code, it could be done safely by patching an address preceding the icall and returning to some location following the icall instruction.

8.3.2 DSP Reset States

DSP reset occurs if a high-to-low logic transition is applied to the RESETN pin or if the DRESETN bit of the DCCON register is reset. For a list of DSP register reset states, see Section 8.15.3.

8 Digital Signal Processor (DSP) Block (continued)

8.4 Interrupts and Traps

The DSP16000 core in T8307 DSP block supports the following interrupts and traps:

- 4 hardware interrupts with three levels of user-assigned priority:
  - 1 timer interrupt.
  - 1 external interrupt pin.
  - 1 ICP interrupt.
  - 1 SSP/I^2S interrupt.
- 64 software interrupts, generated by the execution of an **icall IM6** instruction.

The interrupt and trap vectors are in contiguous locations in memory, and the base (starting) address of the vectors is configurable in the core's **vbase** register. Each interrupt and trap source is preassigned to a unique vector offset that differentiates its service routine.

The core must reach an interruptible or trappable state (completion of an interruptible or trappable instruction) before it services an interrupt or trap. If the core services an interrupt or trap, it saves the contents of its program counter (**PC**) and begins executing instructions at the corresponding location in its vector table. For interrupts, the core saves its **PC** in its program interrupt (**pi**) register. For traps, the core saves its **PC** in its program trap (**ptrap**) register. After servicing the interrupt or trap, the servicing routine must return to the interrupted or trapped program by executing an **ireturn** or **treturn** instruction.

The core's **ins** register (see Table 8.15-16) contains a 1-bit status field for each of its hardware interrupts. If a hardware interrupt occurs, the core sets the corresponding **ins** field to indicate that the interrupt is pending. If the core services that interrupt, it clears the corresponding **ins** field. The **psw1** register (see Table 8.15-22) includes control and status bits for the core's hardware interrupt logic.

If a hardware interrupt is disabled, the core does not service it. If a hardware interrupt is enabled, the core services it according to its priority. Device reset globally disables hardware interrupts. An application can globally enable or disable hardware interrupts and can individually enable or disable each hardware interrupt. An application globally enables hardware interrupts by executing the **ei** (enable interrupts) instruction and globally disables them by executing the **di** (disable interrupts) instruction. An application can individually enable a hardware interrupt at an assigned priority, or individually disable a hardware interrupt by configuring the **inc0** or **inc1** register (see Table 8.15-15).

Software interrupts emulate hardware interrupts for the purpose of software testing. The core services software interrupts even if hardware interrupts are globally disabled.

A trap is similar to an interrupt but has the highest possible priority. An application cannot disable traps by executing a **di** instruction or by any other means. Traps do not nest (i.e., a trap service routine (TSR) cannot be interrupted or trapped). A trap does not affect the state of the **psw1** register.

The *DSP16000 Digital Signal Processor Core* Information Manual provides an extensive discussion of interrupts and traps. The remainder of this section describes the interrupts and traps for the DSP16000 core in T8307 DSP block.

8 Digital Signal Processor (DSP) Block (continued)

8.4 Interrupts and Traps (continued)

8.4.1 Clearing Core Interrupt Requests

Internal hardware interrupt signals are pulses that the core latches into its **ins** register (see Section 8.4.4). Therefore, the user software need not clear the interrupt request.

8.4.2 Globally Enabling and Disabling Hardware Interrupts

A device reset globally disables interrupts (i.e., the core does not service interrupts by default after reset). The application must execute an **ei** instruction to globally enable interrupts (i.e., to cause the core to service interrupts that are individually enabled). Section 8.4.3 describes individually enabling and disabling interrupts. Executing the **di** instruction globally disables interrupts.

The core automatically globally disables interrupts if it begins servicing an interrupt. Therefore, an interrupt service routine (ISR) cannot be interrupted unless the programmer places an **ei** instruction within the ISR. In other words, interrupt nesting is disabled by default. When the **ireturn** instruction that the programmer must place at the end of the ISR is executed, the core automatically globally re-enables interrupts. Therefore, the programmer does not need to explicitly re-enable interrupts by executing an **ei** instruction before exiting the ISR. To nest interrupts, the programmer must place an **ei** and a **di** instruction within an ISR. See Section 8.4.8 for details on nesting.

The 1-bit IEN field (**psw1**[14]—see Table 8.15-22) is cleared if hardware interrupts are globally disabled. The IEN field is set if interrupts are globally enabled.

Table 8.4-1 summarizes global disabling and enabling of hardware interrupts.

Table 8.4-1 Global Disabling and Enabling of Hardware Interrupts

Condition	Caused by	Indicated By	Effect
Hardware interrupts globally disabled	■ Device reset ■ Execution of a **di** instruction ■ The core begins to service an interrupt	IEN (**psw1**[14]) = 0	Core does not service interrupts.
Hardware interrupts globally enabled	■ Execution of an **ei** instruction ■ Execution of an **ireturn** instruction	IEN (**psw1**[14]) = 1	Core services individually enabled interrupts.

8.4.3 Individually Enabling, Disabling, and Prioritizing Hardware Interrupts

An application can individually disable a hardware interrupt by clearing both bits of its corresponding 2-bit field in the **inc0** or **inc1** register (see Table 8.15-15). Reset clears the **inc0** and **inc1** registers, individually disabling all hardware interrupts by default. An application can individually enable a hardware interrupt at one of three priority levels by setting one or both bits of its corresponding 2-bit field in the **inc0** or **inc1** register.

The following are the advantages of interrupt prioritization:

- An ISR can service concurrent interrupts according to their priority.
- Interrupt nesting is supported (i.e., an interrupt can interrupt a lower-priority ISR). See Section 8.4.8 for details on interrupt nesting.

If multiple concurrent interrupts with the same assigned priority occur, the core first services the interrupt that has its status field in the relative least significant bit location of the **ins** register (see Table 8.15-16), i.e., the core first services the interrupt with the lowest vector address (see Table 8.4-2).

Note: If interrupts are globally enabled (see Section 8.4.2), an application must not change **inc**⟨**0—1**⟩. Prior to changing **inc**⟨**0—1**⟩, the application must globally disable interrupts by executing a **di** instruction. After changing **inc**⟨**0—1**⟩, the application can globally re-enable interrupts by executing an **ei** instruction.

8 Digital Signal Processor (DSP) Block (continued)

8.4 Interrupts and Traps (continued)

8.4.4 Hardware Interrupt Status

If a hardware interrupt occurs, the core sets the corresponding bit in the **ins** register (Table 8.15-16) to indicate that the interrupt is pending. If the core services the interrupt, it clears the **ins** bit. Alternatively, if the application uses interrupt polling (Section 8.4.9), the application program must explicitly clear the **ins** bit by writing a 1 to that bit and a 0 to every other **ins** bit. Writing a 0 to an **ins** bit leaves that bit unchanged. A reset clears the **ins** register, indicating that no interrupts are pending.

If a hardware interrupt occurs, the core sets its **ins** bit (i.e., latches the interrupt as pending) regardless of whether the interrupt is enabled or disabled. If a hardware interrupt occurs while it is disabled and the interrupt is later enabled, the core services the interrupt after servicing any other pending interrupts of equal or higher priority.

Note: The DSP core globally disables interrupts when it begins executing instructions in the vector table. If the ISR does not globally enable interrupts by executing **ei**, and the same interrupt reoccurs while the core is executing the ISR, the interrupt is not latched into **ins** and is, therefore, not recognized by the core.

8.4.5 Interrupt and Trap Vector Table

The interrupt and trap vectors for the core are in contiguous locations in memory. The base (starting) address of the vectors is configurable in the core's **vbase** register. Each interrupt and trap source is preassigned to a unique vector offset within a 352-word vector table (see Table 8.4-2). The programmer can place an instruction at the vector location that branches to an interrupt service routine (ISR) or trap service routine (TSR). After servicing the interrupt or trap, the ISR or TSR must return to the interrupted or trapped program by executing an **ireturn** or **treturn** instruction. Alternatively, the programmer can place up to four words of instructions at the vector location that service the interrupt or trap, the last of which must be an **ireturn** or **treturn**.

8 Digital Signal Processor (DSP) Block (continued)

8.4 Interrupts and Traps (continued)

Table 8.4-2 Interrupt and Trap Vector Table

Vector Description	Vector Address*		Priority
	Hexadecimal	Decimal	
Reserved	**vbase** + 0x0	**vbase** + 0	—
Reserved	**vbase** + 0x4	**vbase** + 4	—
UTRAP†	**vbase** + 0x8	**vbase** + 8	5 (Highest)
Reserved	**vbase** + 0xC	**vbase** + 12	—
TIMER	**vbase** + 0x10	**vbase** + 16	0—3‡
Reserved	**vbase** + 0x14	**vbase** + 20	—
Reserved	**vbase** + 0x18	**vbase** + 24	—
Reserved	**vbase** + 0x1C	**vbase** + 28	—
Reserved	**vbase** + 0x20	**vbase** + 32	—
Reserved	**vbase** + 0x24	**vbase** + 36	—
Reserved	**vbase** + 0x28	**vbase** + 40	—
Reserved	**vbase** + 0x2C	**vbase** + 44	—
INT0	**vbase** + 0x30	**vbase** + 48	0—3
Reserved	**vbase** + 0x34	**vbase** + 52	—
SSP	**vbase** + 0x38	**vbase** + 56	0—3
Reserved	**vbase** + 0x3C	**vbase** + 60	—
Reserved	**vbase** + 0x40	**vbase** + 64	—
ICP	**vbase** + 0x44	**vbase** + 68	0—3
Reserved	**vbase** + 0x48	**vbase** + 72	—
Reserved	**vbase** + 0x4C	**vbase** + 76	—
Reserved	**vbase** + 0x50	**vbase** + 80	—
Reserved	**vbase** + 0x54	**vbase** + 84	—
Reserved	**vbase** + 0x58	**vbase** + 88	—
Reserved	**vbase** + 0x5C	**vbase** + 92	—
icall 0§	**vbase** + 0x60	**vbase** + 96	—
icall 1	**vbase** + 0x64	**vbase** + 100	—
⋮	⋮	⋮	—
icall 62	**vbase** + 0x158	**vbase** + 344	—
icall 63	**vbase** + 0x15C	**vbase** + 348	—

* **vbase** contains the base address of the 352-word vector table.

† Reserved for HDS.

‡ The programmer specifies the relative priority levels 0—3 for hardware interrupts via **inc0** and **inc1** (see Table 8.15-15). Level 0 indicates a disabled interrupt. If multiple concurrent interrupts with the same assigned priority occur, the core first services the interrupt that has its status field in the relative least significant bit location of the **ins** register (see Table 8.15-16); i.e., the core first services the interrupt with the lowest vector address.

§ Reserved for system services.

8 Digital Signal Processor (DSP) Block (continued)

8.4 Interrupts and Traps (continued)

8.4.6 Software Interrupts

Software interrupts emulate hardware interrupts for the purpose of software testing. A software interrupt is always enabled and has no assigned priority and no corresponding field in the **ins** register. A program causes a software interrupt by executing an **icall IM6** instruction, where IM6 is replaced with 0—63. When a software interrupt is serviced, the core saves the contents of **PC** in the **pi** register and transfers control to the interrupt vector defined in Table 8.4-2.

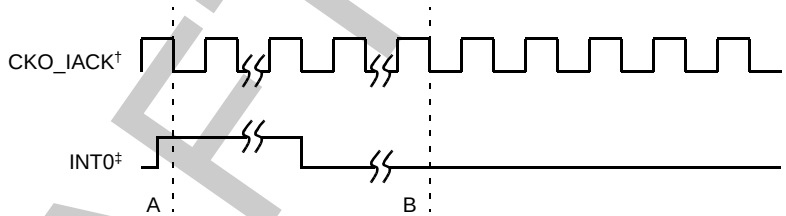
CAUTION: If a software interrupt is inserted into an ISR, it is explicitly nested in the ISR and, therefore, the ISR must be structured for nesting. See Section 8.4.8 and the *DSP16000 Digital Signal Processor Core* Information Manual for more information about nesting.

8.4.7 INT0

T8307 DSP block provides a positive-assertion level-sensitive interrupt pin (INT0).

Figure 8.4-1 is a functional timing diagram for the INT0 pin. A low-to-high transition of INT0 pin asserts the corresponding interrupt. INT0 must be held high for a minimum of two CLK cycles. T8307 synchronizes INT0 on the falling edge of the internal clock CLK.

A minimum of four cycles* after INT0 is asserted, the core services the interrupt by executing instructions starting at the vector location as defined in Table 8.4-2. The exact number of cycles between INT0 assertion and interrupt service depends on the number of wait-states incurred by the interrupted instruction.



† CKO_IACK is programmed to be the internal clock CLK, i.e., the CKO_SEL field (**IOC**[7:5]) is programmed to 0.

Figure 8.4-1 Functional Timing for INT0

8 Digital Signal Processor (DSP) Block (continued)

8.4 Interrupts and Traps (continued)

8.4.8 Nesting Interrupts

The **psw1** register (see Table 8.15-22) contains the IPLC[1:0] and IPLP[1:0] fields that are used for interrupt nesting. See the *DSP16000 Digital Signal Processor Core* Information Manual for details on these fields.

The core automatically globally disables interrupts when it begins servicing an interrupt. Therefore, an interrupt service routine (ISR) cannot be interrupted unless the programmer places an **ei** (enable interrupts) instruction within the ISR. In other words, interrupt nesting is disabled by default. To allow nesting, the ISR must perform the following steps before executing an **ei** instruction:

1. Copy the contents of **psw1** and **pi** to memory. This is needed to save the previous interrupt priority level (IPLP) and the interrupt return address in **pi**, which are overwritten by the core if the ISR is interrupted.
2. Copy the contents of **cstate** to memory and then clear **cstate**. This is needed in case the ISR has interrupted a cache loop (**do** or **redo**). If the ISR is interrupted and **cstate** is not cleared, the nested interrupt's **ireturn** instruction will return to the cache instead of to the ISR. See the *DSP16000 Digital Signal Processor Core* Information Manual for details on **cstate** and the cache.

After performing steps 1 and 2, the ISR can safely globally enable interrupts by executing an **ei** instruction. After servicing the interrupt and before executing an **ireturn** instruction to return the core to its previous state before the interrupt occurred, the ISR must perform the following steps:

1. Globally disable interrupts via the **di** (disable interrupts) instruction. This is needed to ensure that the restoring step (see step 2) is not interrupted.
2. Restore **psw1**, **pi**, and **cstate** so that they contain their original values from the beginning of the ISR execution.

After performing the steps 1 and 2, the ISR can return the core to its previous state by executing an **ireturn** instruction. Executing **ireturn** globally enables interrupts, so it is not necessary for the ISR to explicitly enable interrupts by executing an **ei** instruction before returning.

See the *DSP16000 Digital Signal Processor Core* Information Manual for more detail on interrupt nesting.

8.4.9 Interrupt Polling

If a core disables an interrupt and tests its **ins** field, it can poll that interrupt instead of automatically servicing it. This procedure, however, costs in the amount of code that must be written and executed to replace what the DSP core does by design.

The programmer can poll an interrupt source by checking its pending status in **ins**. The program can clear an interrupt and change its status from pending to not pending by writing a 1 to its corresponding **ins** field. This clears the field and leaves the remaining fields of **ins** unchanged.

8.5 Bit Input/Output Units (BIO)

The BIO unit controls the two bidirectional control I/O pins, IOBIT[1:0]. If an IOBIT pin is configured as an output, it is individually set, cleared, or toggled. If a pin is configured as an input, it is read and/or tested. There are two registers (**sbit** and **cbit**) associated with the BIO.

The lower half of the **sbit** register (see Table 8.15-23) contains current values (VALUE[7:0]) of the two bidirectional pins IOBIT[1:0]. The upper half of the **sbit** register (DIREC[7:0]) controls the direction of each of the pins. A logic 1 configures the corresponding pin as an output; a logic 0 configures it as an input. The upper half of the **sbit** register is cleared upon reset.

The **cbit** register (see Table 8.15-9) contains two 8-bit fields, MODE/MASK[7:0] and DATA/PAT[7:0]. The values of DATA/PAT[7:0] are cleared upon reset. The meaning of a bit in either field depends on whether it has been configured as an input or an output in **sbit**. If a pin is configured to be an output, the meanings are MODE and DATA. For an input, the meanings are MASK and PATTERN. Table 8.15-9 shows the functionality of the MODE/MASK and DATA/PAT bits based on the direction selected for the associated IOBIT pin.

If a BIO pin is switched from being configured as an output to being configured as an input and then back to being configured as an output, the pin retains the previous output value.

8 Digital Signal Processor (DSP) Block (continued)

8.6 Timer Unit (TIMER)

8.6.1 Functional Description

Timer runs from two clocks. The CKI (system clock) is used to drive the timer counters; the CLKR (regional clock) is used to drive the read/write signals for timer module registers on the DSP peripheral bus. Timer consists of four subblocks: **Sync**, **Clkgen**, **Prescaler** and **Counter**.

Sync: It is used to synchronize the asynchronous signals from CLKR clock domain to CKI clock domain. It is implemented using the handshake mechanism (i.e., signal from the first clock domain is held high and is cleared by the signal from the second clock domain).

Clkgen: It gets 2 clocks CKI and CLKR, and generates gated clocks to prescaler, counter and rest of the circuit. All the powerdown and clock enable signals comes to this block. Prescaler and counter runs on the gated version of CKI.

Prescaler: It runs on the CKI and consists of a 16-bit up counter. This counter is cleared upon power on reset. It can also be cleared when PSRST bit in the **timerc** register is set to 1 and one of the following conditions is true:

- LTC bit is set to 0 and a new write occurs to **timer** register.
- LTC bit or RELOAD bit is set to 1 and counter reloads the value from period register after reaching 0.

The prescalar counter is incremented by 1 each clock cycle of CKI. Each bit (bit 0 to 15) of this counter is a clock division of input clock CKI by 2^{N+1} (N = 0 to 15). A 16-1 MUX is used to select one of the 16 bits of this up counter. Control signal for this MUX is 4 bit prescale from the **timerc** control register. Output of this MUX is retimed with CKI to prevent any glitch in the clock signal. This signal is used to generate a prescaled pulse of one CKI cycle.

Counter: It runs on the prescaled CKI derived from prescaler. Counter is cleared upon power-on reset. It is loaded asynchronously (i.e., even when the timer clock is off) with value stored in period register whenever a write occurs and LTC is set to 0.

If TOEN bit is set to 0 counter does nothing but just holds the current value.

If TOEN bit is set to 1, counter starts decrementing from the current value by 1 every prescaled clock cycle till it reaches zero. It then interrupts the DSP (provided interrupt is enabled) and stops until a new write occurs or either LTC or RELOAD bit is set to 1.

When either LTC or RELOAD is set to 1 counter automatically reloads the value from the period register after reaching 0. This mode is used to generate repeated interrupts.

Note: Control and write signals for **timer** and **timerc** are first synchronized to CKI, which causes a 2 or 3 cycles delay on the timer operation. DSP should allow at least 3 CKI cycles between writes to above register. To read the contents of these registers DSP should wait for 3 CKI cycles after a write to these registers. There is no additional delay for successive reads.

Table 8.6-1 explains different modes of operation of timer counter, assuming TOEN bit is set to 1.

Table 8.6-1 State Machine Description of Timer Counter

Reload	LTC	Write to timer	Description
0	0	no write	Stop after counting down to 0.
0	0	write	Load new value immediately and count down to 0.
0	1	no write	Restart from old value after counting down to 0.
0	1	write	Restart from new value after counting down to 0.
1	0	no write	Restart from old value after counting down to 0.
1	0	write	Load new value immediately and count down and restart after counting down to 0.
1	1	no write	Restart from old value after counting down to 0.
1	1	write	Complete counting down to 0 and restart with new value.

8 Digital Signal Processor (DSP) Block (continued)

8.6 Timer Unit (TIMER) (continued)

8.6.2 Registers

Timer consists of the following two registers:

- **timerc**: 16-bit control register that sets different modes of operation of timer counter.
- **timer**: 16-bit counter register that holds the value of period register or down counter.

Table 8.15-24 shows bit field definition of **timerc** register.

timer is a 16-bit counter register. It runs on the CLKR. This register is cleared upon power-on reset. Contents of **timer** register are copied to period register that runs on CKI. Period register is written with the synchronized version of **timer** write signal. When a read occurs to **timer** register, the contents of down counter are read back by default. To read the contents of period register, the user should set the PRDSEL bit to 1 in the **timerc** control register.

timerc is a 16-bit control register. It also runs on CLKR. It is cleared upon power-on reset. So all the control signals are set to 0, which causes timer counter and prescaler to power up by default on power-on reset. Control signals in the **timerc** register are synchronized to CKI. Contents of **timerc** register can be read back anytime.

The interval from writing to period register and occurrence of first interrupt is [**period** * 2^{N+1}/CKI].

Where: N is prescale value, **period** is value stored in period register.

8.6.3 Software Programming Sequence

- Write to **timer** register to initialize value in period register.
- Write to **timerc** register according to Table 8.15-24 to set different modes:
  - — Write any value in the PRESCALER to initialize the prescaler.
  - — Write 1 to TOEN bit to enable count down.
  - — Write 0 to DISABLE bit to enable counter and prescaler clock.
  - — Write 0 to LTC and RELOAD bit for single timer interrupt.
  - — Write 1 to either LTC or RELOAD or both for repeated timer interrupts.
  - — Write 1 to PSRST bit to reset the prescaler whenever down counter starts a new count cycle (either with a new value written to **timer** or when RELOAD or LTC bit causes **timer** to reload value form period register after the counter reaches 0).
  - — Write 0 to PSRST for not resetting prescaler (for backward compatibility with SC4 Timer).
  - — Write 1 to PRDSEL bit to read the period register.
  - — Write 0 to PRDSEL bit to read the down counter register.

8 Digital Signal Processor (DSP) Block (continued)

8.7 Synchronous Serial Port (SSP) with Inter IC Sound (I²S) Support

8.7.1 Operation Modes

T8307 DSP-side SSPI²S port supports all features of *ARM Primecell* PL022 (e.g., *Motorola* SPI, *Texas Instruments SSI*, and *National Semiconductor MICROWIRE*) and *Philips* I²S formats. See Section 8.7.3 through Section 8.7.5 for SPI, SSI, and I²S formats, and refer to *ARM PrimeCell* PL022 document for *MICROWIRE* format.

In SSP modes (SPI, SSI, and *MICROWIRE*), the DSP-side serial bus interface consists of the following four pins: SPTXD1_I2SD, SPRXD1, SPCLK1 and SPFS1. Dynamic master/slave switching capability is provided for SPI, SSI, MW modes because these modes use separate transmit and receive data pins. This feature allows the user to switch the function of SPTXD1_I2SD and SPRXD1 such that the SSPI²S port can be configured as master or slave without changing pin connections on the board. In particular, if this function is enabled (by setting DS bit field of SSPCR1 to 0, which is the default value) and if the slave mode is selected (by setting MS bit field of SSPCR1 to 1), the SPTXD1_I2SD pin is an input pin while the SPRXD1 pin is an output pin. See Table 8.7-1 for a summary of the input/output status for all options.

Table 8.7-1 Functions of the SSP Bus Interface Pins

MS (SSPCR1 Bit 2)	DS (SSPCR1 Bit 7)	SPTXD1_I2SD Pin	SPRXD1 Pin	SPFS1 Pin	SPCLK1 Pin
0 (default)	0 (default)	Output	Input	Output	Output
1	0	Input	Output	Input	Input
0	1	Output	Input	Output	Output
1	1	Output	Input	Input	Input

In I²S mode, the interface consists of three pins: SPTXD1_I2SD, SPCLK1 and SPFS1. The function of these pins are determined by the MS bit and the I²STX bit of SSPCR1 register as summarized in Table 8.7-2.

Table 8.7-2 Functions of the I²S Bus Interface Pins

MS (SSPCR1Bit 2)	I²STX (SSPCR1 Bit 6)	SPTXD1_I2SD Pin	SPFS1 Pin	SPCLK1 Pin
0 (default)	0(default)	Input	Output	Output
1	0	Input	Input	Input
0	1	Output	Output	Output
1	1	Output	Input	Input

In both SSP modes and I²S mode, the SSPI²S supports programmable data sizes of 4 bits to 16 bits. To ensure correct device operation, the maximum expected frequency of SPCLK1 should not exceed 1/12 of the DSP system clock frequency when SPCLK1 is configured as an input pin. In addition, the polarity of the clock signal to or from SPCLK1 pin are programmable through SSPCR1 register.

8 Digital Signal Processor (DSP) Block (continued)

8.7 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

8.7.2 Interrupts

The SSP/I^2S block generates DSP interrupt request (SSPINT, see Table 8.4-2 for DSP interrupt vector assignments) based on the status of transmit and receive FIFOs. Both the transmit and receive FIFOs are 16-bit wide, 8-location deep. DSP data written across the SBUS interface are stored in the transmit FIFO until read out by the transmit logic, while received data from the serial interface are stored in the receive FIFO until read out by the DSP across the SBUS interface.

The SSPINT is asserted if any of the four individual interrupts below are asserted and enabled. The status of the individual interrupt sources are maskable and can be read from SSPRIS and SSPMIS registers.

8.7.2.1 Receive FIFO Service Interrupt Request (SSPRXINTR)

The receive interrupt is asserted when there are four or more valid entries in the receive FIFO.

8.7.2.2 Transmit FIFO Service Interrupt Request (SSPTXINTR)

The transmit interrupt is asserted when there are four or less valid entries in the transmit FIFO. The transmitter interrupt SSPTXINTR is not qualified with the SSP enable signal, which allows operation in one of two ways. Data can be written to the transmit FIFO prior to enabling the SSPI^{2}S and the interrupts. Alternatively, the SSPI^{2}S and interrupts can be enabled so that data can be written to the transmit FIFO by an interrupt service routine.

8.7.2.3 Receive Overrun Interrupt Request (SSPRORINTR)

The receive overrun interrupt SSPORINTR is asserted when the FIFO is already full and an additional data frame is received, causing an overrun of the FIFO. Data is overwritten in the receive shift register but not the FIFO.

8.7.2.4 Time-Out Interrupt Request (SSPRTINTR)

The receive time-out interrupt is asserted when the receive FIFO is not empty and the SSPI^{2}S has remained idle for a fixed 32-bit period. This mechanism ensures that the user is aware that data is still present in the receive FIFO and requires servicing.

8 Digital Signal Processor (DSP) Block (continued)

8.7 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

8.7.3 SSI

To operate in SSI mode, set FRF bit field of control register 0 (SSPCR0) to binary 01. In master mode, SPCLK1 and SPFS1 are forced low, and the transmit data line SPTXD1_I2SD is 3-stated whenever the SSPI^{2}S is idle. Once the bottom entry of the transmit FIFO contains data, SPFS1 is pulsed high for one SPCLK1 clock period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of SPCLK1, the MSB of the 4-bit to 16-bit data frame is shifted out on SPTXD1_I2SD. Likewise, the MSB of the received data is shifted onto SPRXD1 by the off-chip serial slave device. Both the SSPI^{2}S and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each SPCLK1. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SPCLK1 after the LSB has been latched.

Figure 8.7-1 shows the SSI frame format for a single transmitted frame. The nSSPOE signal is the internal output enable control for transmit pin, which is SPTXD1_I2SD in this case.

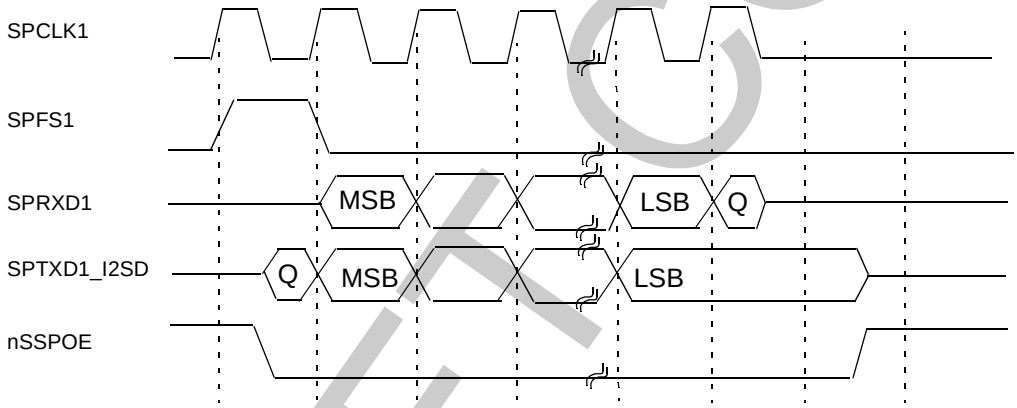


Figure 8.7-1 *Texas Instruments* Synchronous Serial Frame Format (Single Transfer)

Figure 8.7-2 shows the SSI frame format when back-to-back frames are transmitted

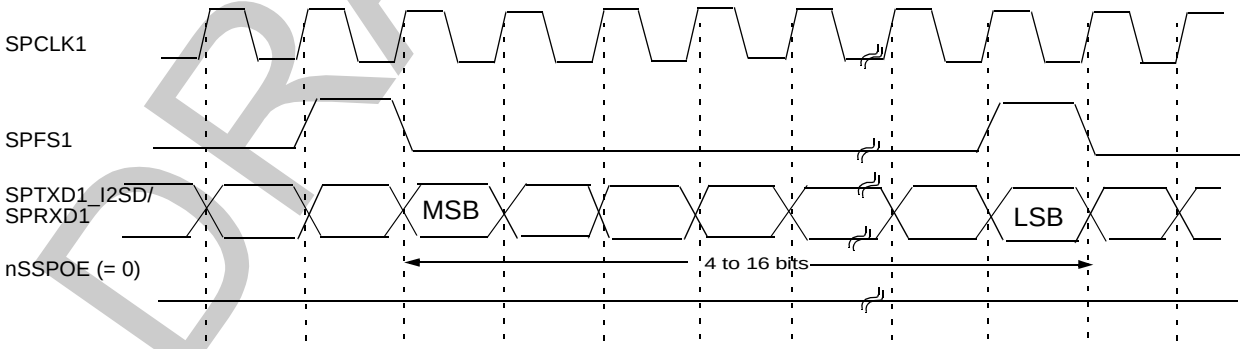


Figure 8.7-2 *Texas Instruments* Synchronous Serial Frame Format (Continuous Transfer)

8 Digital Signal Processor (DSP) Block (continued)

8.7 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

8.7.4 SPI

To operate in SPI mode, set FRF bit field of SSP control register 0 (SSPCR0) to binary 00. In this mode, the SPFS1 signal behaves as a slave select. Another feature is that the inactive state and phase of the SPCLK1 signal are programmable through the SPO and SPH bits within the SSPSCR0 control register.

When the SPO clock polarity control bit is low, it produces a steady state low value on SPCLK1. If the SPO clock polarity control bit is high, a steady-state high value is placed on SPCLK1 when data is not being transferred.

The SPH control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge.When the SPH phase control bit is low, data is captured on the first clock edge transition. If the SPH clock phase control bit is high, data is captured on the second clock edge transition.

8.7.4.1 *Motorola* SPI Format with SPO = 0, SPH = 0

Single and continuous transmission signal sequences for *Motorola* SPI format with SPO = 0, SPH = 0 are shown in Figure 8.7-3 and Figure 8.7-4.

In this configuration, during idle periods, the following occurs:

- The SPCLK1 pin is forced low in master mode, or high impedance in slave mode.
- SPFS1 is forced high.
- The transmit data line SPTXD1_I2SD is high impedance.

If the SSPI^{2}S is enabled and there is valid data in the transmit FIFO, the start of transmission is signified by the SPFS1 master signal being driven low. This causes slave data to be enabled onto the SPRXD1 line of the master. The master SSPTXD output is enabled. One-half SPCLK1 clock period later, valid master data is transferred to SPTXD1_I2SD. Now that both the master and slave data have been set, the SPCLK1 master clock goes high after one further half SPCLK1 period. The data is now captured on the rising and propagated on the falling edges of the SPCLK1 signal.

In the case of a single word transmission, after all bits of the data word have been transferred, the SPFS1 pin is returned to its idle high state one SPCLK1 period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SPFS1 signal must be pulsed high between each data word transfer. This is because the slave select signal freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is logic zero. Therefore, the master device must raise the SPFS1 signal for the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, SPFS1 is returned to its idle state one SPCLK1 period after the last bit has been captured.

8 Digital Signal Processor (DSP) Block (continued)

8.7 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

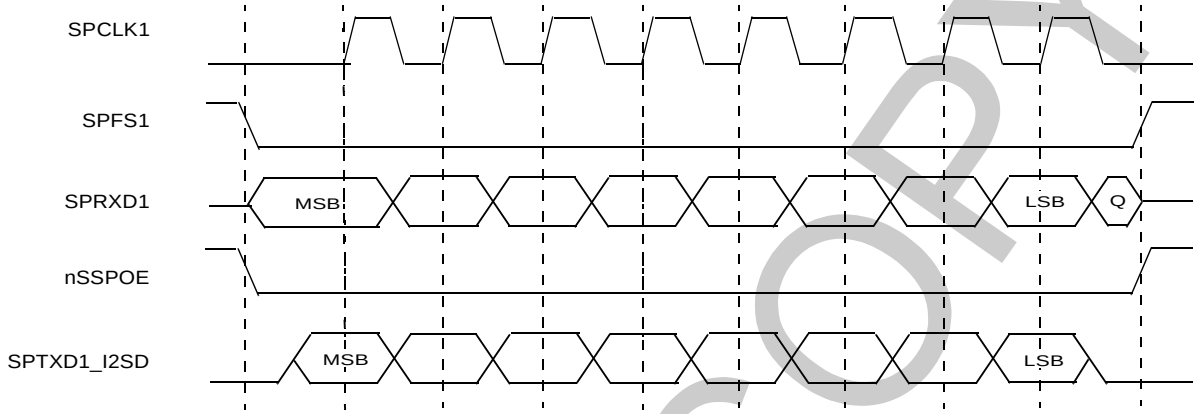


Figure 8.7-3 *Motorola* SPI Frame Format (Single Transfer) SPO = 0, SPH = 0

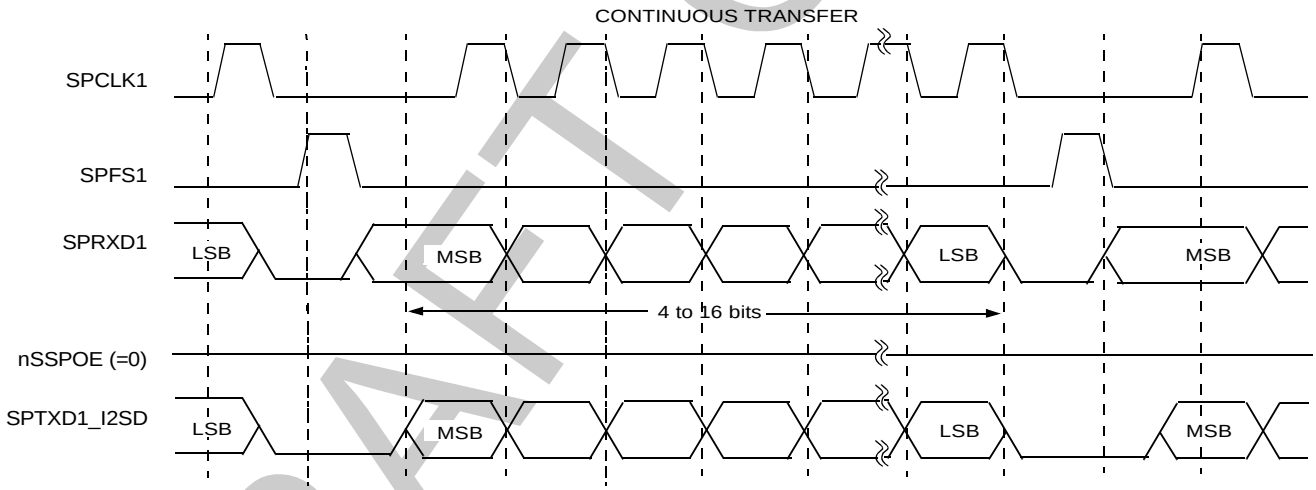


Figure 8.7-4 *Motorola* SPI Frame Format (Continuous Transfer) SPO = 0, SPH = 0

8 Digital Signal Processor (DSP) Block (continued)

8.7 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

8.7.4.2 *Motorola* SPI Format with SPO = 0, SPH = 1

The transfer signal sequence for *Motorola* SPI format with SPO = 0, SPH = 1 is shown in Figure 8.7-5, which covers both single and continuous transfers.

In this configuration, during idle periods, the following occurs:

- The SPCLK1 pin is forced low in master mode, or high impedance in slave mode.
- SPFS1 is forced high.
- The transmit data line SPTXD1_I2SD is high impedance.

If the SSPI^2S is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SPFS1 master signal being driven low. The nSSPOE line is driven low, enabling the master SPTXD1_I2SD output. After a further one-half SPCLK1 period, both master and slave valid data is enabled onto their respective transmission lines. At the same time, the SPCLK1 is enabled with a rising edge transition. Data is then captured on the falling edges and propagated on the rising edges of the SPCLK1 signal.

In the case of a single word transfer, after all bits have been transferred, the SPFS1 line is returned to its idle high state one SPCLK1 period after the last bit has been captured.

For continuous back-to-back transfers, SPFS1 is held low between successive data words and termination is the same as that of the single word transfer.

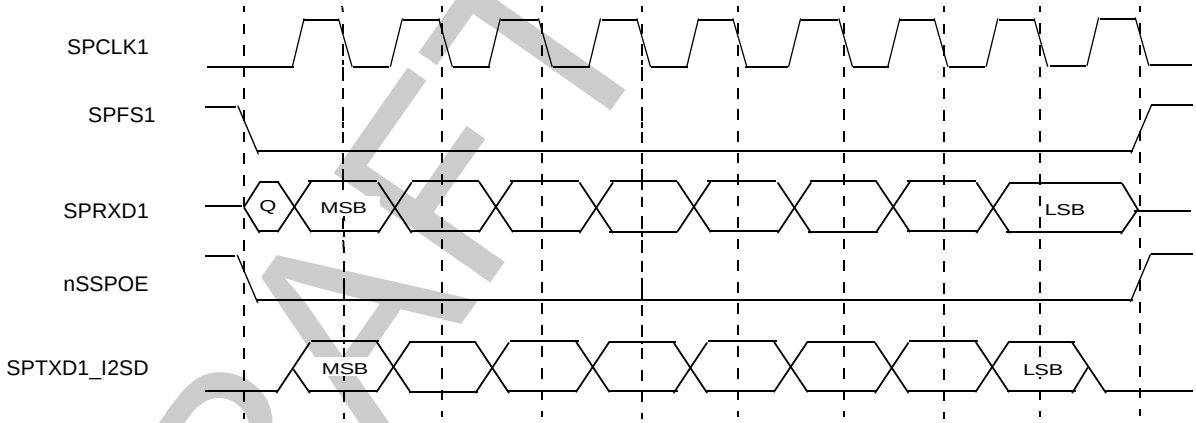


Figure 8.7-5 *Motorola* SPI Frame Format SPO = 0, SPH = 1

8 Digital Signal Processor (DSP) Block (continued)

8.7 Synchronous Serial Port (SSP) with Inter IC Sound (I²S) Support (continued)

8.7.4.3 *Motorola* SPI Format with SPO = 1, SPH = 0

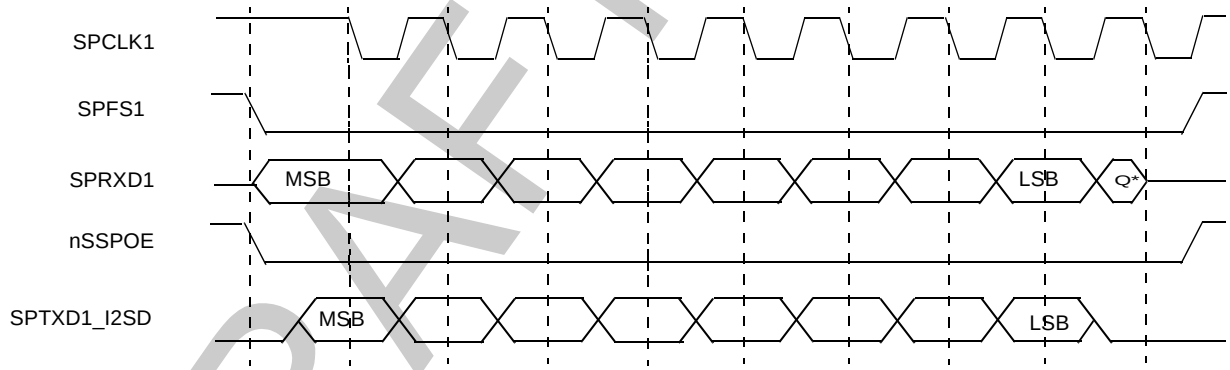
Single and continuous transmission signal sequences for *Motorola* SPI format with SPO = 1, SPH = 0 are shown in Figure 8.7-6 and Figure 8.7-7. In this configuration, during idle periods, the following occurs:

- The SPCLK1 pin is forced high in master mode, or high impedance in slave mode.
- SPFS1 is forced high.
- The transmit data line SPTXD1_I2SD is high impedance.

If the SSPI²S is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SPFS1 master signal being driven low, which causes slave data to be immediately transferred onto the SPRXD1 line of the master. The nSSPOE line is driven low, enabling the master SPTXD1_I2SD output. One-half period later, valid master data is transferred to the SPTXD1_I2SD line. Now that both the master and slave data have been set, the SPCLK1 master clock signal becomes low after one further half SPCLK1 period. This means that data is captured on the falling edges and be propagated on the rising edges of the SPCLK1 signal.

In the case of a single-word transmission, after all bits of the data word are transferred, the SPFS1 line is returned to its idle high state one SPCLK1 period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SPFS1 signal must be pulsed high between each data word transfer. This is because the slave select signal freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is logic 0. Therefore, the master device must raise the SPFS1 signal for the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SPFS1 signal is returned to its idle state one SPCLK1 period after the last bit has been captured.



* Q is an undefined signal.

Figure 8.7-6 *Motorola* SPI Frame Format (Single Transfer) with SPO = 1, SPH = 0

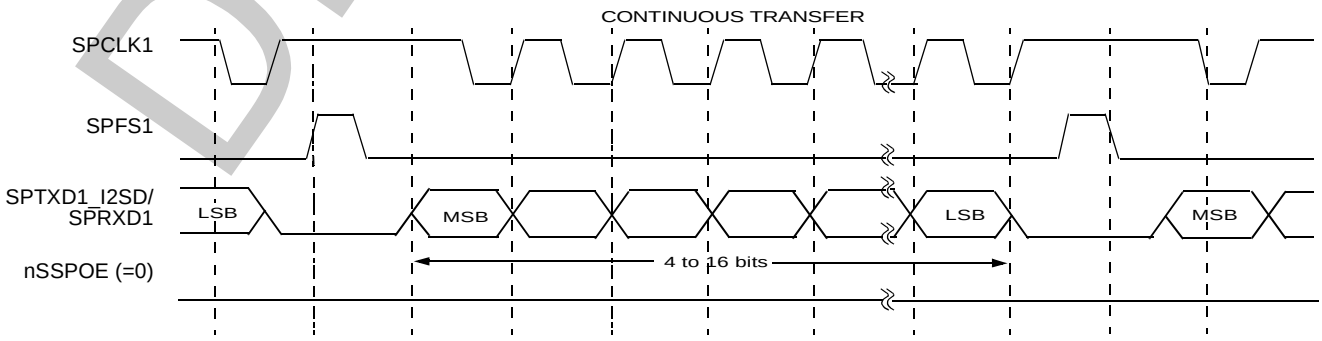


Figure 8.7-7 *Motorola* SPI Frame Format (Continuous Transfer) with SPO = 1, SPH = 0

8 Digital Signal Processor (DSP) Block (continued)

8.7 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

8.7.4.4 *Motorola* SPI Format with SPO = 1, SPH = 1

The transfer signal sequence for *Motorola* SPI format with SPO = 0, SPH = 1 is shown in Figure 8.7-8, which covers both single and continuous transfers. In this configuration, during idle periods, the following occurs:

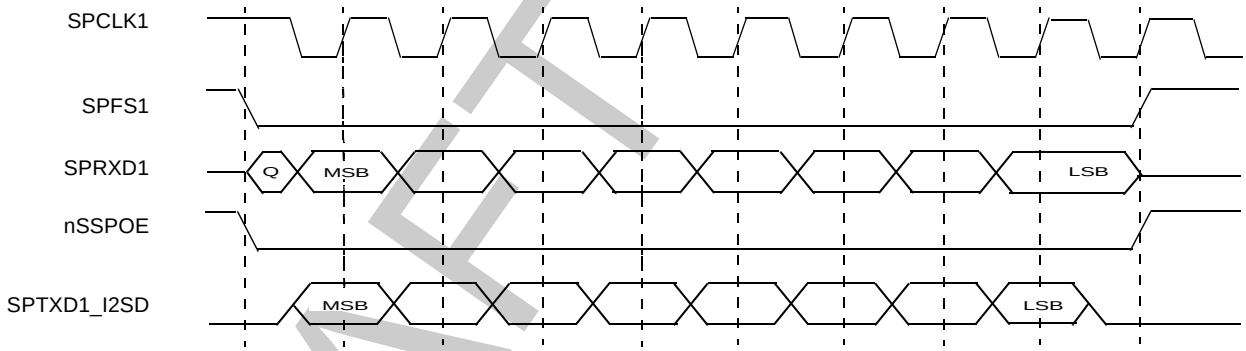
- The SPCLK1 pin is forced high in master mode, or high impedance in slave mode.
- SPFS1 is forced high.
- The transmit data line SPTXD1_I2SD is high impedance.

If the SSPI^{2}S is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SPFS1 master signal being driven low. The nSSPOE line is driven low, enabling the master SPTXD1_I2SD output. After a further one-half SPCLK1 period, both master and slave data are enabled onto their respective transmission lines. At the same time, the SPCLK1 is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SPCLK1 signal.

After all bits have been transferred, in the case of a single word transmission, the SPFS1 line is returned to its idle high state one SPCLK1 period after the last bit has been captured.

For continuous back-to-back transmissions, the SPFS1 signal remains in its active-low state, until the final bit of the last word has been captured, and then returns to its idle state as described above.

For continuous back-to-back transfers, the SPFS1 signal is held low between successive data words and termination is the same as that of the single word transfer.



* Q is an undefined signal.

Figure 8.7-8 *Motorola* SPI Frame Format with SPO = 1, SPH = 1

8 Digital Signal Processor (DSP) Block (continued)

8.7 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

8.7.5 I^2S

To operate in I^2S mode, set FRF bit field of control register 0 (SSPCR0) to binary 11.

In I^2S mode, the serial interface consists of three pins. The SPCLK1 pin and SPFS1 pin are the clock line and the word select line, respectively. The SPTXD1_I2SD pin is used for time-multiplexed left/right audio data channels, while the word select line SPFS1 also acts as the left/right channel select.

The device that generates the serial clock and word select is the master.

In master mode, SPCLK1 and SPFS1 are forced low, and the transmit data line SPTXD1_I2SD is high impedance whenever the SSPI^{2}S is idle. The idle state of SPCLK1 is utilized by the receiver to provide a receive time-out indication that occurs when the receive FIFO still contains data after a time-out period. Once the transmit FIFO contains some data, SPFS1 is synchronized to the trailing edge of SPCLK1 and the value to be transmitted is shifted from transmit FIFO to the serial shifter. On the next falling edge of SPCLK1, the MSB of the data word is shifted out on SPTXD1_I2SD.

In slave mode, the SPCLK1 input signal generated by external master is double synchronized and then delayed to detect an edge. It takes three DSP system clocks to detect an edge on SPCLK1. The MSB of the receiving data is shifted onto SPTXD1_I2SD pin. The receiver latches the data on the rising edge of SPCLK1. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SPCLK1 after the LSB has been latched.

The SSPI^{2}S supports programmable data word size from 4 bits to 16 bits. Varying bit rates can be obtained by programming registers SSPCPSR and SSPCR0. Serial data is transmitted in 2s complement with the MSB first. It isn't necessary for the transmitter to know how many bits the receiver can handle, nor does the receiver need to know how many bits are being transmitted.

The following are recommended programming sequence for I^2S mode:

1. Enable the interrupts (if needed).
2. Write to the various fields of the SSPCR0 register.
3. Write to the various bits in SSPCR1 register, while keeping the SSE bit at 0.
4. Write data to the TxFIFO, if transmitting.
5. Enable SSE bit, to start the operation.

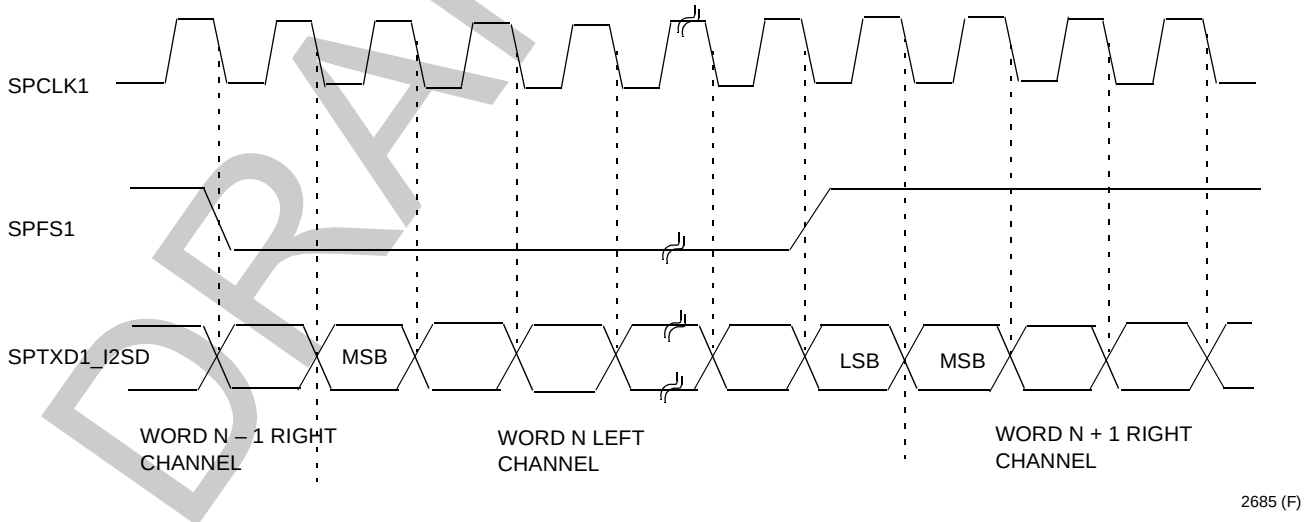


Figure 8.7-9 I^2S Serial Bus Frame Format

8 Digital Signal Processor (DSP) Block (continued)

8.7 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

8.7.6 Registers

The synchronous serial port (SSP) consists of nine registers, shown in Table 8.7-3. In this table, SSP_BASE_ADDR = 0xF3000.

Table 8.7-3 SSP Interface Register Map

Register	Address	Reset Value
Control Register 0 (SSPCR0)	SSP_BASE_ADDR + 0x00	0x0
Control Register 1 (SSPCR1)	SSP_BASE_ADDR + 0x02	0x0
Data Register (SSPDR)	SSP_BASE_ADDR + 0x04	Unknown
Status Register (SSPSR)	SSP_BASE_ADDR + 0x06	0x3
Clock Prescale Register (SSPCPSR)	SSP_BASE_ADDR + 0x08	0x0
Interrupt Mask Set or Clear Register (SSPIMSC)	SSP_BASE_ADDR + 0x0A	0x0
Raw Interrupt Status Register (SSPRIS)	SSP_BASE_ADDR + 0x0C	0x8
Masked Interrupt Status Register (SSPMIS)	SSP_BASE_ADDR + 0x0E	0x0
Interrupt Clear Register (SSPICR)	SSP_BASE_ADDR + 0x10	0x0

8.7.6.1 Control Register 0 (SSPCR0)

SSPCR0 is control register 0 and contains five bit fields that control various functions within the *PrimeCell* SSP. Table 8.15-30 shows the bit assignments for SSPCR0.

8.7.6.2 Control Register 1 (SSPCR1)

SSPCR1 is the control register 1 and contains four different bit fields, which control various functions within the *PrimeCell* SSP. Table 8.15-31 shows the bit assignments for SSPCR1.

8.7.6.3 Data Register (SSPDR)

SSPDR is the data register and is 16 bits wide. When SSPDR is read, the entry in the receive FIFO (pointed to by the current FIFO read pointer) is accessed. As data values are removed by the *PrimeCell* SSP receive logic from the incoming data frame, they are placed into the entry in the receive FIFO (pointed to by the current FIFO write pointer).

When SSPDR is written to, the entry in the transmit FIFO (pointed to by the write pointer) is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. It is loaded into the transmit serial shifter, and then serially shifted out onto SPTXD1_I2SD at the programmed bit rate.

When the data size of less than 16 bits is selected, the user must right justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 bits is automatically right-justified in the receive buffer.

When the *PrimeCell* SSP is programmed for *National MICROWIRE* frame format, the default size for transmit data is eight bits (the most significant byte is ignored). The receive data size is controlled by the programmer. The transmit FIFO and the receive FIFO are not cleared even when SSE is set to zero. This allows the software to fill the transmit FIFO before enabling the *PrimeCell* SSP. Table 8.15-32 shows the bit assignments for SSPDR.

For I^2S, when the system length is greater than the transmitter word length, the word is truncated for data transmission. If the receiver sends more bits than its word length, the bits after the LSB are ignored. If the receiver sends fewer bits than its word length, the missing bits are set to zero internally.

8 Digital Signal Processor (DSP) Block (continued)

8.7 Synchronous Serial Port (SSP) with Inter IC Sound (I^2S) Support (continued)

8.7.6.4 Status Register (SSPSR)

SSPSR is a read-only status register that contains bits that indicate the FIFO fill status and the *PrimeCell* SSP busy status. Table 8.15-37 shows the bit assignments for SSPSR.

8.7.6.5 Clock Prescale Register (SSPCPSR)

SSPCPSR is the clock prescale register and specifies the division factor by which the input SPCLK1 must be internally divided before further use. The value programmed into this register must be an even number between 2 to 254. The least significant bit of the programmed number is hardcoded to zero. If an odd number is written to this register, data read back from this register has the least significant bit as zero. Table 8.15-29 shows the bit assignments for SSPCPSR.

8.7.6.6 Interrupt Mask Set or Clear Register (SSPIMSC)

The SSPIMSC register is the interrupt mask set or clear register. It is a read/write register. On a read, this register gives the current value of the mask on the relevant interrupt. A write of 1 to the particular bit sets the mask, enabling the interrupt to be read. A write of 0 clears the corresponding mask. All the bits are cleared to 0 when reset. Table 8.15-34 shows the bit assignment of the SSPIMSC register.

8.7.6.7 Raw Interrupt Status Register (SSPRIS)

The SSPRIS register is the raw interrupt status register. It is a read-only register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect. Table 8.15-36 shows the bit assignment of the SSPRIS register.

When In I^2S mode, the raw interrupt signals are suppressed in the following way. When in transmit mode, the receive interrupts (RXRIS, RTRIS, and RORIS) are held at 0. When in receive mode the transmit interrupt (TXRIS) is held at 0. If I^2S is in slave receive mode and the master sends in a word of size less than what is programmed in DSS, the RTRIS (receive time-out interrupt) may become active. If there is possibility for a such a situation, then RTIM should be set to 0 masking receive time-out interrupt.

8.7.6.8 Masked Interrupt Status Register (SSPMIS)

The SSPMIS register is the masked interrupt status register. It is a read-only register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect. Table 8.15-35 shows the bit assignment of the SSPMIS register.

8.7.6.9 Interrupt Clear Register (SSPICR)

The SSPICR register is the interrupt clear register and is write-only. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect. Table 8.15-33 shows the bit assignment of the SSPICR register.

8 Digital Signal Processor (DSP) Block (continued)

8.8 Hardware Development System (HDS)

T8307 DSP block contains an on-chip hardware development module for the DSP16000 core (HDS).

HDS is available for debugging assembly-language programs that execute on the DSP core at the core's rated speed. The main capability of the HDS is allowing controlled visibility into the core's state during program execution.

The fundamental steps in debugging an application using the HDS include the following:

1. Setup: Download program code and data into the correct memory regions and set breakpointing conditions.
2. Run: Start execution or single step from a desired starting point (i.e., allow device to run under simulated or real-time conditions).
3. Break: Break program execution on satisfying breakpointing conditions; upload and allow user accessibility to internal state of the device and its pins.
4. Resume: Resume execution (normally or single step) after hitting a breakpoint and finally upload internal state at the end of execution.

A powerful debugging capability of the HDS is the ability to break program execution on complex breakpointing conditions. A complex breakpoint condition, for example, can be an instruction that executes from a particular instruction-address location (or from a particular instruction-address range such as a subroutine) and accesses a coefficient/data element from a specific memory location (or from a memory region such as inside an array or outside an array). Complex conditions can also be chained to form more complex breakpoint conditions. For example, a complex breakpoint condition can be defined as the back-to-back execution of two different subroutines.

The HDS also provides a debugging feature that allows a number of complex breakpoints to be ignored. The number of breakpoints ignored is programmable by the user.

An intelligent trace mechanism for recording discontinuity points during program execution is also available in the HDS. This mechanism allows unambiguous reconstruction of program flow involving discontinuity points such as gotos, calls, returns, and interrupts. The trace mechanism compresses single-level (non-nested) loops and records them as a single discontinuity. This feature prevents single-level loops from filling up the trace buffers. Also, cache loops do not get registered as discontinuities in the trace buffers. Therefore, two-level loops with inner cache loops are registered as a single discontinuity.

The HDS provides a 32-bit cycle counter for accurate code profiling during program development. The cycle counter records processor CLK cycles between user-defined start and end points. The cycle counter can optionally be used to break program execution after a user-specified number of clock cycles.

8 Digital Signal Processor (DSP) Block (continued)

8.9 JTAG Test Port (JTAG)

T8307 DSP block contains an on-chip *IEEE*® 1149.1 compliant JTAG port for the DSP16000 core (JTAG). JTAG is an on-chip hardware module that controls the HDS. All communication between the HDS software, running on the host computer, and the on-chip HDS is in a bit-serial manner through the JTAG port. The JTAG port pins consist of test data input, TDI, test data output, TDO, test mode select, TMS, test clock, TCK, and test reset, TRSTN.

The set of test registers includes the JTAG identification register (ID), the boundary-scan register, and the scannable peripheral registers.

8.9.1 Port Identification

JTAG port has a read-only identification register, **ID**, as defined in Table 8.9-1. As specified in the table, the content of the **ID** register is 0x0C835321 for AA, 0x082B5321 for PB, 0x08335321 for PC and so on.

Note: This register is not memory-mapped. It is not register-mapped either. The user can only access this register through the DSP JTAG port.

Table 8.9-1 ID (JTAG Identification) Register (Only Accessible Through JTAG Port)

Bit	31—30	29—28	27—19	18—12	11—0
Name	RSVD	VERSION ID	ROMCODE	PART ID	AGERE ID

Bit	**Name**	**Value**	**Features**
31—30	RSVD	0x0	Reserved.
29—28	VERSION ID	0x0	Version identification, where the version values are listed below. **Bit [29:28]** / **Version** 00 — T8307. 01 — Reserved. 10 — Reserved. 11 — Reserved.
27—19	ROMCODE	0x190 or 0x105 . . .	User's ROMCODE ID: the ROMCODE ID is the 9-bit binary value of the following expression for the initial version: *400 + (10 x value of the first letter) + (value of the second letter)* For following versions the expression is updated with: *(20 x value of the first letter) + (value of the second letter)*. The values of the letters are shown in the following table. T8307's ROMCODE field contains AA = 0x190, PB=0x105 and so on.
18—12	PART ID	0x35	T8307 part identification.
11—0	AGERE ID	0x321	Agere identification.

ROMCODE Letter	A	B	C	D	E	F	G	H	J	K	L	M	N	P	R	S	T	U	W	Y
Value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

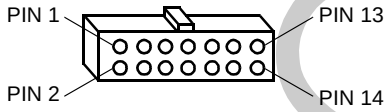
8 Digital Signal Processor (DSP) Block (continued)

8.9 JTAG Test Port (JTAG) (continued)

8.9.2 Emulation Interface Signals (TCS 14-Pin Header)

For in-circuit emulation and application software debugging, the Agere *TargetView*™ Communication System (TCS) provides communication between a host PC and one or more T8307 digital baseband processors.

The TCS interface pod provides a 14-pin, dual-row (0.10 in. x 0.10 in.) socket (female) for connection to the user's target hardware. Figure 8.9-1 illustrates the pinout of this connector. Table 8.9-2 describes the signal names and their relationship to T8307 pins.



5-7333 (F)

Figure 8.9-1 TCS 14-Pin Connector

Table 8.9-2 TCS 14-Pin Socket Pinout

TCS Pin Number	TCS Signal Name	Description	TCS I/O	T8307 Digital Baseband Processor Pin Number	T8307 Digital Baseband Processor Signal Name	T8307 Digital Base-band Processor I/O
1	TCK	Test clock	O	U14	TCK	I
2	NC	No connect	NA	NA	NA	NA
3	Ground	System ground	G	—	VSS	G
4	Ground	System ground	G	—	VSS	G
5	TMS	Test mode select	O	W14	TMS	I
6	VTARG	Target I/O voltage	I	—	VDD_IO_1P8	P
7	NC	No connect	NA	NA	NA	NA
8	NC	No connect	NA	NA	NA	NA
9	TDO	Test data output	I	T12	TDO	O
10	TDI	Test data input	O	V14	TDI	I
11	Ground	System ground	G	—	VSS	G
12	Ground	System ground	G	—	VSS	G
13	NC	No connect	NA	NA	NA	NA
14	NC	No connect	NA	NA	NA	NA

8 Digital Signal Processor (DSP) Block (continued)

8.9 JTAG Test Port (JTAG) (continued)

8.9.3 Test Access Port (JTAG) and Enhanced On-Chip Emulator (EOnCE)

For board-level testing purposes, all of the output and bidirectional pins of T8307 can be 3-stated by issuing the High-Z command to the JTAG port. The JTAG port acts as the *IEEE*-1149.1 compliance TAP port.

In the typical application, the user's board ties T8307 JTAG reset signals, TRSTN, to the device reset, RSTN. Figure 8.9-2 illustrates the connection between the TCS hardware and the T8307 digital baseband processor.

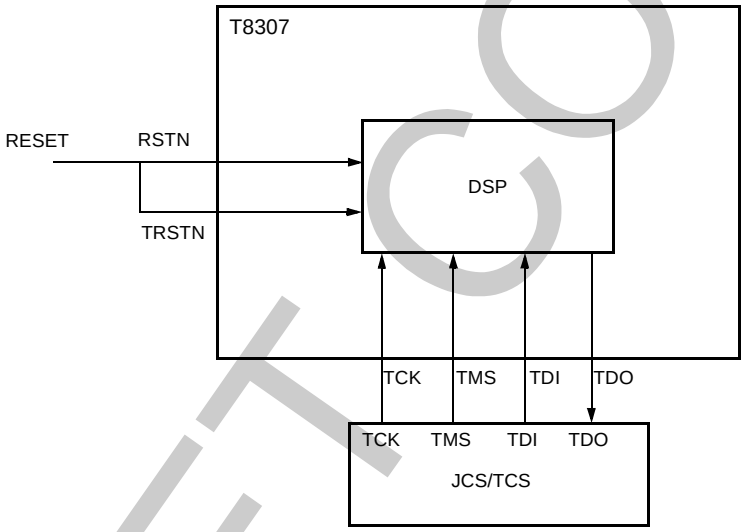


Figure 8.9-2 T8307 JTAG Interface

8.9.4 Boundary-Scan

JTAG contains a full boundary-scan register. The list of boundary-scan cells can be found in the BSD file distributed along with the *LUxWORKS*™ file package.

8 Digital Signal Processor (DSP) Block (continued)

8.9 JTAG Test Port (JTAG) (continued)

8.9.5 DSP JTAG and *ARM* JTAG Daisy Chain

T8307 DSP JTAG and *ARM* JTAG can be daisy chained together. Figure 8.9-3 shows the internal connection of *ARM* and DSP JTAGs. In this figure, when the control signal C is equal to 1 (default mode), *ARM* and DSP JTAGs are completely separate. When C is equal to 0 (daisy chain mode), *ARM* and DSP JTAGs are daisy chained together and connected to the DSP JTAG pins, leaving *ARM* JTAG pins available for their secondary functions.

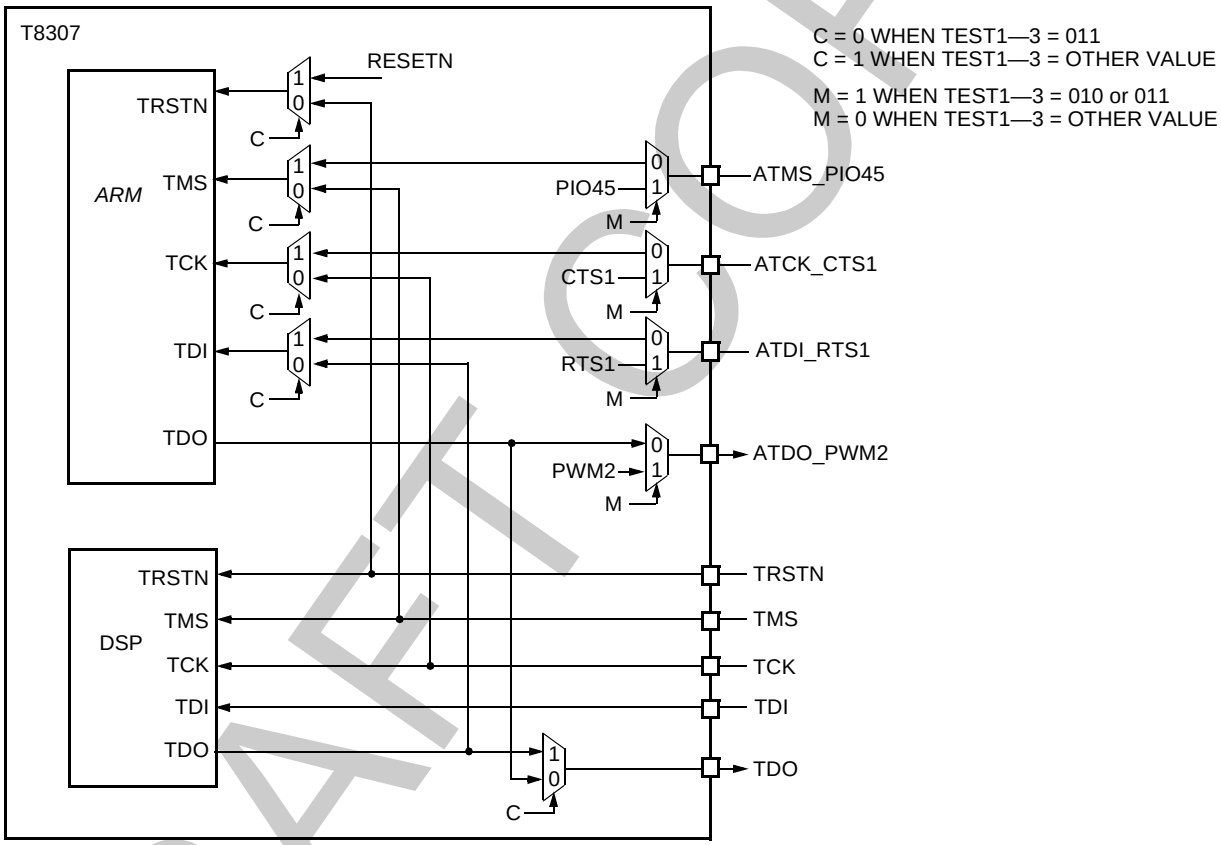


Figure 8.9-3 DSP-JTAG and *ARM*-JTAG Daisy Chain Control

8 Digital Signal Processor (DSP) Block (continued)

8.10 Dual-Port Random-Access Memory (DPRAM)

DSP has a private block of DPRAM consisting of six banks (banks 0—5) of zero wait-state memory. Each bank consists of 4K x 16-bit words and has two separate address and data ports: one port to the core's instruction/coefficient (X-memory) space, and the second port to the core's data (Y-memory) space. DPRAM is organized into even and odd interleaved banks for which each even/odd address pair is a 32-bit wide module, as illustrated in Figure 8.10-1. The core's data buses (XDB and YDB) are each 32 bits wide, and therefore, 32-bit data in the DPRAM with an aligned (even) address can be accessed in a single cycle. Typically, a misaligned double word is accessed in two cycles.

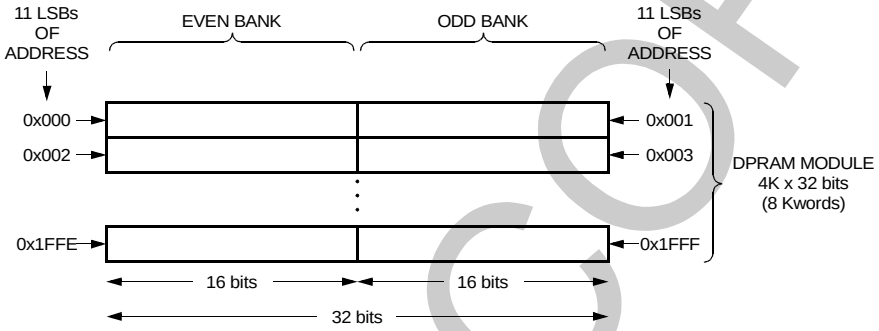


Figure 8.10-1 Interleaved Internal DPRAM

Figure 8.10-2 illustrates an example arrangement of single words (16 bits) and double words (32 bits) in memory. It also illustrates an aligned double word and a misaligned double word. See the *DSP16000 Digital Signal Processor Core* Information Manual for details on word alignment and misalignment wait-states.

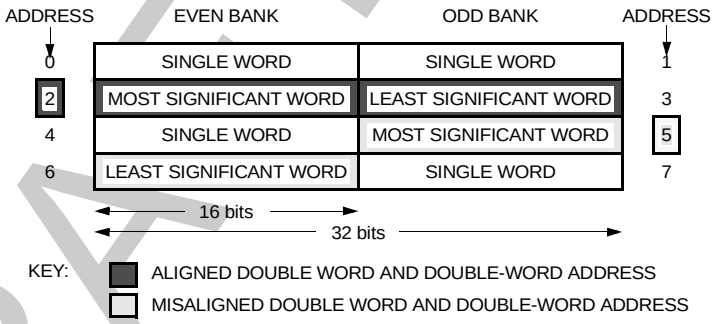


Figure 8.10-2 Example Memory Arrangement

The core's X and Y ports can access separate modules within a DPRAM simultaneously with no wait-states incurred by the core. If the same module of DPRAM is accessed from multiple ports simultaneously, the DPRAM automatically sequences the accesses in the following priority order: X port (instruction/coefficient), then Y port (data). This sequencing can cause the core to incur a conflict wait-state.

8.11 Dual-Port Read-Only Memory (DPROM)

Each bank of DPROM is implemented as 8K x 16 bits ROM and is dual-ported (X and Y). DSP has 144K.

8 Digital Signal Processor (DSP) Block (continued)

8.12 System and External Memory Interface (SEMI)

SEMI is the T8307 DSP block interface to memory-mapped on-chip/off-chip peripherals and memory, including the following:

- The SEMI supports a glueless interface to Agere's CSP8307 conversion signal processor.
- The SEMI supports a maximum total external memory size of 64K (16-bit words).
- The SEMI supports a 16-bit external data bus.
- The SEMI provides programmable enable assertion, setup, and hold times for external asynchronous memory and peripherals.

These features are controlled via SEMI control registers. Some additional features of the SEMI are the following:

- The SEMI arbitrates and prioritizes accesses from the core.
- The SEMI controls the internal system bus, which allows the core to access the shared internal I/O memory component.

Figure 8.12-1 depicts the internal and external interfaces to the SEMI. The SEMI interfaces directly to the X-memory space buses and Y-memory space buses. This allows the following:

- Core to perform external program or data accesses.
- Core to access the internal system bus.

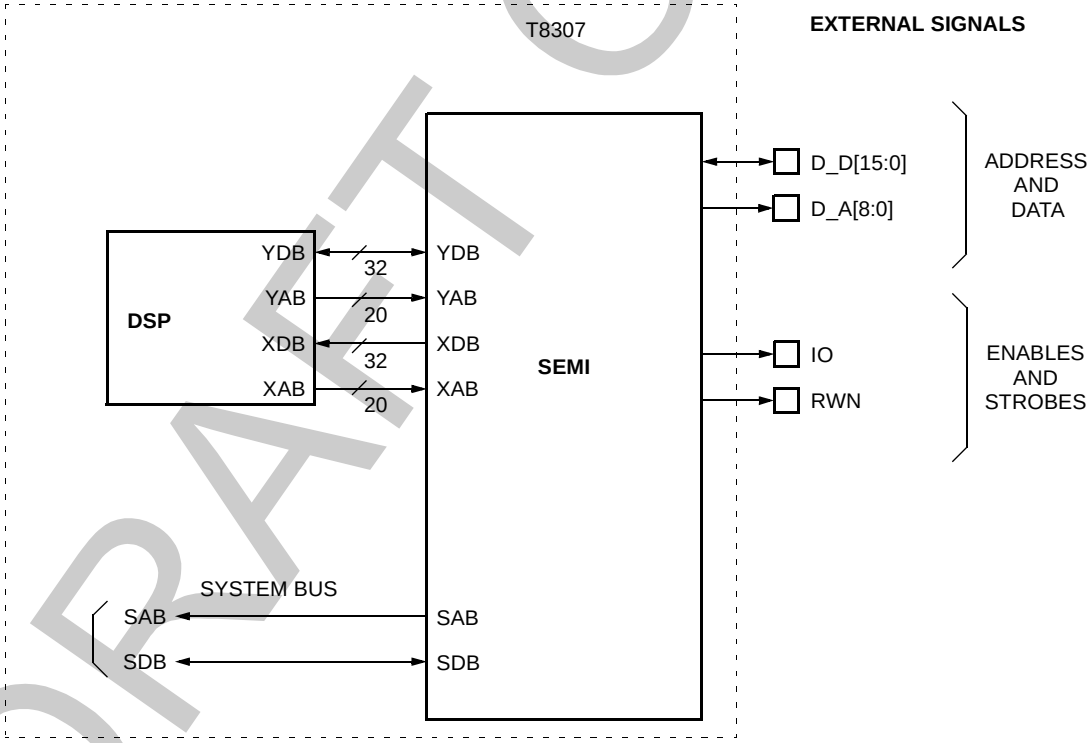


Figure 8.12-1 SEMI Interface Block Diagram

8 Digital Signal Processor (DSP) Block (continued)

8.12 System and External Memory Interface (SEMI) (continued)

8.12.1 External Interface

Table 8.12-1 provides an overview of the SEMI pins. These pins are described in detail in the remainder of this section.

Table 8.12-1 Overview of SEMI Pins

Function	Pin	Type	Description
Enables and Strobes	IO	O	EIO component enable (negative assertion).
	RWN	O	External read/write not.
Address and Data	D_D[15:0]	I/O/Z	Bidirectional 16-bit external data bus.
	D_A[8:0]	O	External address bus.

8.12.1.1 Enables and Strobes

The SEMI provides a negative-assertion external memory enable output pin for the external memory component EIO. The EIO pin is the active-low enable for the external memory component EIO (external I/O). Refer to the memory maps described in Section 6.4 and shown in Figure 6.4-1—Figure 6.4-2 for details about these memory components. The SEMI provides a negative-assertion write strobe output pins, RWN. Table 8.12-2 details the SEMI enables and strobe pins.

Table 8.12-2 Enable and Strobe Pins for the SEMI External Interface

Pin	Value	Description
IO (negative-assertion output)	0	The SEMI is selecting the EIO memory component for an access. The SEMI asserts I/O for the number of instruction cycles specified by the IATIME[3:0] field (**ECON0**[11:8]; see Table 8.15-28).
	1	The SEMI is not selecting the EIO memory component for an access.
RWN (negative-assertion output)	0	The SEMI is performing an external write access over the external data bus (D_D[15:0]).
	1	The SEMI is not performing an external write access over the external data bus (D_D[15:0]).

8 Digital Signal Processor (DSP) Block (continued)

8.12 System and External Memory Interface (SEMI) (continued)

8.12.1.2 Address and Data

The SEMI provides a 16-bit external data bus, D_D[15:0], and a 16-bit external address bus, D_A[8:0], to select a location within the selected external memory component (EIO).

8.12.2 16-Bit External Bus Accesses

Each access by the core can be a 16-bit (single-word) or 32-bit (double-word) access. The SEMI will perform two 16-bit external operations for a 32-bit (double-word) access to the external memory space.

8.12.3 Registers

There is a 16-bit memory-mapped control register that configures the operation of the SEMI, as shown in Table 8.12-3.

Table 8.12-3 SEMI Memory-Mapped Registers

Register Name	Address	Description	Size (Bits)	R/W	Type	Reset Value
ECON0	0xF0000	SEMI Control.	16	R/W	Control	0x0FFF
Reserved	0xF0002—0xF000A	Reserved. Do not write.	32	—	—	0xxxxx
Reserved	0xF000C—0xF000F	Reserved.	32	—	—	0xxxxx

8.12.3.1 ECON0 Register

ECON0 determines the setup, hold, and assertion times for the external memory component enable (EIO).

8 Digital Signal Processor (DSP) Block (continued)

8.12 System and External Memory Interface (SEMI) (continued)

8.12.4 Asynchronous Memory

This section describes the functional timing and interfacing for external memory components. In this section, the following are assumed:

- The designation *ENABLE* refers to the I/O pin.
- The designation *RWN* refers to the RWN pin.
- The designation *D_A* refers to the external address pins D_A[8:0].
- The designation *D_D* refers to the external data pins D_D[15:0].
- The designation *ATIME* refers to IATIME (**ECON0**[11:8]) for accesses to the EIO space.
- RSETUP refers to the RSETUP field (**ECON0**[12]; see Table 8.15-28).
- RHOLD refers to the RHOLD field (**ECON0**[14]).
- WSETUP refers to the WSETUP field (**ECON0**[13]).
- WHOLD refers to the WHOLD field (**ECON0**[15]).

8.12.4.1 Functional Timing

The following describes the functional timing for an asynchronous read operation:

1. On a rising edge of the internal clock (CLK), the SEMI asserts *ENABLE* and drives the read address onto *D_A*. If RSETUP is set, the SEMI asserts *ENABLE* one CLK cycle later.
2. The SEMI asserts *ENABLE* for *ATIME* CLK cycles.
3. The SEMI deasserts *ENABLE* on a rising edge of CLK and latches the data from *D_D*.
4. The SEMI continues to drive the read address onto *D_A* for a minimum of one CLK cycle to guarantee an address hold time of at least one cycle. If RHOLD is set, the SEMI continues to drive the read address for an additional CLK cycle.

The SEMI continues to drive the address until another external memory access is initiated. Another read or a write to the same memory component can immediately follow the read cycle described previously.

The following describes the functional timing for an asynchronous write operation:

1. On a rising edge of the internal clock (CLK), the SEMI asserts *RWN* and drives the write address onto *D_A*. If WSETUP is set, the SEMI asserts *RWN* one CLK cycle later.
2. One CLK cycle after the SEMI asserts *RWN*, the SEMI asserts *ENABLE* and drives valid data onto *D_D* to guarantee one CLK cycle of setup time.
3. The SEMI asserts *ENABLE* for *ATIME* CLK cycles.
4. The SEMI deasserts *ENABLE* on a rising edge of CLK.
5. The SEMI continues to drive *D_D* with the write data, drive *D_A* with the write address, and assert *RWN* for one additional CLK cycle to guarantee one cycle of hold time. If WHOLD is set, the SEMI continues to drive the write address for an additional CLK cycle.

The SEMI continues to drive the address until another external memory access is initiated. Another write to the same memory component can immediately follow the write cycle described previously. If a read to the same memory component follows the write cycle described previously, the SEMI inserts an idle bus cycle (one CLK cycle).

8 Digital Signal Processor (DSP) Block (continued)

8.12 System and External Memory Interface (SEMI) (continued)

8.12.4.2 Interfacing Examples

Figure 8.12-2 illustrates an example of interfacing CSP8307 to the SEMI. The programmer can configure the access time (defined as the number of CLK cycles that the enable is asserted) for the enable. The IATIME field (**ECON0**[11:8]) specifies the number of CLK cycles that the I/O enable is asserted. The range of values for these fields is from 0 to 15 (corresponding to a range of 1 to 15 CLK cycles). A value of 0 or 1 programs a 1 CLK assertion time.

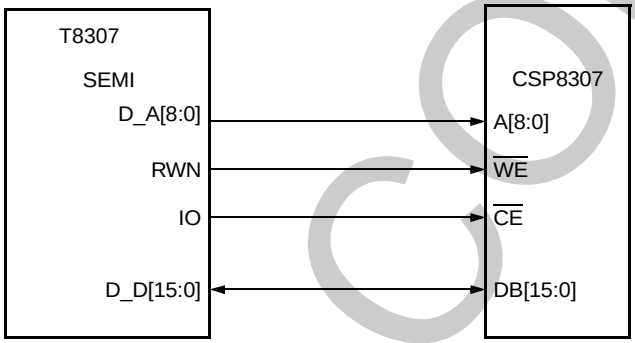


Figure 8.12-2 16-Bit External Interface with CSP8307

8.12.5 System Bus Peripherals

The SEMI system bus enables integration of on-chip DSP memory-mapped peripherals. The bus is divided into sixteen 4K segments, enabling the attachment of 16 peripherals (peripherals 15—0). The first group of eight peripherals (peripherals 7—0) have zero read and write access delays. The second group of eight peripherals (peripherals 15—8) have programmable read and write access delays.

8 Digital Signal Processor (DSP) Block (continued)

8.12 System and External Memory Interface (SEMI) (continued)

8.12.6 Performance

The following terms are used in this section:

- The core requests the SEMI to access external memory or the system bus.
- Contention refers to multiple requests for the same resource at the same time.
- The designation *ATIME* refers to IATIME (**ECON0**[11:8]; see Table 8.15-28) for accesses to the EIO space.
- RSETUP refers to the RSETUP field (**ECON0**[12]).
- RHOLD refers to the RHOLD field (**ECON0**[14]).
- WSETUP refers to the WSETUP field (**ECON0**[13]).
- WHOLD refers to the WHOLD field (**ECON0**[15]).
- Misaligned refers to 32-bit accesses to odd addresses.
- T_{CLK} refers to one period of the internal clock CLK.

The SEMI controls and arbitrates two types of memory accesses. The first is to external memory. The second is to the internal I/O segment accessed via the system bus. Section 8.12.6.1 describes the SEMI performance for system bus accesses. Section 8.12.6.2 describes the SEMI performance for asynchronous external memory accesses.

For the remainder of this section, unless otherwise stated, the following assumptions apply:

- There is only a single requester (i.e., no contention).

The type of access (read vs. write) determine the throughput of any external memory access. Section 8.12.6.2 describes the performance for all combinations.

8.12.6.1 System Bus

The SEMI controls and arbitrates accesses to internal I/O segment accessed via the system bus. The system bus is used to access all the memory-mapped registers in SEMI. See Section 8.15.2 for details on the memory-mapped registers.

Table 8.12-4 specifies the minimum system bus access time for either a single-word (16-bit) or double-word (32-bit) access by a single requester. The SEMI processes system bus accesses by multiple requesters at a maximum rate of one access per CLK cycle.

Table 8.12-4 System Bus Minimum Access Times

Access	Minimum Access Time
Read	$5 \cdot T_{CLK}$
Write	$2 \cdot T_{CLK}$

8 Digital Signal Processor (DSP) Block (continued)

8.12 System and External Memory Interface (SEMI) (continued)

8.12.6.2 External Memory, Asynchronous Interface

External Accesses by Either Core, 16-Bit SEMI Data Bus

The following describes the SEMI performance for read and write operations by the core to asynchronous memory with the external data bus:

READS. For the core, 16-bit external asynchronous memory reads occur with a minimum period of the enable assertion time (as programmed in *ATIME*), plus one CLK cycle enforced hold time, plus three CLK cycles for the SEMI pipeline to complete the core access. This assumes that RSETUP and RHOLD are cleared. The SEMI coordinates two separate accesses for aligned 32-bit reads, adding two CLK cycles to the previous description. The core treats misaligned 32-bit reads as two separate 16-bit reads requiring two complete SEMI accesses.

The core read access time for a 16-bit data bus is the following:

$$[ATIME + aligned + \text{RSETUP} + \text{RHOLD}] \cdot misaligned \cdot T_{CLK}$$

where:

- *aligned* = 4 and *misaligned* = 1 for 16-bit accesses.
- *aligned* = 6 and *misaligned* = 1 for 32-bit aligned accesses.
- *aligned* = 4 and *misaligned* = 2 for 32-bit misaligned accesses.

WRITES. For the core, 16-bit asynchronous memory writes can occur with a minimum period of the enable assertion time (as programmed in *ATIME*), plus one CLK cycle enforced setup time, plus one CLK cycle enforced hold time. This assumes that WSETUP and WHOLD are cleared. Unlike read cycles, the core does not wait for the SEMI pipeline to complete the access, so the three CLK cycle pipeline delay is not incurred on core writes. The SEMI coordinates and treats aligned 32-bit writes as two separate accesses. The core treats misaligned 32-bit writes as two separate 16-bit writes requiring two complete SEMI accesses.

The core write access time for a 16-bit data bus is the following:

$$[ATIME + 2 + \text{WSETUP} + \text{WHOLD}] \cdot longword \cdot T_{CLK}$$

where:

- *longword* = 1 for 16-bit accesses.
- *longword* = 2 for 32-bit accesses.

8 Digital Signal Processor (DSP) Block (continued)

8.12 System and External Memory Interface (SEMI) (continued)

8.12.6.3 Summary of Access Times

Table 8.12-5 summarizes the information in Section 8.12.6.2.

Table 8.12-5 Access Time Per SEMI Transaction, Asynchronous Interface

Requester	Access Type	Reads	Writes
Core	16-bit	[*ATIME* + 4 + RSETUP + RHOLD] · T_{CLK}	[*ATIME* + 2 + WSETUP + WHOLD] · T_{CLK}
	32-bit aligned	[*ATIME* + 6 + RSETUP + RHOLD] · T_{CLK}	[*ATIME* + 2 + WSETUP + WHOLD] · 2 · T_{CLK}
	32-bit misaligned	[*ATIME* + 4 + RSETUP + RHOLD] · 2 · T_{CLK}	[*ATIME* + 2 + WSETUP + WHOLD] · 2 · T_{CLK}

Table 8.12-6 shows example access times under various conditions. These access times are derived from actual measurements. For the asynchronous access times, it is assumed that the programmed enable assertion time is one (*ATIME* = 1) and that RSETUP = RHOLD = WSETUP = WHOLD = 0. The actual value of these fields is application-dependent.

Table 8.12-6 Example Average Access Time Per SEMI Transaction, 16-Bit Data Bus

Requester	Access Type	Reads	Writes
Core	16-bit	5 · T_{CLK}	3 · T_{CLK}
	32-bit aligned	7 · T_{CLK}	6 · T_{CLK}
	32-bit misaligned	10 · T_{CLK}	6 · T_{CLK}

8.12.7 Priority

SEMI prioritizes the requests from the core in the following order:

1. DSP program (X) and data (Y) requests have the highest priority. If DSP requires a simultaneous X and Y access, X is performed first, then Y.

8 Digital Signal Processor (DSP) Block (continued)

8.13 Clock Synthesis

The DSP master clock source is selected from one of the following sources:

- CKI: This is derived from an external clock through the small-signal buffer.
- PLL: The PLL generates a clock source with a programmable frequency that is a multiple of the CKI clock. Note that the DSP and *ARM* share a common PLL, with most of the frequency control residing on the *ARM* side.
- Crystal oscillator (OSC): The DSP can switch over to the crystal oscillator clock to save power while waiting for any event (such as an interrupt) to happen.
- TCK: The JTAG port clock can be selected as the DSP system clock via the JTAG mode (jmode) register. Software may select either a synchronous or asynchronous switchover to and from TCK. TCK is the only clock source for which an asynchronous switchover can be performed.

WARNING: An asynchronous switchover to or from TCK is not guaranteed to be glitch-free.

8.13.1 Clock Switch Module

The clock switch module controls selection of the DSP clock via the PLL control (**pllc**) and power control (**powerc**) registers. In normal operation, the DSP clock is selected synchronously through a combination of the PLLSEL bit in **pllc** and the SLOWCK bit in **powerc**. As mentioned previously, the JTAG test port clock, TCK, may be selected either synchronously or asynchronously via the **jmode** register in the JTAG module. Status regarding the currently active clock source and the lock status for the PLL are accessible from the PLL status register **pllsac**.

Note: After reset, CKI is used as the clock source for the DSP system clock.

8.13.2 Phase-Locked Loop (PLL)

T8307 contains two PLL's, but the operation of these PLLs differs from previous Trident devices. One PLL (UPLL) is dedicated to the USB. The other PLL (ADPLL) provides the high speed master clocks for both the *ARM* and DSP. When the ADPLL is being used, both DSP and *ARM* clocks are generated from the same VCO. Note that separate postdividers are provided. Thus, the *ARM* and DSP PLL clocks can be set to different frequencies over a limited range. Most of the PLL settings are controlled solely from the *ARM* side. The DSP has control only of the bits for setting its postdivide value.

The input to the phase-locked loop (PLL) comes from the small-signal clock buffer CKI. The PLL block is illustrated in Figure 8.13-1. The PLL requires an external input clock for its operation. Setting the appropriate bits in the **pllc** register (see Table 8.15-18) enables the PLL to become the clock source. Selection of the PLL overrides selection of the crystal oscillator (i.e., setting SLOWCK in the **powerc** register).

Note: Selecting TCK from the JTAG JMODE register overrides all other clock sources.

8 Digital Signal Processor (DSP) Block (continued)

8.13 Clock Synthesis (continued)

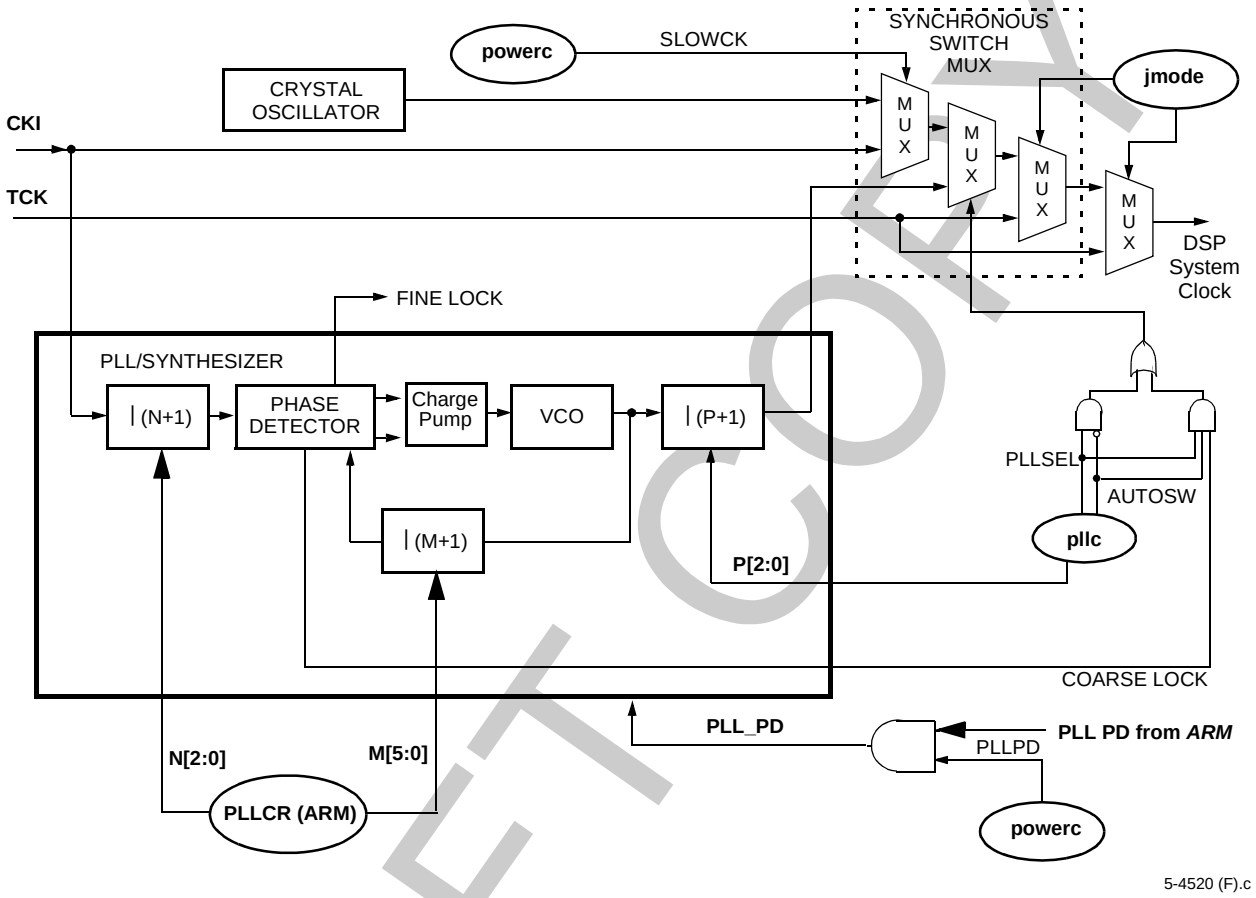


Figure 8.13-1 PLL Block

Setting the PLLSEL bit in the **pllc** register switches source from either CKI or OSC to the PLL without glitching. It is important to note that the P setting of the **pllc** register must be maintained when setting the PLLSEL bit. Although changing the P setting will not cause the PLL to lose lock, a few clock cycles are necessary for the postdivider itself to reinitialize and settle to the correct frequency. When selecting the PLL with a new value of P, the new P value should first be programmed with DIV_RSTN low and PLLSEL low to reset the P divider. When PLLSEL is brought high to select the PLL, the DIV_RSTN bit should be brought high along with PLLSEL. Clearing the PLLSEL bit deselects the PLL so that the DSP is clocked by either CKI or OSC as determined by the SLOWCK bit in powerc. Six (6) nop instructions should follow any instruction that changes the state of PLLSEL.

8 Digital Signal Processor (DSP) Block (continued)

8.13 Clock Synthesis (continued)

The AUTOSW bit is used in conjunction with PLLSEL to allow the DSP system clock to automatically switch over to the PLL after the PLL has obtained lock. Setting AUTOSW and PLLSEL will ensure that the PLL has locked to the programmed frequency before it can drive the DSP. If the PLL is not locked when AUTOSW and PLLSEL are set, the currently active system clock will continue to operate until the PLL locks. If the PLL is locked when these bits are set, the system clock will be switched to the PLL. In the latter case, the switch occurs after the synchronization delay across the clock switch circuit. When the automatic PLL switchover mode is desired, AUTOSW must be set when setting PLLSEL. AUTOSW is set by default on DSP system reset.

The frequency of the PLL output clock (f_{PLL}) is determined by the values loaded into the 3-bit N and P dividers and the 6-bit M multiplier. Note that M and N values are controlled by the *ARM*. When the PLL is selected and locked, the frequencies of the PLL and the DSP system clocks are related to the frequency of CKI (f_{CKI}) by the following equations:

$$f_{DCLK} = f_{PLL} = f_{CKI} \times \frac{M + 1}{(N + 1) \times (P + 1)}$$

The general procedure for changing the PLL output frequency is as follows:

1. Deselect the PLL by clearing PLLSEL in **pllc**.
2. Wait for DSP system clock to switch over to CKI or OSC.
3. Change P field in **pllc** and clear DIV_RSTN. (M and N values have already been set by the *ARM*.)
4. Set PLLSEL, DIV_RSTN and, optionally, AUTOSW in **pllc**.

8 Digital Signal Processor (DSP) Block (continued)

8.13 Clock Synthesis (continued)

The lock status of the PLL is available in the FINLCK and CRSLCK bits of **pllsac**. Each bit reads the lock detect signals output from the PLL. When FINLCK is set, the PLL has determined itself to be locked within 0.1% of the target frequency programmed in the **pllc** register. CRSLCK is set when the PLL is locked within 1.5% of the target. Both FINCLK and CRSLCK are cleared on system reset.

Note: CRSLCK controls the automatic switchover of the PLL to the system clock when AUTOSW is set in the **pllc** register.

ACTVCLK displays the currently selected system clock source. Upon system reset, CKI is selected as the clock source, which is reflected in the value of this field.

PLL Programming Example. The following section of code illustrates how the PLL would be initialized on powerup of the DSP, assuming the following operating conditions:

- CKI input reference frequency (f_{REF}) 13 MHz
- DSP system clock frequency (f_{DCLK}) 91 MHz (target) 91 MHz (actual)
- PLL predivider value N + 1 1 (Must have been previously set by *ARM*.)
- PLL multiplier value M + 1 42 (Must have been previously set by *ARM*.)
- PLL Postdivider value P + 1 6 (P[2:0] = 101)

The device comes out of reset with the PLL disabled and deselected. Before the DSP can operate from the PLL, the *ARM* software must enable the PLL and program the VCO frequency by setting the M and N values.

```
pllinit:    pllc   = 0x0A00   /* Set P = 0x5 */
            6*nop
            powerc = 0x0000   /* Make sure PLL stays on even if ARM turns it off */
                              /* If ARM did turn it off, ARM should still maintain */
                              /* valid M and N values in ARM PLLCR register */
            pllc   = 0xDA00   /* Set PLLSEL, DIV_RSTN, AUTOSW, P = 0x5 */
            6*nop
            goto start        /* Branch to user's code. */
                              /* DSP will switch to 91 MHz after PLL locks. */
```

Frequency Accuracy and Jitter. Table 8.13-1 summarizes the specifications for the PLL.

Table 8.13-1 PLL Specifications

Parameter	Min	Max	Unit
Phase Detector Input Frequency	10	100	MHz
VCO Frequency	300	1000*	MHz
Multiplier (M+1) (Set by *ARM*)	4†	64†	—
Reference Input Predivider (N+1) (Set by *ARM*)	1†	8†	—
VCO Output Postdivider (P+1)	1†	8†	—
Output Duty Cycle (P + 1 = 2, 4, 6, 8)	48	52	%
Output Duty Cycle (P + 1 = 3, 5, 7)	45	55	%
Output Duty Cycle (P + 1 = 1)	40	60	%
Peak-to-peak Jitter at VCO (t_{jit})	–150	150	ps
Lock Time, Trim Enabled	—	50	∝s
Lock Time, Trim Disabled	—	2	ms

* Maximum VCO frequency limited to 728 MHz for P values of 2, 4, or 6.
† These are the absolute possible ranges. The actual range for each depends upon the input reference clock frequency used.

8 Digital Signal Processor (DSP) Block (continued)

8.13 Clock Synthesis (continued)

Frequency Accuracy and Jitter. When using the PLL to multiply the input clock frequency up to the instruction clock rate, it is important to realize that although the average frequency of the internal DSP system clock and the CKO pin has almost the same relative accuracy as the input clock, noise sources within T8307 produce jitter on the PLL clock; therefore, each individual clock period has some error associated with it.

PLL V_{DD} and V_{SS} Connections. Since the PLL contains analog circuitry, the PLL V_{DD} will tend to be more sensitive to supply noise than other V_{DD} pins. Dedicated decoupling capacitors should be provided for the PLL V_{DD} pin, and a series ferrite bead or resistor may also be needed depending on the characteristics of the supply noise. The PLL V_{SS} pin can be connected directly to the main ground plane.

8.13.3 Reset

The term **reset** refers to the establishment of a defined initial state for the DSP device. There are four main sources for DSP reset during normal device operation. The sources include the following:

- External pin reset (RESETN).
- Reset from ICP (DRESETN bit of DCCON).
- JTAG instruction register (JIR) reset command.
- JTAG mode register (JMODE) control field.

The JTAG module reset is reserved for use by the DSP software tools.

Assertion of the external RESETN pin reset immediately resets the DSP system; however, deassertion of RESETN is delayed internally by the on-chip DSP reset circuit to allow time for the reset initialization sequence to complete. After reset the core executes boot instructions from the dual-port memory in ICP.

8.13.4 External Clock (CKO) Selection

The CKO output is available on the CKO_IACK pin. This signal is programmed using the CKOSEL field (bits 5 through 7) of the **ioc** register (see Table 8.15-13) to select one of the following outputs:

- DCLK: Free running DSP system clock.
- CKI: Output of CKI clock buffer.
- IACK: Interrupt acknowledge.
- ONE: Held high.

8 Digital Signal Processor (DSP) Block (continued)

8.14 Power Management

There are four different controls available for placing the DSP16000 into low-power modes: the SSOFFD and PLLPD fields in the **powerc** register and the AWAIT bit in the **alf** register. The fourth mode is selection of the 32 kHz clock source using the SLOWCK field in the **powerc** register.

8.14.1 powerc Control Register Bits

The **powerc** register (see Table 8.15-20) controls the powering down of various portions of the chip and the selection of some clock modes.

PLLPD: Setting this bit powers down the PLL only if the *ARM* has also requested a power down of the PLL. This bit is hardware interlocked, preventing power-down of the PLL when the PLL is the active DSP system clock source. Writes to this bit will not take effect until the PLL is no longer the system clock, and reading this bit will return zero while the PLL is the active source. The PLL must first be deselected by clearing the PLLSEL bit in **pllc** before PLLPD is written. Six **nop** instructions should follow the clearing of PLLSEL before any change is made to PLLPD. Upon clearing PLLPD, the PLL will power up and will output the frequency determined by the P field in **pllc**.

SSOFFD: Setting the SSOFFD bit disables the small-signal buffer. This reduces the power consumed by the small-signal buffer. Note that if the CP needs the CKI clock, the small-signal buffer will not be disabled, even if SSOFFD = 1. After re-enabling the small-signal clock buffer, the DSP has to wait for a period equal to the start-up time of the small-signal buffer before switching over to CKI. Like PLLPD, SSOFFD is hardware interlocked to prevent disabling the small-signal buffer when either CKI or the PLL is selected as the DSP system clock. Writes to this bit will not take effect while either clock is the active source, and reading this bit will return zero. Note that the only clock sources available to the DSP when the small-signal buffer is disabled are the crystal oscillator (OSC) and the JTAG clock (TCK).

SEMIDIS: This bit controls the clock to the SEMI. Setting this bit disables the SEMI clock, while clearing this bit enables the clock. Two (2) **nop** instructions should follow any instruction that changes the state of SEMIDIS. This bit is cleared on system reset (clock enabled).

SLOWCK: Setting the SLOWCK bit enables the crystal oscillator (OSC) to be selected as the clock source for the DSP system clock. Selecting the PLL (PLLSEL in the **pllc** register) takes precedence over SLOWCK. Switching of the clocks is synchronized so that no partial or short clock pulses occur. Six (6) **nop** instructions should follow any instruction that changes the state of SLOWCK.

NOCK: Setting the NOCK bit synchronously turns off the DSP system clock, regardless of the active source, and halts program execution. Six (6) **nop** instructions should follow any instruction that sets NOCK. The recommended sequence of setting the NOCK bit is as follows:

```
powerc = 0x00C0
nop
powerc = 0x02C0
6 * nop
```

Writing 0x0040 followed by 0x0240 or 0x0080 followed by 0x0280 are supported also.

The NOCK bit can be cleared by asserting the INT0 or ICPINT signals (if the INT0EN or ICPEN bit is set, respectively), allowing the halted program to resume execution from where it halted without any loss of state. If INT0EN or ICPEN is set, it is recommended that the programmer disable the corresponding interrupt in **inc0** before setting NOCK. This will avoid an unintentional interrupt due to the subsequent assertion of the corresponding interrupt signal. After the halted program resumes, it should clear the corresponding interrupt by writing to the **ins** register. Resetting the device by asserting the RESETN pin also clears the NOCK bit, but the halted program cannot resume execution.

INT0EN: This bit allows the INT0 pin to asynchronously clear the NOCK bit, thereby allowing the device to continue program execution from where it halted without loss of state. No chip reset is required. It is recommended that when INT0EN is to be used, the INT0 interrupt be disabled in the **inc0** register to prevent an unintended interrupt. After the program execution resumes, the INT0 interrupt should be cleared in the **ins** register.

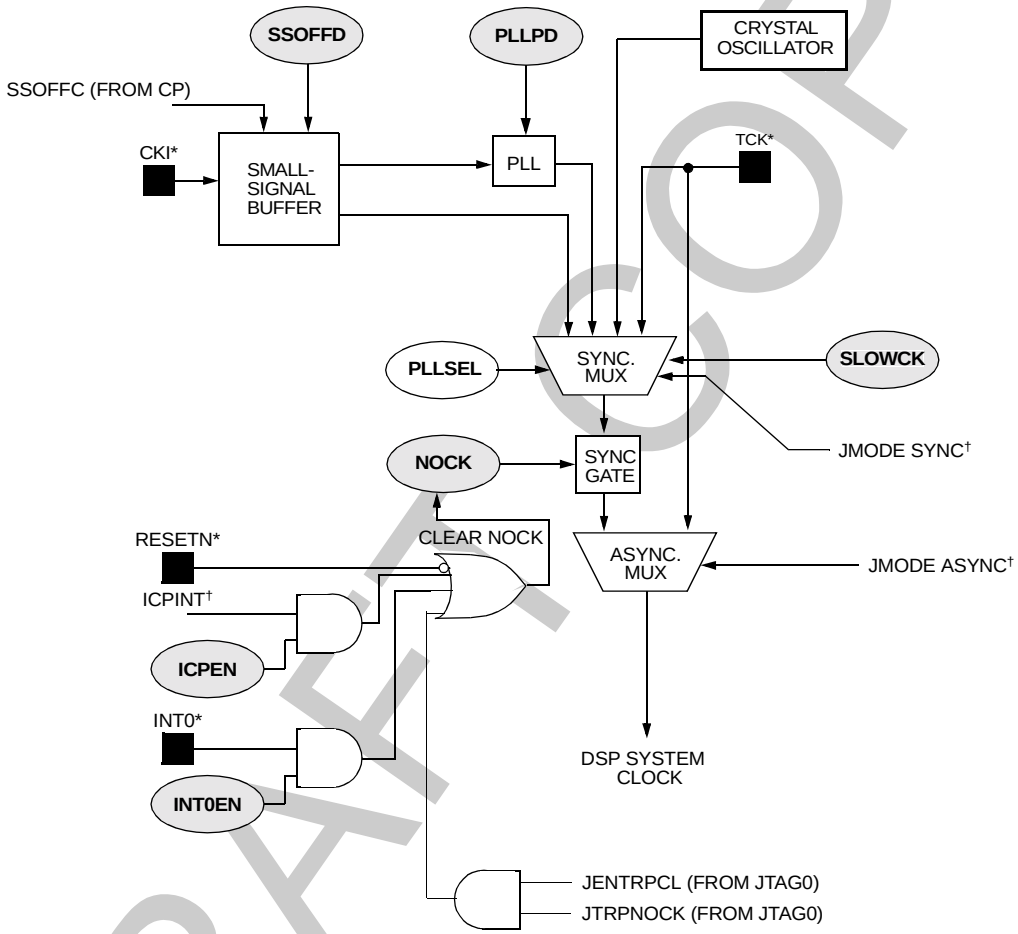
ICPEN: This bit allows the CP to awaken the DSP via the ICP interrupt to the DSP core. The DSP will continue program execution from where it halted without loss of state. No chip reset is required.

SSPDIS: This bit controls the clock to the SSP/I^2S. Setting this bit disables the SSP/I^2S clock, while clearing this bit enables the clock. Two (2) **nop** instructions should follow any instruction that changes the state of SSPDIS. This bit is cleared on system reset (clock enabled).

8 Digital Signal Processor (DSP) Block (continued)

8.14 Power Management (continued)

TMRDIS: This bit controls the clock to the timer for DSP. Setting the bit to one disables the timer, reducing power consumption. Clearing the bit enables the timer. The function of TMRDIS is identical to that of the DISABLE field of the **timerc** register. Two (2) **nop** instructions should follow any instruction that changes the state of the TMRDIS field. These bits are cleared on system reset (clock enabled).



* Signal from external pin.

† Internal signals.

Notes:
The functions in the shaded ovals represent bits in the **powerc** register. The functions in the nonshaded ovals represent bits in the **pllc** register.

Deep sleep is the state entered either by a software stop of the internal processor clock.

The switching of the synchronous multiplexer (sync. MUX) and the synchronous gate is designed so that no glitching occurs during switch. Switching via the asynchronous multiplexer (async. MUX) is not guaranteed glitch-free.

PLL is selected by the PLLSEL bit of **pllc**; PLL powerdown is controlled by the PLLPD bit of **powerc**.

Figure 8.14-1 Power Management Using the powerc and the pllc Registers

8 Digital Signal Processor (DSP) Block (continued)

8.14 Power Management (continued)

8.14.2 Low-Power Standby Mode, AWAIT Bit of the alf Register

Setting the AWAIT bit of the **alf** register causes the processor to go into the standard sleep state or power-saving standby mode. In this mode, the minimum circuitry required to process an incoming interrupt remains active, and the PLL remains active if enabled. An interrupt returns the processor to the previous state, and program execution continues. The action resulting from setting the AWAIT bit and the action resulting from setting bits in the **powerc** register are mostly independent. As long as the processor is receiving a clock, whether slow or fast, the DSP is put into standard sleep mode with the AWAIT bit. Once the AWAIT bit is set, the STOP pin may be used to stop and later restart the processor clock, returning to the standard sleep state. If the processor clock is not running, however, the AWAIT bit is not set.

To properly program this node, two **nop** instructions are programmed after the AWAIT bit is set. The first **nop** (one cycle) is executed before sleeping; the second is executed after the interrupt signal awakens the DSP and before the interrupt service routine is executed.

The DSP can be awakened by an interrupt from the timer, providing the interrupt is enabled in the **inc0** register.

8 Digital Signal Processor (DSP) Block (continued)

8.14 Power Management (continued)

8.14.3 Power Management Sequencing

There are important considerations for sequencing the power management modes. The following examples illustrate proper methods.

8.14.4 Power Management Examples Without the PLL

The following examples illustrate significant options for reducing the power dissipation without the PLL. These are valid only if the PLL is deselected (PLLSEL = 0 in **pllc**) and is powered down (PLLPD = 1 in **powerc**). The small-signal clock buffer also consumes power. It is possible to disable the clock buffer by setting SSOFFD in **powerc** to save power. Re-enabling the clock buffer incurs start-up latency. The start-up latency could be around tens of microseconds. Therefore, any powerdown mode that shuts down the small-signal buffer should incorporate a software wait loop for a period greater than the start-up time of the clock buffer.

Low-Power Standby Mode. This mode uses the await bit in the **alf** register to freeze the clocks in all of the blocks of the DSP16000 core, except for the interrupt circuits. The DSP peripherals continue to be clocked even after the **alf** register's AWAIT bit is set. The clock to these units can be turned off to further reduce the standby power.

```
      powerc = 0x8000   /* Turn off PLL */
      2*nop             /* Wait for it to take effect */
      ioc = 0x01E0      /* Hold CKO high */
standby:alf = 0x8000    /* Set AWAIT bit, stop internal processor clock, */
      nop               /* interrupt circuits active */
      nop               /* Needed for bedtime execution. Only standby power */
      nop               /* consumed here. Interrupt wakes up the device */
cont: ...               /* User code executes here */
      2*nop             /* Wait for it to take effect */
      ioc = 0x0000      /* CKO is free-running */
```

Standby with Slow Internal Clock. In this case, the crystal oscillator is selected to clock the DSP section before the device is put into standby mode via AWAIT in **alf**. This reduces the power dissipation while waiting for an interrupt to continue program execution. The PLL is not selected and is disabled (powered off).

Note: The small-signal clock buffer is not shut off and continues to run; therefore, there is no latency.

```
      powerc = 0x8400   /* Select oscillator as slow clock */
      6*nop             /* Wait for oscillator selection to take effect */
      ioc = 0x01E0      /* Hold CKO high */
standby:alf = 0x8000    /* Set AWAIT bit, stop internal processor clock */
      nop               /* interrupt circuits active */
      nop               /* Needed for bedtime execution.*/
                        /* Reduce standby power */
      nop               /* consumed here. Interrupt wakes the device */
cont: ...               /* User code executes here */
      powerc = 0x8000   /* Select CKI clock, turn on peripherals */
      6*nop             /* Wait for it to take effect */
      ioc = 0x0000      /* CKO is free-running */
```

8 Digital Signal Processor (DSP) Block (continued)

8.14 Power Management (continued)

Software Stop with Small-Signal Clock Buffer Running. In this case, all internal clocking is disabled and ICP interrupt is used to re-enable the clocks. Alternatively, INT0 can be used to re-enable the clocks. The DSP restarts with the crystal oscillator as the processor clock before changing to CKI. The PLL is not selected and is disabled (powered off).

```
      powerc = 0x8449   /* Select oscillator for slow clock, assert ICPEN */
                        /* turn off peripherals */
      6*nop             /* Wait for it to take effect */
      inc = NO_ICP      /* Disable the ICP interrupt if its corresponding service
                           routine need not be executed. The ICP interrupt is used
                           only to reset the nock bit of powerc. */
      ioc = 0x01E0      /* Hold CKO high */
nock: powerc = 0x8649   /* NOCK bit set to stop clocks */
                        /* No switching power consumed in the DSP section here */
                        /* Small-signal clock buffer still runs */
      6*nop             /* Wait for clocks to stop */
                        /* ICP interrupt (asserted by ARM) clears the NOCK bit,
                           clocking resumes */
cont: ...               /* User code executes here */
      powerc = 0x8000   /* Clear ICPEN bit, select CKI, turn on peripherals */
      6*nop             /* Wait for it to take effect */
      ins = 0x0080      /* Clear the ICP status bit */
      ioc = 0x0000      /* CKO is free-running */
```

Software Stop with Small-Signal Clock Buffer Disabled. In this case, all the internal clocking is disabled and ICPINT interrupt is used to re-enable the clocks. Alternatively, INT0 can be used to re-enable the clocks. The DSP first switches over to the crystal oscillator clock (OSC), and then sets the SSOFFD bit in **powerc** to shut off the small-signal clock buffer. It also sets the ICPIEN bit of the **powerc** register to allow the ICPINT interrupt wake up the DSP.

Next, it sets NOCK to 1 in **powerc** and freezes the DSP's clocks. If the CP does not need any clocks, then the entire device consumes minimum power for this case.

The DSP wakes up when NOCK is reset by an ICPINT interrupt. It restarts execution using the crystal oscillator clock. The SSOFFD bit is reset, and the DSP waits for a period longer than the small-signal buffer start-up time before switching over to CKI.

8 Digital Signal Processor (DSP) Block (continued)

8.14 Power Management (continued)

The following segment of code illustrates this case.

```
      powerc = 0xA449   /* Select OSC as slow clock, assert ICPIEN */
                        /* Turn off peripherals */
      6*nop             /* Wait for it to take effect */
      inc = NO_ICPINT   /* Load appropriate pattern in inc0 to
                         disable ICPINT if its service routine need not be executed */
      ioc = 0x01E0      /* Hold CKO high */
      powerc = 0xE449   /* Assert SSOFFD to disable small-signal buffer */
      2*nop             /* Wait for it to take effect */
      powerc = 0xE649   /* NOCK bit set to stop system clock */
      6*nop             /* Wait for clock to stop */
                        /* No switching power consumed in DSP section here */
                        /* ICP interrupt clears NOCK; DSP execution resumes on OSC */
      powerc = 0x8400   /* Enable small-signal buffer, turn on
                         peripherals, and continue running on OSC */
      2*nop             /* Wait for it to take effect */
      c0 = 1 - (SSBUFF_STARTUPTIME/(2 * OSC_PERIOD))
ss_buzz: if c0lt goto ss_buzz/* wait here for SSBUFF_STARTUPTIME */
      powerc = 0x8000   /* Select CKI as DSP clock */
      6*nop
      ins = 0x0080      /* Clear ICPINT bit in ins if it need not be serviced */
      ioc = 0x0000      /* CKO is free-running */
```

8.14.5 Power Management Examples with the PLL

The following examples show options for reducing power dissipation if operation with the PLL clock synthesizer is desired.

Note: For all cases in which the PLL is enabled, the input to the clock synthesizer, CKI, must remain running.

Low-Power Standby Mode, PLL Enabled and Selected. In this case, the PLL is enabled and selected to run at 91 MHz before the device is put into standby mode via AWAIT in **alf**. This reduces the power dissipation while waiting for an interrupt to continue program execution; however, the PLL continues to run and dissipate power.

```
pllinit:
      pllc = 0x0A00     /* Deselect PLL, P = 5 */
      6*nop
      powerc = 0x0000   /* Power on PLL */
      pllc = 0xDA00     /* Set PLLSEL, DIV_RSTN and await PLL lock */
                        /* DSP will switch to 91 MHz after PLL locks. */
      6*nop
```

8 Digital Signal Processor (DSP) Block (continued)

8.14 Power Management (continued)

```
                              . . .
      powerc = 0x2009    /* Turn off peripherals. */
      2*nop              /* wait for it to take effect */
      ioc = 0x01E0       /* Hold CKO high */
standby:alf = 0x8000     /* Set AWAIT, stop internal processor clock*/
      nop                /* Interrupts active */
      nop                /* Needed for bedtime execution */
                         /* Only standby power plus PLL power */
      nop                /* consumed here. Any enabled interrupt wakes up the device */
cont: ...                /* User code executes here */
      powerc = 0x0000    /* Turn peripheral units back on */
      2*nop              /* Wait for it to take effect */
      ioc = 0x0000       /* CKO is free-running */
```

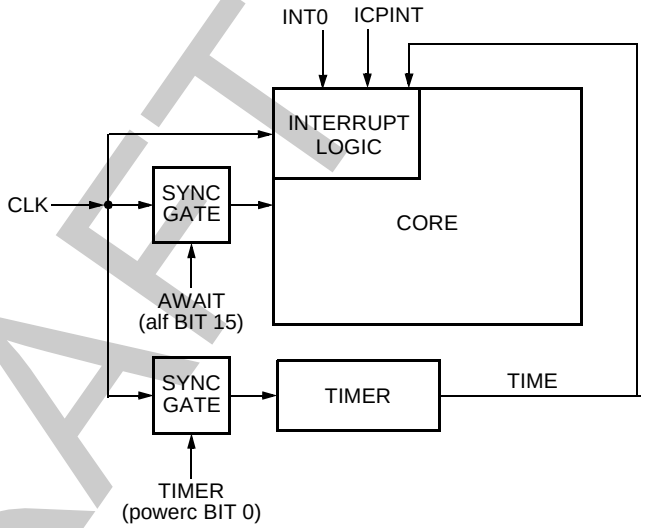


Figure 8.14-2 Low-Power Standby Control of Core Interrupt and the Peripherals

8 Digital Signal Processor (DSP) Block (continued)

8.14 Power Management (continued)

Low-Power Standby Mode, PLL Enabled and Not Selected. In this case, the PLL is enabled to run at 91 MHz but is not selected. The crystal oscillator (OSC) is selected to clock the processor before the device is put into standby mode via AWAIT in **alf**. This reduces the power dissipation while waiting for an interrupt to continue program execution; however, the PLL continues to run and dissipate power.

```
pllinit:
      pllc = 0x0A00     /* Deselect PLL, P = 5 */
      6*nop
      powerc = 0x0000   /* Power on PLL */
      pllc = 0xDA00     /* Set PLLSEL, DIV_RSTN and await PLL lock */
                        /* DSP will switch to 91 MHz after PLL locks. */
      6*nop
                              . . .
      powerc = 0x2409   /* Turn off peripherals and select OSC for slow clock */
      2*nop             /* Wait for it to take effect */
      pllc = 0x1A00     /* Clear PLLSEL to deselect PLL */
      6*nop             /* Wait for system clock to switch from PLL to OSC */
      ioc = 0x01E0      /* Hold CKO high */
standby:alf = 0x8000    /* Set AWAIT, stop internal processor clock*/
      nop               /* Interrupts active */
      nop               /* Bedtime execution. Reduced to standby power mode */
      nop               /* consumed here. Any enabled interrupt wakes up device */
cont: ...
      pllc = 0xDA00     /* Set PLLSEL bit. System clock switches from OSC to PLL */
                        /* since PLL already locked */
      6*nop
      powerc = 0x0000   /* Turn on peripherals and select CKI for slow clock */
      6*nop             /* Wait for peripheral enable and PLL switch to take effect */
      ioc = 0x0000      /* CKO is free-running */
```

8 Digital Signal Processor (DSP) Block (continued)

8.14 Power Management (continued)

Software Stop, PLL Enabled and Not Selected. In this case, all internal clocking is disabled and ICPINT is used to re-enable the clocks. Device restarts with the crystal oscillator (OSC) as the processor clock before changing to the PLL clock. Alternatively, INT0 can be used to re-enable the clocks. The PLL continues to run and dissipate power.

```
pllinit:
      pllc = 0x0A00     /* Deselect PLL, P = 5 */
      6*nop
      powerc = 0x0000   /* Power on PLL */
      pllc = 0xDA00     /* Set PLLSEL, DIV_RSTN and await PLL lock */
                        /* DSP will switch to 91 MHz after PLL locks. */
      6*nop
                              . . .
      powerc = 0x2449   /* Select OSC as slow clock, set ICPEN, turn off peripherals */
      2*nop             /* Wait for it to take effect */
      pllc = 0x1A00     /* Clear PLLSEL to deselect PLL */
      6*nop             /* Wait for system clock to switch from PLL to OSC */
      inc0 = NO_ICP     /* Disable the ICP interrupt if it need not be serviced */
      ioc = 0x01E0      /* Hold CKO high */
nock: powerc = 0x3E7F   /* NOCK bit set to stop system clock */
      6*nop             /* Wait for clock to stop */
                        /* Minimum switching power consumed here (assuming ARM also
                         off*/
                           . . .
                        /* ICPINT signal clears the NOCK bit, clocking resumes */
cont: ...
      pllc = 0xDA00     /* Set PLLSEL bit. System clock switches from OSC to PLL */
                        /* since PLL already locked */
      6*nop
      powerc = 0x0000   /* Turn on peripherals and select CKI for slow clock */
      6*nop             /* Wait for peripheral enable and PLL switch to take effect */
      ins = 0x0080      /* Clear the ICP status bit */
      ioc = 0x0000      /* CKO is free-running */
```

8 Digital Signal Processor (DSP) Block (continued)

8.14 Power Management (continued)

Software Stop, PLL Disabled and Not Selected. In this case, all internal clocking and the PLL is disabled and ICPINT is used to re-enable the clocks. Device restarts with the crystal oscillator as the processor clock before starting and changing to the high-speed PLL clock. Alternatively, INT0 can be used to re-enable the clocks.

```
pllinit:
      pllc = 0x0A00     /* Deselect PLL, P = 5 */
      6*nop
      powerc = 0x0000   /* Power on PLL */
      pllc = 0xDA00     /* Set PLLSEL, DIV_RSTN and await PLL lock */
                        /* DSP will switch to 91 MHz after PLL locks. */
                        /* Execute instructions until powerdown */
      6*nop
                              . . .
      powerc = 0x2449   /* Select OSC as slow clock, set ICPEN, turn off peripherals */
      2*nop             /* Wait for it to take effect */
      pllc = 0x1A00     /* Clear PLLSEL to deselect PLL */
      6*nop             /* Wait for system clock to switch from PLL to OSC */
      inc0 = NO_ICP     /* Disable the ICP interrupt if it need not be serviced */
      powerc = 0xE449   /* Shut off PLL and small-signal clock buffer */
      2*nop             /* Wait for it to take effect */
      ioc = 0x01E0      /* Hold CKO high */
nock: powerc = 0xE649   /* NOCK bit set to stop system clock */
      6*nop             /* Wait for clock to stop */
                        /* Minimum switching power consumed here */
                        /* Wait for ICPINT */
                              . . .
                        /* ICPINT signal clears the NOCK bit, clocking resumes */
cont: ...
      powerc = 0x0400   /* Enable PLL, small-signal buffer, turn on */
                        /* peripherals, and continue to run on OSC */
      2*nop             /* Wait for it to take effect */
      c0 = 1 – SSBUFF_STARTUPTIME/(2 * OSC_PERIOD))
ss_buzz: if c0lt goto ss_buzz /* wait here for SSBUFF_STARTUPTIME */
      powerc = 0x0000   /* Switch to CKI clock */
      6*nop             /* Wait for it to take effect */
      ins = 0x0080      /* Clear the ICPINT status bit */
      ioc = 0x0000      /* CKO is free-running */
      pllc = 0xDA00     /* Re-enable PLL and switch to PLL clock after PLL lock */
      6*nop             /* Wait for sync MUX */
```

8 Digital Signal Processor (DSP) Block (continued)

8.14 Power Management (continued)

The previous examples are not an exhaustive list of options available to the user. Other clocking possibilities exist. These depend on all of the following:

- The clock source to the processor.
- Whether the user chooses to power down the peripheral units.
- Whether the internal processor clock is disabled through software.
- The combination of power management modes chosen.
- Whether or not the PLL is enabled.

8.14.6 Considerations in Standby Mode

A program running in the core can place it into low-power standby mode by setting the AWAIT field (**alf**[15]; see Table 8.15-6). In this mode, the clock to the core and its associated DPRAM are disabled except for the minimum core circuitry required to process an incoming interrupt or trap. The clock to the peripherals is unaffected.

Figure 8.14-2 illustrates the following:

- Distribution of CLK to the core and peripherals.
- Function of the AWAIT field.
- Interrupts to the core used to exit low-power standby mode.
- DSP CKO_IACK pin selection logic (see Section 8.13.4 for details).

If the core is in low-power standby mode, its program execution is suspended without loss of state. If an interrupt that was enabled by the core occurs or if a trap occurs, the core clears its AWAIT field, exits low-power standby mode, resumes program execution, and services the interrupt or trap. See Section 8.4.2 and Section 8.4.3 for information on enabling interrupts.

If DSP is entering low-power standby mode, it can further save power by doing one or more of the following prior to entering standby mode:

1. Select the CKI pin as the source clock to the core and peripherals by clearing the PLLSEL field in **pllc** (see Table 8.15-18) and clearing the SLOWCK bit in **powerc** (see Table 8.15-20).
2. Disable (power down) the PLL by setting the PLLPD field (**powerc**).
3. Select 32 kHz RTC and powerdown CKI buffer as the source clock to the core and peripherals by clearing the PLLSEL field in **pllc** and setting the SLOWCK bits in **powerc**.

The above options result in increased wake-up latency, which is the delay from the time that the core exits standby mode (due to an interrupt) to the time that the core resumes full-speed execution. Before selecting these options, the programmer must ensure that the increased wake-up latency is acceptable in the application. Table 8.14-1 compares the wake-up latency for various selections of clocks during standby mode. It also illustrates the trade-off of wake-up latency vs. power consumption. Disabling the PLL and CKI buffer during low-power standby mode results in the minimum power consumption and highest wake-up latency.

8 Digital Signal Processor (DSP) Block (continued)

8.14 Power Management (continued)

Table 8.14-1 Wake-Up Latency and Power Consumption for Low-Power Standby Mode

Source Clock Selected In Standby Mode	Status of PLL In Standby Mode	Wake-Up Latency	Latency vs. Power Consumption Trade-off
PLL	PLL Enabled	3 PLL Clock Cycles	←Minimum wake-up latency (highest power).
CKI	PLL Enabled	14 CKI Clock Cycles (minimum)	←Minimum power (highest wake-up latency).
	PLL Disabled	3 CKI Clock Cycles (minimum)	←Minimum power (highest wake-up latency).
X1RTC	PLL Disabled (CKI Buffer Also Disabled)	3 RTC clock cycles	

If the program running in DSP selects the CKI pin as the source clock before entering standby mode, that clock is selected as the source clock immediately after the core exits standby mode. Likewise, if the program running in DSP disables the PLL before entering standby mode, the PLL is disabled immediately after the core exits standby mode. Assuming the PLL is the source clock for normal operation, the DSP program must re-enable and then reselect the PLL after exiting standby mode in order to resume full-speed processing.

An interrupt causes the core to exit standby mode and immediately service the interrupt. If the interrupt service routine (ISR) performs time-critical processing, it must re-enable and then reselect the PLL before performing any processing to service the interrupt.

If the program selects the CKI pin as the source clock before entering standby mode, the peripherals also operate at the slower rate. This can result in an increased delay for a peripheral to interrupt the core to exit standby mode.

If DSP is entering low-power standby mode, it can save additional power by powering down its timer (set **timerc**[6]) prior to entering low-power standby mode. Section 8.6 describes the procedures for powering down the timer.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers

T8307 DSP block registers fall into one of the following three categories:

- Directly program-accessible (or register-mapped) registers are directly accessible in instructions and are designated with lower-case bold (e.g., **timer**). These registers are described in Section 8.15.1.
- Memory-mapped registers are accessible at a memory address and are designated with upper-case bold (e.g., **DSTAT**). These registers are described in Section 8.15.2.
- Pin-accessible registers are accessible only through the external device pins and are designated with upper-case bold (i.e., **ID**). Each JTAG port contains the pin-accessible identification register, **ID**, described in Table 8.9-1. This register is accessible via its associated JTAG port.

Note: The program counter (**PC**) is an addressing register not accessible to the programmer or through external pins. The core automatically controls this register to properly sequence the instructions.

8.15.1 Directly Program-Accessible (Register-Mapped) Registers

Figure 8.15-1 depicts the directly program-accessible (register-mapped) registers. The figure differentiates core and off-core registers.

Note: There is write-to-read latency associated with the pipelined IDB. The assembler compensates for this. See the *DSP16000 Digital Signal Processor Core* Information Manual for further details.

As shown in Figure 8.15-1, the register-mapped registers consist of three types:

Data registers store data either from the result of instruction execution or from memory. Data registers become source operands for instructions. This class of registers also includes postincrement registers whose contents are added to address registers to form new addresses.

Control and Status registers are used to determine the state of the machine or to set different configurations to control the machine.

Address registers are used to hold memory location pointers. In some cases, the user can treat address registers as general-purpose data registers accessible by data move instructions.

Table 8.15-1 summarizes the register-mapped registers. It lists all valid register designators as they appear in an instruction syntax. For each register, the table specifies its size, whether it is readable or writable, its type, whether it is signed or unsigned, and the hardware function block in which it is located. It also indicates whether the register is in the core or is off-core. Off-core register-mapped registers cannot be stored to memory in a single instruction.

To store the contents of an off-core register to memory, first store the register to an intermediate register and then store the intermediate register to memory.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

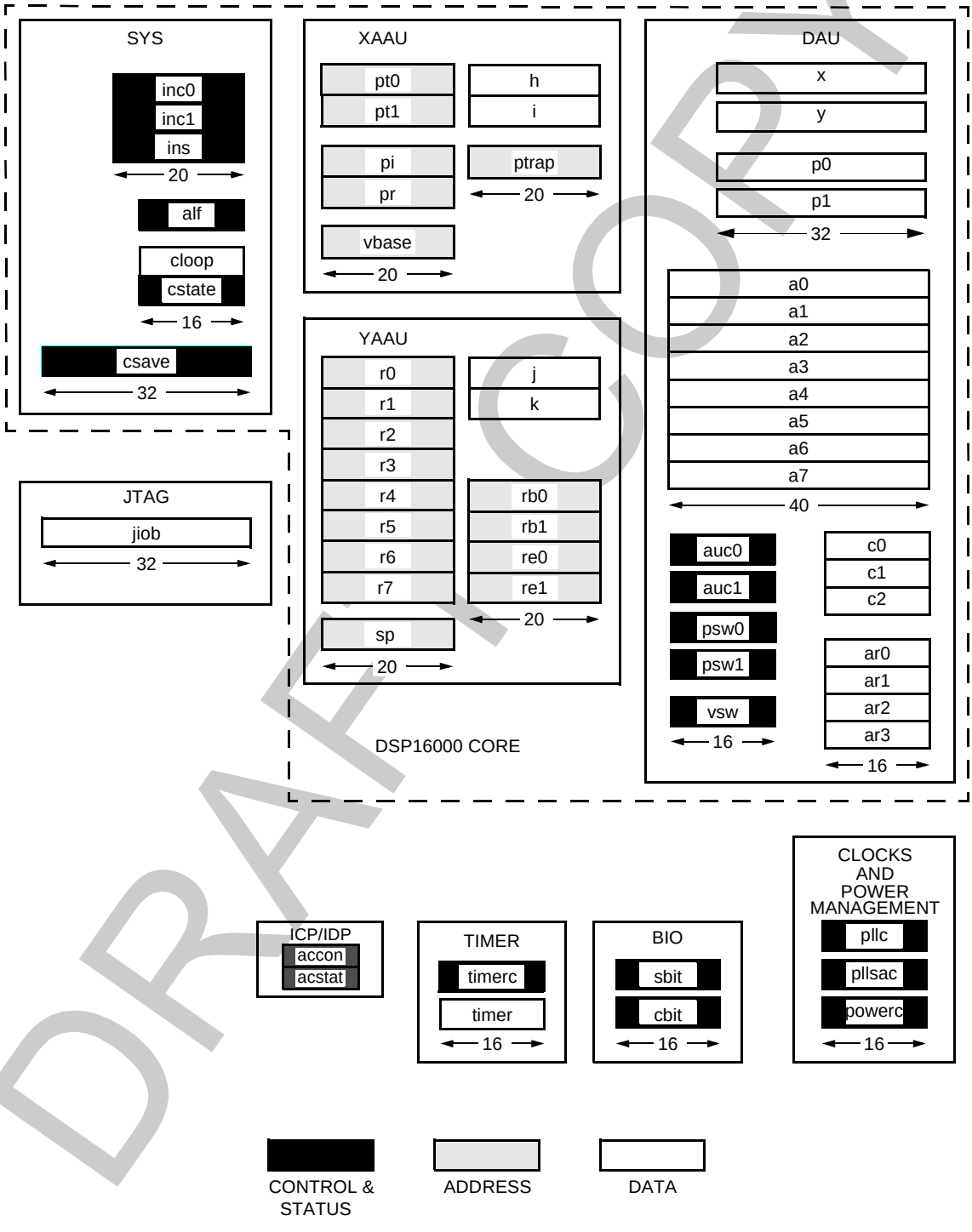


Figure 8.15-1 T8307 DSP Block Program-Accessible Registers

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-1 Program-Accessible (Register-Mapped) Registers by Type, Listed Alphabetically

Register Name	Description	Size (Bits)	R/W†	Type‡	Signed§/ Unsigned	Core/ Off-Core	Function Block
a0, a1, a2, a3, a4, a5, a6, a7	Accumulators 0—7	40	R/W	data	signed	core	DAU
a0h, a1h, a2h, a3h, a4h, a5h, a6h, a7h	Accumulators 0—7, high halves (bits 31—16)	16	R/W	data	signed	core	DAU
a0l, a1l, a2l, a3l, a4l, a5l, a6l, a7l	Accumulators 0—7, low halves (bits 15—0)	16	R/W	data	signed	core	DAU
a0g, a1g, a2g, a3g, a4g, a5g, a6g, a7g	Accumulators 0—7, guard bits (bits 39—32)	8	R/W	data	signed	core	DAU
a0_1h, a2_3h, a4_5h, a6_7h	Accumulator vectors (concatenated high halves of two adjacent accumulators)	32	R/W	data	signed	core	DAU
accon	ICP control registers	16	R/W	control	unsigned	off-core	ICP
acstat	ICP status registers	16	R/W	c & s	unsigned	off-core	ICP
ahcon	IDP control register	16	R/W	control	unsigned	off-core	IDP
ahstat	IDP status register	16	R	status	unsigned	off-core	IDP
alf	AWAIT and flags	16	R/W	c & s	unsigned	core	SYS
ar0, ar1, ar2, ar3	Auxiliary registers 0—3	16	R/W	data	signed	core	DAU
auc0, auc1	Arithmetic unit control	16	R/W	c & s	unsigned	core	DAU
c0, c1	Counters 0 and 1	16	R/W	data	signed	core	DAU
c2	Counter holding register	16	R/W	data	signed	core	DAU
cbit	BIO control	16	R/W	control	unsigned	off-core	BIO
cloop	Cache loop count	16	R/W	data	unsigned	core	SYS
csave	Cache save	32	R/W	control	unsigned	core	SYS
cstate	Cache state	16	R/W	control	unsigned	core	SYS
h	Pointer postincrement	20	R/W	data	signed	core	XAAU
i	Pointer postincrement	20	R/W	data	signed	core	XAAU
ioc	Memory configuration register—clock and memory map selection	16	R/W	control	unsigned	off-core	SEMI
inc0, inc1	Interrupt control 0 and 1	20	R/W	control	unsigned	core	SYS
ins	Interrupt status	20	R/C††	status	unsigned	core	SYS
j	Pointer postincrement/offset	20	R/W	data	signed	core	YAAU
jhb	High byte of **j** (bits 15—8)	8	R	data	unsigned	core	YAAU
jlb	Low byte of **j** (bits 7—0)	8	R	data	unsigned	core	YAAU
jiob	JTAG test	32	R/W	data	unsigned	off-core	JTAG
k	Pointer postincrement/offset	20	R/W	data	signed	core	YAAU
p0	Product 0	32	R/W	data	signed	core	DAU

† R indicates that the register is readable by instructions; W indicates the register is writable by instructions.
‡ c & s means control and status.
§ Signed registers are in two's complement format.
†† C indicates that the register is cleared and not set.
‡‡ The IEN field (bit 14) of the **psw1** register is read only (writes to this bit are ignored).
§§ The VALUE[6:0] field (bits 6—0) are read only (writes to these bits are ignored).

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-1 Program-Accessible (Register-Mapped) Registers by Type, Listed Alphabetically (continued)

Register Name	Description	Size (Bits)	R/W†	Type‡	Signed§/ Unsigned	Core/ Off-Core	Function Block
p0h	High half of **p0** (bits 31—16)	16	R/W	data	signed	core	DAU
p0l	Low half of **p0** (bits 15—0)	16	R/W	data	signed	core	DAU
p1	Product 1	32	R/W	data	signed	core	DAU
p1h	High half of **p1** (bits 31—16)	16	R/W	data	signed	core	DAU
p1l	Low half of **p1** (bits 15—0)	16	R/W	data	signed	core	DAU
pi	Program interrupt return	20	R/W	address	unsigned	core	XAAU
pllc	Phase-locked loop control (DSP0 only)	16	R/W	control	unsigned	off-core	Clocks
pllsac	Phase-locked loop status	16	R	status	unsigned	off-core	Clocks
powerc	Power control	16	R/W	control	unsigned	off-core	Clocks
pr	Subroutine return	20	R/W	address	unsigned	core	XAAU
psw0, psw1	Program status words 0 and 1	16	R/W‡‡	c & s	unsigned	core	DAU
pt0, pt1	Pointers 0 and 1 to X-memory space	20	R/W	address	unsigned	core	XAAU
ptrap	Program trap return	20	R/W	address	unsigned	core	XAAU
r0, r1, r2, r3, r4, r5, r6, r7	Pointers 0—7 to Y-memory space	20	R/W	address	unsigned	core	YAAU
rb0, rb1	Circular buffer pointers 0 and 1 (begin address)	20	R/W	address	unsigned	core	YAAU
re0, re1	Circular buffer pointers 0 and 1 (end address)	20	R/W	address	unsigned	core	YAAU
sbit	BIO status/control	16	R/W§§	c & s	unsigned	off-core	BIO
sp	Stack pointer	20	R/W	address	unsigned	core	YAAU
timer	Timer running count for Timer	16	R/W	data	unsigned	off-core	Timer
timerc	Timer control	16	R/W	control	unsigned	off-core	Timer
vbase	Vector base offset	20	R/W	address	unsigned	core	XAAU
vsw	Viterbi support word	16	R/W	control	unsigned	core	DAU
x	Multiplier input	32	R/W	data	signed	core	DAU
xh	High half of **x** (bits 31—16)	16	R/W	data	signed	core	DAU
xl	Low half of **x** (bits 15—0)	16	R/W	data	signed	core	DAU
y	Multiplier input	32	R/W	data	signed	core	DAU
yh	High half of **y** (bits 31—16)	16	R/W	data	signed	core	DAU
yl	Low half of **y** (bits 15—0)	16	R/W	data	signed	core	DAU

† R indicates that the register is readable by instructions; W indicates the register is writable by instructions.
‡ c & s means control and status.
§ Signed registers are in two's complement format.
†† C indicates that the register is cleared and not set.
‡‡ The IEN field (bit 14) of the **psw1** register is read only (writes to this bit are ignored).
§§ The VALUE[6:0] field (bits 6—0) are read only (writes to these bits are ignored).

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-2 through Table 8.15-26 list register encodings for register-mapped registers.

Table 8.15-2 ACCON Control Register in ICP (Controlled by the DSP16000 Core)

Bit	15	14	13	12	11	10	9—8	7—0
Name	DCSIG5	DCSIG4	DCSIG3	DCSIG2	DCSIG1	DCSIG0	RSVD	AIRQ[0:7]

Bit	Name	Description*
15	DCSIG5	User handshake signal 5 from the DSP16000 core to *ARM* core—readable by *ARM* core via DCSTAT register bit 15.
14	DCSIG4	User handshake signal 4 from the DSP16000 core to *ARM* core—readable by *ARM* core via DCSTAT register bit 14.
13	DCSIG3	User handshake signal 3 from the DSP16000 core to *ARM* core—readable by *ARM* core via DCSTAT register bit 13.
12	DCSIG2	User handshake signal 2 from the DSP16000 core to *ARM* core—readable by *ARM* core via DCSTAT register bit 12.
11	DCSIG1	User handshake signal 1 from the DSP16000 core to *ARM* core—readable by *ARM* core via DCSTAT register bit 11.
10	DCSIG0	User handshake signal 0 from the DSP16000 core to *ARM* core—readable by *ARM* core via DCSTAT register bit 10.
9—8	RSVD	Reserved.
7—0	AIRQ[0:7]	Interrupt request 0 through 7 to *ARM* core—readable and clearable by *ARM* core via DCSTAT register bits 0—7.

* All reset values are 0.

Table 8.15-3 ACSTAT Status Register in ICP (Controlled by the DSP16000 Core)

Bit	15	14	13	12	11	10	9—8	7—0
Name	ACSIG5	ACSIG4	ACSIG3	ACSIG2	ACSIG1	ACSIG0	RSVD	DIRQ[0:7]

Bit	Name	Description*
15	ACSIG5	User handshake signal 5 from *ARM* core to the DSP16000 core—reflects value in bit 15 of DCCON register.
14	ACSIG4	User handshake signal 4 from *ARM* core to the DSP16000 core—reflects value in bit 14 of DCCON register.
13	ACSIG3	User handshake signal 3 from *ARM* core to the DSP16000 core—reflects value in bit 13 of DCCON register.
12	ACSIG2	User handshake signal 2 from *ARM* core to the DSP16000 core—reflects value in bit 12 of DCCON register.
11	ACSIG1	User handshake signal 1 from *ARM* core to the DSP16000 core—reflects value in bit 11 of DCCON register.
10	ACSIG0	User handshake signal 0 from *ARM* core to the DSP16000 core—reflects value in bit 10 of DCCON register.
9—8	RSVD	Reserved.
7—0	DIRQ[0:7]	Interrupt request 0 through 7 to the DSP core from *ARM* core. Writing a one to each bit acknowledges the interrupt, clearing DIRQ[n] in both DCCON and ACSTAT registers. Writing a zero to each bit leaves the value unchanged.

* All reset values are 0.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-4 AHCON Control Register in IDP (Controlled by the DSP16000 Core)

Bit	15	14	13—9	8	7	6	5	4	3	2	1	0
Name	DHSIG1	DHSIG0	RSVD	ADHWAIT	AAHWAIT	AAHSLP	ADSLP	AASLP	ASYTRAP	AEXTERN	AA_DBK	AD_ABK

Bit	Name	Description*
15	DHSIG1	HDS handshake signal 1 from the DSP16000 core to *ARM* core; readable by the DSP16000 core via DHSTAT register bit 15.
14	DHSIG0	HDS handshake signal 0 from the DSP16000 core to *ARM* core; readable by the DSP16000 core via DHSTAT register bit 14.
13—9	RSVD	Reserved.
8	ADHWAIT	If set, stop the DSP16000 core on exit from HDS trap handler until *ARM* core also exits from DBGRQ handler.
7	AAHWAIT	If set, stop the *ARM* core on exit from DBGRQ handler until the DSP16000 core also exits from HDS trap handler.
6	AAHSLP	If set, stop the *ARM* core automatically whenever the DSP16000 core is executing in debug mode (based on value of HACTIVE signal).
5	ADSLP	If set, allow the *ARM* core to stop the DSP16000 core either by setting DHCON[5] = 1 or by setting DHCON[6] = 1 and entering debug mode.
4	AASLP	If set and DHCON[4] = 1, stop the *ARM* core until AASLP is subsequently cleared by the DSP16000 core.
3	ASYTRAP	If set, allow *ARM* core to assert SYSTRAP by setting DHCON[3] = 1.
2	AEXTERN	If set and DHCON[2] = 1, generate an immediate DBGRQ request to the *ARM* ICE module.
1	AA_DBK	If set and DHCON[1] = 1, the detection of a breakpoint to the *ARM* ICE module will also assert SYSTRAP to the DSP16000.
0	AD_ABK	If set and DHCON[0] = 1, the assertion of HTRAP to the DSP16000 will also assert DBGRQ to the *ARM* ICE module.

* All reset values are 0.

Table 8.15-5 AHSTAT Status Register in IDP (Readable by the DSP16000 Core)

Bit	15	14	13—5	4	3	2	1	0
Name	AHSIG1	AHSIG0	RSVD	ASLRDY	ASREQ	ADREQ	ADEBUG	ASLEEP

Bit	Name	Description*
15	AHSIG1	HDS handshake signal 1 from *ARM* core to the DSP16000 core; reflects value in bit 15 of DHCON register.
14	AHSIG0	HDS handshake signal 0 from *ARM* core to the DSP16000 core; reflects value in bit 14 of DHCON register.
13—5	RSVD	Reserved.
4	ASLRDY	If 1, the *ARM* core will allow itself to be stopped by the DSP16000 core (reflects the status of DHCON[4]). The DSP16000 core can then stop the *ARM* core by setting AHCON[4].
3	ASREQ	If 1, the *ARM* core is trying to stop the DSP16000 core, due to DBG_ACK and DHCON[6] = 1, or if DHCON[5] = 1.
2	ADREQ	If 1, the *ARM* core is trying to assert SYSTRAP to the DSP16000, due to DBGRQ and DHCON[1] = 1, or if DHCON[3] = 1.
1	ADEBUG	If 1, the *ARM* core is currently executing in debug mode (reflects the value of DBG_ACK).
0	ASLEEP	If 1, the *ARM* core has been stopped via AHCON bit 4 or bit 6 (AASLP/AAHSLP).

* All reset values are 0.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-6 through Table 8.15-26 list register encodings for register-mapped registers. For the encoding of **accon**, **acstat**, **ahcon** and **ahstat** registers, see Table 9.1-3, Table 9.1-4, Table 9.2-3, and Table 9.2-4.

Table 8.15-6 alf (AWAIT Low-Power and Flag) Register

Bit	15	14—10	9	8	7	6	5—0
Name	AWAIT	RSVD	JOBF	JIBE	JCONT	LOCK	RSVD

Bit	**Name**	**Value**	**Description**	**R/W**	**Reset Value***
15	AWAIT	0	Core operates normally.	R/W	0
		1	Core enters power-saving standby mode.		
14—10	RSVD	0	Reserved—write with 0.	R/W	0
9	JOBF	0	JTAG **jiob** output buffer is empty.	R/W	X
		1	JTAG **jiob** output buffer is full.		
8	JIBE	0	JTAG **jiob** input buffer is full.	R/W	X
		1	JTAG **jiob** input buffer is empty.		
7	JCONT	—	JTAG continue flag.	R/W	X
6	LOCK	0	The PLL is not locked.	R/W	0
		1	The PLL is locked.		
5—0	RSVD	0	Reserved—write with 0.	R/W	0

* For this column, X indicates unknown on external reset and unaffected on subsequent reset.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-7 auc0 (Arithmetic Unit Control 0) Register

Bit	15—14	13—11	10	9	8	7	6	5—4	3—2	1—0
Name	P1SHFT[1:0]	RSVD	FSAT	SHFT15	RAND	X=Y=	YCLR	ACLR[1:0]	ASAT[1:0]	P0SHFT[1:0]

Bit	Name	Value	Description	R/W	Reset Value
15—14	P1SHFT[1:0]	00	**p1** not shifted.	R/W	00
		01	**p1**>>2.		
		10	**p1**<<2.		
		11	**p1**<<1.		
13—11	RSVD	0	Reserved—write with 0.	R/W	0
10	FSAT	0	Disabled when zero.	R/W	0
		1	Enable 32-bit saturation for the following results: the scaled outputs of the **p0** and **p1** registers, the intermediate result of the 3-input ADDER*, and the results of the ALU/ACS, ADDER/ACS, and BMU.	R/W	0
9	SHFT15	0	**p1**>>15 in F1E operations performs normally.	R/W	0
		1	To support GSM-EFR, **p1**>>15 in F1E operations actually performs (**p1**>>16)<<1, clearing the least significant bit.		
8	RAND	0	Enable pseudorandom sequence generator (PSG).†	R/W	0
		1	Reset and disable pseudorandom sequence generator (PSG).		
7	X=Y=	0	Normal operation.	R/W	0
		1	Data transfer statements that load the **y** register also load the **x** register with the same value.‡		
6	YCLR	0	The DAU clears **yl** if it loads **yh**.	R/W	0
		1	The DAU leaves **yl** unchanged if it loads **yh**.		
5	ACLR[1]	0	The DAU clears **a1l** if it loads **a1h**.	R/W	0
		1	The DAU leaves **a1l** unchanged if it loads **a1h**.		
4	ACLR[0]	0	The DAU clears **a0l** if it loads **a0h**.	R/W	0
		1	The DAU leaves **a0l** unchanged if it loads **a0h**.		
3	ASAT[1]	0	Enable **a1** saturation§ on 32-bit overflow.	R/W	0
		1	Disable **a1** saturation on 32-bit overflow.		
2	ASAT[0]	0	Enable **a0** saturation§ on 32-bit overflow.	R/W	0
		1	Disable **a0** saturation on 32-bit overflow.		
1—0	P0SHFT[1:0]	00	**p0** not shifted.	R/W	00
		01	**p0**>>2.		
		10	**p0**<<2.		
		11	**p0**<<1.		

* Saturation takes effect only if the ADDER has three input operands and there is no ALU/ACS operation in the same instruction.

† After re-enabling the PSG by clearing RAND, the program must wait one instruction cycle before testing the heads or tails condition.

‡ The following apply:
- Instructions that explicitly load any part of the **x** register (i.e., **x**, **xh**, or **xl**) take precedence over the X = Y = mode.
- Instructions that load **yh** (but not **x** or **xh**) load **xh** with the same data. If YCLR is zero, the DAU clears **yl** and **xl**.
- Instructions that load **yl** load **xl** with the same data and leave **yh** and **xh** unchanged.

§ If enabled, 32-bit saturation of the accumulator value occurs if the DAU stores the value to memory or to a register. Saturation also applies if the DAU stores the low half, high half, or guard bits of the accumulator. There is no change to the contents stored in the accumulator; only the value stored to memory or a register is saturated.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-8 auc1 (Arithmetic Unit Control 1) Register

Bit	15	14—12	11—6	5—0
Name	RSVD	XYFBK[2:0]	ACLR[7:2]	ASAT[7:2]

Bit	Name	Value	Description	R/W	Reset Value
15	RSVD	0	Reserved—write with 0.	R/W	0
14—12	XYFBK[2:0]*	000	Normal operation.	R/W	000
		001	Any DAU function result stored into **a6**[31:0] is also stored into **x**.†		
		010	Any DAU function result stored into **a6**[31:16] is also stored into **xh**.†		
		011	Any DAU function result stored into **a6**[31:16] is also stored into **xh**, and any DAU function result stored into **a7**[31:16] is also stored into **xl**.†		
		100	Reserved.		
		101	Any DAU function result stored into **a6**[31:0] is also stored into **y**.‡		
		110	Any DAU function result stored into **a6**[31:16] is also stored into **yh**.‡§		
		111	Any DAU function result stored into **a6**[31:16] is also stored into **yh**, and any DAU function result stored into **a7**[31:16] is also stored into **yl**.‡**		
11	ACLR[7]	0	The DAU clears **a7l** if it loads **a7h**.	R/W	0
		1	The DAU leaves **a7l** unchanged if it loads **a7h**.		
10	ACLR[6]	0	The DAU clears **a6l** if it loads **a6h**.	R/W	0
		1	The DAU leaves **a6l** unchanged if it loads **a6h**.		
9	ACLR[5]	0	The DAU clears **a5l** if it loads **a5h**.	R/W	0
		1	The DAU leaves **a5l** unchanged if it loads **a5h**.		
8	ACLR[4]	0	The DAU clears **a4l** if it loads **a4h**.	R/W	0
		1	The DAU leaves **a4l** unchanged if it loads **a4h**.		
7	ACLR[3]	0	The DAU clears **a3l** if it loads **a3h**.	R/W	0
		1	The DAU leaves **a3l** unchanged if it loads **a3h**.		
6	ACLR[2]	0	The DAU clears **a2l** if it loads **a2h**.	R/W	0
		1	The DAU leaves **a2l** unchanged if it loads **a2h**.		
5	ASAT[7]	0	Enable **a7** saturation†† on 32-bit overflow.	R/W	0
		1	Disable **a7** saturation on 32-bit overflow.		
4	ASAT[6]	0	Enable **a6** saturation†† on 32-bit overflow.	R/W	0
		1	Disable **a6** saturation on 32-bit overflow.		
3	ASAT[5]	0	Enable **a5** saturation†† on 32-bit overflow.	R/W	0
		1	Disable **a5** saturation on 32-bit overflow.		
2	ASAT[4]	0	Enable **a4** saturation†† on 32-bit overflow.	R/W	0
		1	Disable **a4** saturation on 32-bit overflow.		
1	ASAT[3]	0	Enable **a3** saturation†† on 32-bit overflow.	R/W	0
		1	Disable **a3** saturation on 32-bit overflow.		
0	ASAT[2]	0	Enable **a2** saturation†† on 32-bit overflow.	R/W	0
		1	Disable **a2** saturation on 32-bit overflow.		

* If the application enables any of the XYFBK modes (i.e., XYFBK[2:0] . 000), the following apply:
- Only if the DAU writes its result to **a6** or **a7** (e.g., **a6=a3+p0**) will the result be written to **x** or **y**. Data transfers or data move operations (e.g., **a6=*r2**) leave the **x** or **y** register unchanged regardless of the state of the XYFBK[2:0] field setting.
- If the instruction itself loads the same portion of the **x** or **y** register that the XYFBK[2:0] field specifies, the instruction load takes precedence.

† If the application enables the X=Y= mode (**auc0**[7] = 1), the XYFBK mode takes precedence.

‡ If the application enables the X=Y= mode (**auc0**[7] = 1), the DAU also writes the **y** register value into the **x**, **xh**, or **xl** register, as appropriate.

§ If the application enables the YCLR mode (**auc0**[6] = 0), the DAU clears **yl**.

** If the application enables the YCLR mode (**auc0**[6] = 0) and the instruction contains a result written to **a6** and the operation writes no result to **a7**, the DAU clears **yl**. If the application enables the YCLR mode and the instruction writes a result to **a7**, the XYFBK mode takes precedence and the DAU does not clear **yl**.

†† If saturation is enabled and any portion of an accumulator is stored to memory or a register, the DAU saturates the entire accumulator value and stores the appropriate portion. The DAU does not change the contents of the accumulator.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-9 BIO Control (cbit) Register

Bit	15—8	7—0
Name	MODE/MASK[7:0]	DATA/PAT[7:0]

DIREC[n]*	**MODE/MASK[n]***	**DATA/PAT[n]***	**Action**
1 (Output)	0	0	Clear
1 (Output)	0	1	Set
1 (Output)	1	0	No Change
1 (Output)	1	1	Toggle
0 (Input)	0	0	No Test
0 (Input)	0	1	No Test
0 (Input)	1	0	Test for Zero
0 (Input)	1	1	Test for One

* 0 = n = 7. DIREC[n] is a field in the **sbit** register.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-10 cloop (Cache Loop) Register

Bit	15—0
Name	Cache Loop Count

Bit	Name	Description	R/W	Reset Value
15—0	Cache Loop Count	Contains the count for the number of loop iterations for a **do K**, **redo K**, **do cloop**, or **redo cloop** instruction. The core decrements **cloop** after every loop iteration, and **cloop** contains zero after the loop has completed.	R/W	0

Table 8.15-11 csave (Cache Save) Register

Bit	31—0
Name	Cache Save

Bit	Name	Description	R/W	Reset Value*
31—0	Cache Save	Contains the opcode of the instruction following a **do K**, **redo K**, **do cloop**, or **redo cloop** instruction.	R/W	X

* For this column, X indicates unknown on external reset and unaffected on subsequent reset.

Table 8.15-12 cstate (Cache State) Register

Bit	15	14	13	12—10	9—5	4—0
Name	SU	EX	LD	RSVD	PTR[4:0]	N[4:0]

Bit	Name	Value	Description	R/W	Reset Value
15	SU	0	The cache is not suspended; the core is not executing an interrupt or trap service routine that has interrupted or trapped a cache loop.	R/W	0
		1	The cache is suspended; the core is executing an interrupt or trap service routine that has interrupted or trapped a cache loop.		
14	EX	0	The core is not executing from cache; it is either loading the cache (executing iteration 1 of a cache loop) or it is not executing a cache loop.	R/W	0
		1	The core is executing from cache; it is executing iteration 2 or higher of a cache loop.		
13	LD	0	The core is not loading the cache; it is either not executing a cache loop or it is executing iteration 2 or higher of a cache loop.	R/W	0
		1	The core is loading the cache; it is executing iteration 1 of a cache loop.		
12—10	RSVD	0	Reserved; write with 0.	R/W	0
9—5	PTR[4:0]	0—30	Pointer to current instruction in cache to load or execute.	R/W	0
4—0	N[4:0]*	0—31	Number of instructions in the cache loop to load/save/restore.	R/W	0

* After execution of the first **do K** or **do cloop** instruction, N[4:0] contains a nonzero value.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-13 I/O Configuration (ioc) Register

Bit	15—14	13	12—8	7—5	4—0
Name	RSVD	DSP_SELECT	RSVD	CKOSEL	RSVD

Bit	Name	Value	Description
15—14	RSVD	—	Reserved.
13	DSP_SELECT	0	1 allows DSP to take control over the shared PLL during DSP PLL speed tests. All control inputs to the shared PLL including P, M, N, PLL_PD, Autotrim reset, and PLL bypass are in control of DSP **pllc** register. If “0”, DSP does not control the PLL.
12—8	RSVD	—	Reserved.
7—5	CKOSEL	XXX	CKO pin output selection.
4—0	RSVD	—	Reserved.

Table 8.15-14 CKOSEL

Bit	Description
000	Select DSP system clock.
001	Reserved.
010	Reserved.
011	Reserved.
100	Select CKI from CKI pin.
101	DSP IACK.
110	Reserved.
111	Pull CKO_IACK pin high.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-15 inc0 and inc1 (Interrupt Control) Registers

Bit	19—18	17—16	15—2	1—0
inc0	RSVD	INT0[1:0]	RSVD	TIMER[1:0]

Bit	19—8	7—6	5—2	1—0
inc1	RSVD	ICP[1:0]	RSVD	SSP[1:0]

Name	Value	Description	R/W	Reset Value
INT0[1:0] TIMER[1:0] ICP[1:0] SSP[1:0]	00	Disable the selected interrupt (no priority).	R/W	00
	01	Enable the selected interrupt at priority 1 (lowest).		
	10	Enable the selected interrupt at priority 2.		
	11	Enable the selected interrupt at priority 3 (highest).		

Table 8.15-16 ins (Interrupt Status) Register

Bit	19—14	13	12—11	10	9	8	7—1	0
Name	RSVD	ICP	RSVD	SSP	RSVD	INT0	RSVD	TIMER

Name	Value	Description	R/W	Reset Value
INT0 SSP ICP TIMER	0	Read—corresponding interrupt not pending. Write—no effect.	R/Clear	0
	1	Read—corresponding interrupt is pending. Write—clears bit and changes corresponding interrupt status to not pending.		

Table 8.15-17 patchc Register

Bit	31	30—27	26—20	19—0
Name	S	P	RSVD	ADDRESS

Bit	Name	Description
31	S	1 = set, 0 = clear.
30—27	P	Patch number (0—15).
26—20	RSVD	Reserved.
19—0	ADDRESS	Address of the code to be patched.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-18 Phase-Locked Loop Control (pllc) Register

Bit	15	14	13	12	11—9	8—6	5—0
Name	PLLSEL	DIV_RSTN	TRMRST	AUTOSW	P[2:0]	N[2:0]	M[5:0]
Reset	0	0	0	1	111	000	00000

Bit	Name	Value	Description
15	PLLSEL	0	PLL not selected as DSP clock source.
		1	PLL selected as clock source.
14	DIV_RSTN	0	P divider reset (active-low). Must be brought low, then high, when updating P divider bits.
13	TRMRST	0	PLL trim feature enabled.
		1	Reset PLL trim circuit. Operation of trim feature disabled. PLL will perform trim sequence when bit is cleared.
12	AUTOSW	0	Switch to PLL immediately upon setting PLLSEL regardless of lock.
		1	Delay switchover to PLL until lock detect asserted.
11—9	P[2:0]*	—	PLL VCO frequency postdivider. Divides VCO frequency by P + 1.
8—6	N[2:0]	—	PLL REFCK frequency predivider. Divides input reference by N + 1.
5—0	M[5:0]	—	PLL VCO frequency multiplier. Multiplies VCO frequency by M + 1.

* For proper initialization of divider logic, make sure P is odd so that P+1 is even.

Table 8.15-19 Phase-Locked Loop Status (pllsac) Register

Bit	15—5	4	3	2	1—0
Name	RSVD	JTRPEN	FINLCK	CRSLCK	ACTVCLK
Reset	0	0	0	0	00

Bit	Name	Value	Description
15—5	RSVD	0	Reserved—return zero when read.
4	JTRPEN	1	Reset of NOCK in powerc register via JTAG trap is enabled.
3	FINLCK	0	PLL has not obtained high precision lock.
		1	PLL has obtained high precision lock (frequency within 0.1% of target).
2	CRSLCK	0	PLL has not obtained low precision lock.
		1	PLL has obtained low precision lock (frequency within 1.5% of target).
1—0	ACTVCLK	00	CKI is currently selected system clock.
		01	OSC is currently selected system clock.
		10	PLL is currently selected system clock.
		11	TCK is currently selected system clock.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-20 Power Control (powerc) Register

Bit	15	14	13	12—11	10	9	8
Name	PLLPD	SSOFFD	SEMIDIS	RSVD	SLOWCK	NOCK	RSVD
Reset	1	0	0	0	0	0	0

Bit	7	6	5—4	3	2—1	0
Name	INT0EN	ICPEN	RSVD	SSPDIS	RSVD	TMRDIS
Reset	0	0	0	0	0	0

Bit	Name	Value	Description
15	PLLPD	0	Power up PLL.
		1	Power down (disable) PLL (signal ANDed with PLL power down from *ARM*).
14	SSOFFD	0	Prevent small-signal buffer from being powered down.
		1	Allow small-signal buffer to be powered down.
13	SEMIDIS	0	Enable SEMI clock.
		1	Disable SEMI clock.
12—11	RSVD	—	Reserved.
10	SLOWCK	0	Disable selection of crystal oscillator as DSP system clock.
		1	Enable selection of crystal oscillator as DSP system clock.
9	NOCK	0	Enable system clock (DCLK).
		1	Disable system clock (DCLK).
8	RSVD	—	Reserved.
7	INT0EN	0	Prevent INT0 pin from restarting DSP system clock.
		1	Allow INT0 pin to restart DSP system clock.
6	ICPEN	0	Prevent ICP interrupt from restarting DSP system clock.
		1	Allow ICP interrupt to restart DSP system clock.
5—4	RSVD	—	Reserved.
3	SSPDIS	0	Enable clock to SSP/I^2S.
		1	Disable clock to SSP/I^2S.
2—1	RSVD	—	Reserved.
0	TMRDIS	0	Enable clocks to timer.
		1	Disable clocks to timer.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-21 psw0 (Processor Status Word 0) Register

Bit	15	14	13	12	11	10	9	8—5	4	3—0
Name	LMI	LEQ	LLV	LMV	SLLV	SLMV	a1V	a1[35:32]	a0V	a0[35:32]

Bit	Name	Value	Description	R/W	Reset Value*
15	LMI	0	Most recent DAU result† is not negative.	R/W	X
		1	Most recent DAU result‡ is negative (minus).		
14	LEQ	0	Most recent DAU result‡ is not zero.	R/W	X
		1	Most recent DAU result‡ is zero (equal).		
13	LLV	0	Most recent DAU operation‡ did not result in logical overflow.	R/W	X
		1	Most recent DAU operation‡ resulted in logical overflow.§		
12	LMV	0	Most recent DAU operation did not result in mathematical overflow.	R/W	X
		1	Most recent DAU operation‡ resulted in mathematical overflow.**		
11	SLLV	0	Previous DAU operation did not result in logical overflow.	R/W	0
		1	Sticky version of LLV that remains active once set by a DAU operation until explicitly cleared by a write to **psw0**.		
10	SLMV	0	Previous DAU operation did not result in mathematical overflow.	R/W	0
		1	Sticky version of LMV that remains active once set by a DAU operation until explicitly cleared by a write to **psw0**.		
9	a1V	0	The current contents of **a1** is not mathematically overflowed.	R/W	X
		1	The current contents of **a1** is mathematically overflowed.††		
8—5	a1[35:32]	—	Reflects the four lower guard bits of **a1**.‡‡	R/W	XXXX
4	a0V	0	The current contents of **a0** is not mathematically overflowed.	R/W	X
		1	The current contents of **a0** is mathematically overflowed.††		
3—0	a0[35:32]	—	Reflects the four lower guard bits of **a0**.‡‡	R/W	XXXX

* In this column, X indicates unknown on external reset and unaffected on subsequent reset.
† ALU/ACS result or operation if the instruction uses the ALU/ACS; otherwise, ADDER or BMU result, whichever applies.
‡ ALU/ACS result if the DAU operation uses the ALU/ACS; otherwise, ADDER or BMU result, whichever applies.
§ The ALU or ADDER cannot represent the result in 40 bits or the BMU control operand is out of range.
** The ALU/ACS, ADDER, or BMU cannot represent the result in 32 bits. For the BMU, other conditions can also cause mathematical overflow.
†† The most recent DAU result that was written to that accumulator resulted in mathematical overflow (LMV) with FSAT = 0.
‡‡ Required for compatibility with DSP16XX family.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-22 psw1 (Processor Status Word 1) Register

Bit	15	14	13—12	11—10	9—7	6	5—0
Name	RSVD	IEN	IPLc[1:0]	IPLp[1:0]	RSVD	EPAR	a[7:2]V

Bit	Name	Value	Description	R/W	Reset Value*
15	RSVD	0	Reserved—write with 0.	R/W	0
14	IEN†	0	Hardware interrupts are globally disabled.	R	0
		1	Hardware interrupts are globally enabled.		
13—12	IPLc[1:0]	00	Current hardware interrupt priority level is 0; core handles pending interrupts of priority 1, 2, or 3.	R/W	00
		01	Current hardware interrupt priority level is 1; core handles pending interrupts of priority 2 or 3.		
		10	Current hardware interrupt priority level is 2; core handles pending interrupts of priority 3 only.		
		11	Current hardware interrupt priority level is 3; core does not handle any pending interrupts.		
11—10	IPLp[1:0]	00	Previous hardware interrupt priority level‡ was 0.	R/W	XX
		01	Previous hardware interrupt priority level‡ was 1.		
		10	Previous hardware interrupt priority level‡ was 2.		
		11	Previous hardware interrupt priority level‡ was 3.		
9—7	RSVD	0	Reserved—write with 0.	R/W	X
6	EPAR	0	Most recent BMU or special function shift result has odd parity.	R/W	X
		1	Most recent BMU or special function shift result has even parity.		
5	a7V	0	The current contents of **a7** are not mathematically overflowed.	R/W	X
		1	The current contents of **a7** are mathematically overflowed.§		
4	a6V	0	The current contents of **a6** are not mathematically overflowed.	R/W	X
		1	The current contents of **a6** are mathematically overflowed.§		
3	a5V	0	The current contents of **a5** are not mathematically overflowed.	R/W	X
		1	The current contents of **a5** are mathematically overflowed.§		
2	a4V	0	The current contents of **a4** are not mathematically overflowed.	R/W	X
		1	The current contents of **a4** are mathematically overflowed.§		
1	a3V	0	The current contents of **a3** are not mathematically overflowed.	R/W	X
		1	The current contents of **a3** are mathematically overflowed.§		
0	a2V	0	The current contents of **a2** are not mathematically overflowed.	R/W	X
		1	The current contents of **a2** are mathematically overflowed.§		

* In this column, X indicates unknown on external reset and unaffected on subsequent reset.

† The user clears this bit by executing a **di** instruction and sets it by executing an **ei** or **ireturn** instruction. The core clears this bit whenever it begins to service an interrupt.

‡ Previous interrupt priority level is the priority level of the interrupt most recently serviced prior to the current interrupt. This field is used for interrupt nesting.

§ The most recent DAU result that was written to that accumulator resulted in mathematical overflow (LMV) with FSAT = 0.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-23 BIO Status/Control (sbit) Register

Bit	15—8	7—0
Name	DIREC[7:0]	VALUE[7:0]

Bit	Name	Value	Description
15—8	DIREC[n]*	1xxxxxxx	Reserved.
		x1xxxxxx	Reserved.
		xx1xxxxx	Reserved.
		xxx1xxxx	Reserved.
		xxxx1xxx	Reserved.
		xxxxx1xx	Reserved.
		xxxxxx1x	IOBIT[1] is an output (input when 0).
		xxxxxxx1	IOBIT[0] is an output (input when 0).
7—0	VALUE[n]*	Rxxxxxxx	Reserved.
		xRxxxxxx	Reserved.
		xxRxxxxx	Reserved.
		xxxRxxxx	Reserved.
		xxxxRxxx	Reserved.
		xxxxxRxx	Reserved.
		xxxxxxRx	Reads the current value of IOBIT[1].
		xxxxxxxR	Reads the current value of IOBIT[0].

* 0 = n = 7.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-24 Timer Control (timerc) Register

Bit	15—10	9	8	7	6	5	4	3—0
Name	RSVD	PRDSEL	PSRST	LTC	DISABLE	RELOAD	TOEN	PRESCALE

Bit	Name	Value	Description
15—10	RSVD	—	Reserved—write with 0.
9	PRDSEL	0	Reads content of down counter.
		1	Reads content of period register.
8	PSRST	0	Prescaler is not reset with write to **timer**.
		1	Prescaler is reset with write to **timer**.
7	LTC	0	Loads new value into the counter immediately.
		1	Loads new value into the counter at the end of current count (counter = 0).
6	DISABLE	0	Counter and prescalar enabled.
		1	Counter and prescalar disabled.
5	RELOAD	0	Counter stops after reaching 0.
		1	After reaching zero, counter resumes with value stored in period register.
4	TOEN	0	Counter holds the current count.
		1	Counter starts decrementing.
3—0	PRESCALE	0—15	Provides divided clock CKI/2^{N+1}, N = 0—15, (e.g., CKI/2, CKI/4, CKI/8).

Table 8.15-25 timer (TIMER Running Count) Registers

Bit	15—0
Name	Down Counter
Name	Period Register

Bit	Name*	Description	R/W†	Reset Value‡
15—0	Down Counter	If the COUNT field (**timerc**[4]) is set, TIMER decrements this portion of the **timer** register every prescale period. When the down counter reaches zero, TIMER generates an interrupt.	R/W	0
15—0	Period Register	If the COUNT field (**timerc**[4]) and the RELOAD field (**timerc**[5]) are both set and the down counter contains zero, TIMER reloads the down counter with the contents of this portion of the **timer** register.	R/W	X

* **timer** is a 16 bit counter register. It runs on the CLKR. Contents of **timer** register are copied to period register that runs on CKI. Period register is written with the synchronized version of **timer** write signal. When a read occurs to **timer** register the contents of down counter are read back by default. To read the contents of period register the user should set the PRDSEL bit to 1 in the **timerc** control register.

† To read or write the **timer** register, TIMER must be powered up (i.e., the PWR_DWN field (**timerc**[6]) must be cleared).

‡ For this column, X indicates unknown on external reset and unaffected on subsequent reset.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-26 vsw (Viterbi Support Word) Register

Bit	15—6	5	4	3	2	1	0
Name	RSVD	VEN	MAX	TB2	RSVD	CFLAG1	CFLAG0

Bit	Name	Value	Description	R/W	Reset Value
15—6	RSVD	0	Reserved—write with 0.	R/W	0
5	VEN	0	Disables Viterbi side effects.	R/W	0
		1	Enables Viterbi side effects.		
4	MAX	0	The **cmp0( )**, **cmp1( )**, and **cmp2( )** functions select the minimum value from the input operands.	R/W	0
		1	The **cmp0( )**, **cmp1( )**, and **cmp2( )** functions select the maximum value from the input operands.		
3	TB2	0 (GSM/IS95-compatible mode)	For the single-ACS (40-bit) **cmp1( )** function, the traceback encoder stuffs one traceback bit into **ar0**. For the single-ACS (40-bit) **cmp0( )** function, the traceback encoder stuffs one old traceback bit from **ar0** into **ar1**. For the dual-ACS (16-bit) **cmp1( )** function, the traceback encoder stuffs CFLAG into **ar0** and **ar2**.	R/W	0
		1 (IS54/IS136-compatible mode)	For the single-ACS (40-bit) **cmp1( )** function, the traceback encoder stuffs two traceback bits into **ar0**. For the single-ACS (40-bit) **cmp0( )** function, the traceback encoder stuffs two old traceback bits from **ar0** into **ar1**.		
2	RSVD	0	Reserved—write with 0.	R/W	0
1	CFLAG1	—	Previous value of CFLAG0. The traceback encoder copies the value of CFLAG0 to CFLAG1 if the DAU executes a **cmp2( )** function and VEN=1.	R/W	0
0	CFLAG0	—	Previous value of CFLAG*. The traceback encoder copies the value of CFLAG to CFLAG0 if the DAU executes a **cmp2( )** function and VEN=1.		

* For the **cmp2(aSE,aDE)** function, CFLAG=0 if MAX=0 and aSE=aDE or if MAX=1 and aSE<aDE, and CFLAG=1 if MAX=0 and aSE<aDE or if MAX=1 and aSE=aDE.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

8.15.2 Memory-Mapped Registers

The memory-mapped registers located in their associated peripherals are each mapped to an even address. The sizes of these registers are 16 bits, 20 bits, or 32 bits. A register that is 20 bits or 32 bits must be accessed as an aligned double word. A register that is 16 bits can be accessed as a single word with an even address or as an aligned double word with the same even address. If a register that is 16 bits or 20 bits is accessed as a double word, the contents of the register are right-justified. Memory-mapped registers have the same internal format as other registers and are different from memory. Figure 8.15-2 illustrates three memory-mapped registers.

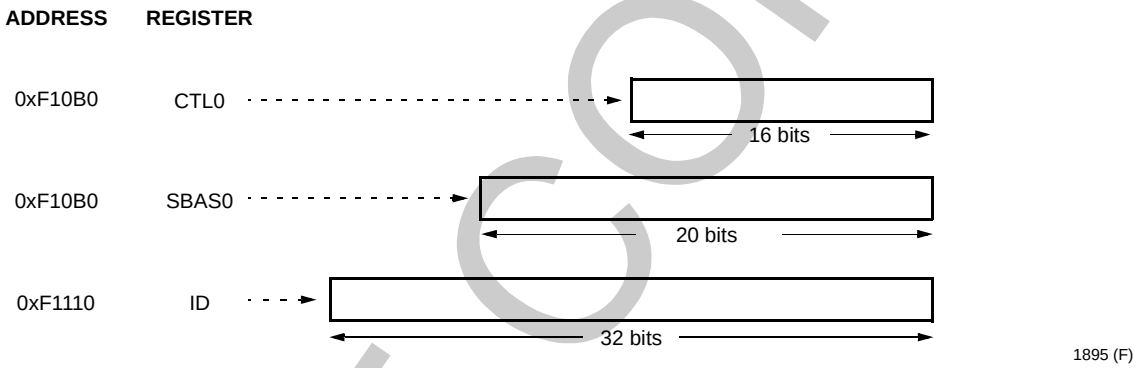


Figure 8.15-2 Example Memory-Mapped Registers

Note: Accessing memory-mapped registers with an odd address yields undefined results. The memory-mapped registers are defined by name and equated to their even memory addresses in the include file that is provided with the *LUxWORKS* tools.

Memory-mapped registers are designated with upper-case letters. See Table 8.15-27 for list of memory-mapped registers.

Table 8.15-27 DSP Block Memory-Mapped Register Table

Name	Address	Description	Bits	R/W	Type	Reset Value	Table #
System External Memory Interface (SEMI)							
ECON0	0xF0000	SEMI control.	16	R/W	Control	0x0FFF	8.15-28
DSP-Side SSP/I²S (SSP1)							
SSPCR0	0xF3000	Control register 0.	16	RW	Control	0x0	8.15-30
SSPCR1	0xF3002	Control register 1.	16	RW	Control	0x0	8.15-31
SSPDR	0xF3004	Data register.	16	RW	Data	Unknown	8.15-32
SSPSR	0xF3006	Status register.	16	R	Status	0x3	8.15-37
SSPCPSR	0xF3008	Clock prescale register.	16	RW	Control	0x0	8.15-29
SSPIMSC	0xF300A	Interrupt mask set or clear register.	16	RW	Control	0x0	8.15-34
SSPRIS	0xF300C	Raw interrupt status register.	16	R	Status	0x8	8.15-36
SSPMIS	0xF300E	Masked interrupt status register.	16	R	Status	0x0	8.15-35
SSPICR	0xF3010	Interrupt clear register.	16	W	Control	0x0	8.15-33

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-28 through Table 8.15-37 list register encodings for memory-mapped registers.

Table 8.15-28 ECON0 (External Control 0) Register, Address (0xF0000)

Bit	15	14	13	12	11—8	7—0
Name	WHOLD	RHOLD	WSETUP	RSETUP	IATIME[3:0]	RSVD

Bit	Name	Value	Description	R/W	Reset Value
15	WHOLD	0	The SEMI does not extend the write cycle.	R/W	0
		1	The SEMI extends the write cycle for one CLK cycle, applies the target address, deasserts all enables, deasserts all write strobes, and 3-states D_D[15:0].		
14	RHOLD	0	The SEMI does not extend the read cycle.	R/W	0
		1	The SEMI extends the read cycle for one CLK cycle, applies the target address, and deasserts all enables.		
13	WSETUP	0	The SEMI does not delay the assertion of the write strobe, the memory enable, and the assertion of D_D[15:0] for write operations.	R/W	0
		1	The SEMI delays the assertion of the write strobe, the memory enable, and D_D[15:0] during a write cycle for one CLK cycle. During the setup time, the SEMI applies the target address to D_A[8:0], deasserts all enables and RWN signals, and 3-states D_D[15:0].		
12	RSETUP	0	The SEMI does not delay the assertion of the memory enable for read operations.	R/W	0
		1	The SEMI delays the assertion of the memory enable during a read cycle for one CLK cycle. During the setup time, the SEMI applies the target address to D_A[8:0], deasserts all enables and the RWN signal, and 3-states D_D[15:0].		
11—8	IATIME[3:0]	0—15	The duration in CLK cycles (1—15) that the SEMI asserts I/O for an asynchronous access to the EIO component. A value of 0 or 1 corresponds to a 1 CLK cycle assertion time.	R/W	0xF
7—0	RSVD	0—15	Reserved.	R/W	0xFF

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-29 Clock Prescale Register (SSPCPSR), Address (SSP_BASE_ADDR + 0x10)

Bit	15—8	7—0
Name	RSVD	CPSDVSR

Bit	Name	Type	Function
15—8	RSVD	—	Reserved. Unpredictable results on reads, must be written as 0.
7—0	CPSDVSR	Read/write	Clock prescale divisor. Must be an even number from 2 to 254, depending on the frequency of SPCLK1. The least significant bit always returns 0 on reads.

Table 8.15-30 Control Register 0 (SSPCR0), Address (SSP_BASE_ADDR + 0x00)

Bit	15—8	7	6	5—4	3—0
Name	SCR	SPH	SPO	FRF	DSS

Bit	Name	Type	Function
15—8	SCR	Read/write	Serial clock rate. The value SCR is used to generate the transmit and receive bit rate of the SSPI²S. The bit rate is: $\frac{F_{DCLK}}{CPSDVR \times (1 + SCR)}$ where CPSDVSR is an even value from 2 to 254, programmed through the SSPCPSR register, and SCR is a value from 0 to 255. DCLK is DSP system clock.
7	SPH	Read/write	SPCLK1 output phase (applicable to *Motorola* SPI frame format only).
6	SPO	Read/write	SPCLK1 output polarity (applicable to *Motorola* SPI frame format only).
5—4	FRF	Read/write	Frame format: 00 *Motorola* SPI frame format. 01 *Texas Instruments* synchronous serial frame format. 10 *National MICROWIRE* frame format. 11 I²S serial bus format.
3—0	DSS	Read/write	Data size select: 0000 Reserved, undefined operation. 0001 Reserved, undefined operation. 0010 Reserved, undefined operation. 0011 4-bit data. 0100 5-bit data. 0101 6-bit data. 0110 7-bit data. 0111 8-bit data. 1000 9-bit data. 1001 10-bit data. 1010 11-bit data. 1011 12-bit data. 1100 13-bit data. 1101 14-bit data. 1110 15-bit data. 1111 16-bit data.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-31 Control Register 1 (SSPCR1), Address (SSP_BASE_ADDR + 0x04)

Bit	15—10	9	8	7	6	5	4	3	2	1	0
Name	RSVD	nCLKIN	nCLKOUT	DS	I2STX	I2STP	I2SRP	SOD	MS	SSE	LBM

Bit	Name	Type	Function
15—10	RSVD	Read	Reserved. Unpredictable results on reads.
9	nCLKIN	Read/Write	When set to 0, no inversion of input clock on SPCLK1(default). When set to 1, inverts incoming clock on SPCLK1.
8	nCLKOUT	Read/write	When set to 0, no inversion of output clock on SPCLK1 (default). When set to 1, inverts outgoing clock on SPCLK1.
7	DS	Read/Write	Disable dynamic master/slave switching. When set to 1, dynamic master/slave switching is off. When set to 0, dynamic master/slave switching is on (default).
6	I2STX	Read/write	This bit works in conjunction with the MS bit field (see bit 2). When set to 1, I²S is in transit mode. When set to 0, I²S is in receive mode (default).
5	I2STP	Read/write	This bit applies to the I²S transmitter in master mode. When set to 0 (default), the word select pin (SPFS pin) is low for all odd numbered transmissions, and high for all even numbered transmissions. When set to 1, the polarity of the word select pin is inverted. Toggling the SSE bit has no effect on the polarity of the word select pin, since the status of this pin is maintained for I²S mode whether this block is disabled or switched into other serial formats. In order to achieve left/right channel synchronization, the software can track the total number of words transmitted in I²S mode since reset, and adjust the I2STP bit accordingly. It is recommended that the software make sure an even number of words are written to TX FIFO during each session of I²S transmission before pausing it or switching to other serial transmission modes.
4	I2SRP	Read/write	During receive, this bit is set to 0 if the left channel word is received first (default). This bit is set to 1 if the first word received is for the right channel. This bit is valid only after at least one word is received by the SSPI²S block.
3	SOD	Read/write	Slave-mode output disable. This bit is relevant only in the slave mode (MS = 1). In multiple-slave systems, it is possible for an SSP master to broadcast a message to all slaves in the system while ensuring that only one slave drives data onto its serial output line. In such systems, the RXD lines from multiple slaves could be tied together. To operate in such systems, the SOD bit can be set if the SSP slave is not supposed to drive the SSPTXD line. 0 = SSP can drive the SPTXD1_I2SD output in slave mode (default). 1 = SSP must not drive the SPTXD1_I2SD output in slave mode.
2	MS	Read/write	Master or slave mode select. This bit can be modified only when the SSPI²S is disabled (SSE=0): 0 = device configured as master (default). 1 = device configured as slave.
1	SSE	Read/write	Synchronous serial port enable. 0 = SSPI²S operation disabled (default). 1 = SSPI²S operation enabled.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-31 Control Register 1 (SSPCR1), Address (SSP_BASE_ADDR + 0x04) (continued)

Bit	Name	Type	Function
0	LBM	Read/write	Loopback mode. 0 = Normal serial port operation enabled (default). 1 = Output of transmit serial shifter is connected to input of receive serial shifter internally.

Table 8.15-32 Data Register (SSPDR), Address (SSP_BASE_ADDR + 0x08)

Bit	15—0
Name	DATA

Bit	Name	Type	Function
15—0	DATA	Read/write	Transmit/receive FIFO: Read = Receive FIFO. Write = Transmit FIFO. Data must be right-justified when the SSPI^{2}S is programmed for a data size that is less than 16 bits. Unused bits at the top are ignored by transmit logic. The receive logic automatically right-justifies.

Table 8.15-33 Interrupt Clear Register (SSPICR), Address (SSP_BASE_ADDR + 0x020)

Bit	15—2	1	0
Name	RSVD	RTIC	RORIC

Bit	Name	Type	Function
15—2	RSVD	—	Reserved. Read as zero, do not modify.
1	RTIC	Write	Clears the SSPRTINTR interrupt.
0	RORIC	Write	Clears the SSPRORINTR interrupt.

Table 8.15-34 Interrupt Mask Register (SSPIMSC), Clear/Set Address (SSP_BASE_ADDR + 0x14)

Bit	15—4	3	2	1	0
Name	RSVD	TxIM	RxIM	RTIM	RORIM

Bit	Name	Type	Function
15—4	RSVD	—	Reserved. Read as zero, do not modify.
3	TxIM	Read/write	Transmit FIFO interrupt mask: 0 = Tx FIFO half empty or less condition interrupt is masked. 1 = Tx FIFO half empty or less condition interrupt is not masked.
2	RxIM	Read/write	Receive FIFO interrupt mask: 0 = Rx FIFO half full or less condition interrupt is masked. 1 = Rx FIFO half full or less condition interrupt is not masked.
1	RTIM	Read/write	Receive time-out interrupt mask: 0 = RxFIFO not empty and no read prior to time-out period interrupt is masked. 1 = RxFIFO not empty and no read prior to time-out period interrupt is not masked.
0	RORIM	Read/write	Receive overrun interrupt mask: 0 = RxFIFO written to while full condition interrupt is masked. 1 = RxFIFO written to while full condition interrupt is not masked.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-35 Masked Interrupt Status Register (SSPMIS), Address (SSP_BASE_ADDR + 0x1C)

Bit	15—4	3	2	1	0
Name	RSVD	TxMIS	RxMIS	RTMIS	RORMIS

Bit	Name	Type	Function
15—4	RSVD	—	Reserved. Read as zero, do not modify.
3	TxMIS	Read	Gives the transmit FIFO masked interrupt state (after masking) of the SSPTXINTR interrupt.
2	RxMIS	Read	Gives the receive FIFO masked interrupt state (after masking) of the SSPRXINTR interrupt.
1	RTMIS	Read	Gives the receive time-out masked interrupt state (after masking) of the SSPRTINTR interrupt.
0	RORMIS	Read	Gives the receive over run masked interrupt status (after masking) of the SSPRORINTR interrupt.

Table 8.15-36 Raw Interrupt Status Register (SSPRIS), Address (SSP_BASE_ADDR + 0x18)

Bit	15—4	3	2	1	0
Name	RSVD	TxRIS	RxRIS	RTRIS	RORRIS

Bit	Name	Type	Function
15—4	RSVD	—	Reserved. Read as zero, do not modify.
3	TxRIS	Read	Gives the raw interrupt state (prior to masking) of the SSPTXINTR interrupt.
2	RxRIS	Read	Gives the raw interrupt state (prior to masking) of the SSPRXINTR interrupt.
1	RTRIS	Read	Gives the raw interrupt state (prior to masking) of the SSPRTINTR interrupt.
0	RORRIS	Read	Gives the raw interrupt state (prior to masking) of the SSPRORINTR interrupt.

Table 8.15-37 Status Register (SSPSR), Address (SSP_BASE_ADDR + 0x0C)

Bit	15—5	4	3	2	1	0
Name	RSVD	BSY	RFF	RNE	TNF	TFE

Bit	Name	Type	Function
15—5	RSVD	—	Reserved. Unpredictable results on reads, should be written as 0.
4	BSY	Read	SSPI^{2}S busy flag (read-only): 0 = SSPI^{2}S is idle. 1 = SSPI^{2}S is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.
3	RFF	Read	Receive FIFO full (read-only): 0 = Receive FIFO is not full. 1 = Receive FIFO is full.
2	RNE	Read	Receive FIFO not empty (read-only): 0 = Receive FIFO is empty. 1 = Receive FIFO is not empty.
1	TNF	Read	Transmit FIFO not full (read-only): 0 = Transmit FIFO is full. 1 = Transmit FIFO is not full.
0	TFE	Read	Transmit FIFO empty (read-only): 0 = Transmit FIFO is not empty. 1 = Transmit FIFO is empty.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

8.15.3 Reset States

Pin reset occurs if a high-to-low transition is applied to the RSTN pin. Table 8.15-38—Table 8.15-42 show how reset affects the core and off-core registers. The following bit codes apply:

- Bit code • indicates that this bit is unknown on external reset and unaffected on a subsequent pin reset.
- Bit code P indicates the value on the corresponding input pin.

Table 8.15-38 Core Register States After Reset—40-Bit Registers

Register	Bits[39:0]
a0	•••• •••• •••• •••• •••• •••• •••• •••• •••• ••••
a1	•••• •••• •••• •••• •••• •••• •••• •••• •••• ••••
a2	•••• •••• •••• •••• •••• •••• •••• •••• •••• ••••
a3	•••• •••• •••• •••• •••• •••• •••• •••• •••• ••••
a4	•••• •••• •••• •••• •••• •••• •••• •••• •••• ••••
a5	•••• •••• •••• •••• •••• •••• •••• •••• •••• ••••
a6	•••• •••• •••• •••• •••• •••• •••• •••• •••• ••••
a7	•••• •••• •••• •••• •••• •••• •••• •••• •••• ••••

Table 8.15-39 Core Register States After Reset—32-Bit Registers

Register	Bits[31:0]
csave	•••• •••• •••• •••• •••• •••• •••• ••••
p0	•••• •••• •••• •••• •••• •••• •••• ••••
p1	•••• •••• •••• •••• •••• •••• •••• ••••
x	•••• •••• •••• •••• •••• •••• •••• ••••
y	•••• •••• •••• •••• •••• •••• •••• ••••

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-40 Core Register States After Reset—20-Bit Registers

Register	Bits[19:0]	Register	Bits[19:0]
h	•••• •••• •••• •••• ••••	r1	•••• •••• •••• •••• ••••
i	•••• •••• •••• •••• ••••	r2	•••• •••• •••• •••• ••••
inc0	0000 0000 0000 0000 0000	r3	•••• •••• •••• •••• ••••
inc1	0000 0000 0000 0000 0000	r4	•••• •••• •••• •••• ••••
ins	0000 0000 0000 0000 0000	r5	•••• •••• •••• •••• ••••
j	•••• •••• •••• •••• ••••	r6	•••• •••• •••• •••• ••••
k	•••• •••• •••• •••• ••••	r7	•••• •••• •••• •••• ••••
PC*	XXXX 0000 0000 0000 0000	rb0	0000 0000 0000 0000 0000
pi	•••• •••• •••• •••• ••••	rb1	0000 0000 0000 0000 0000
pr	•••• •••• •••• •••• ••••	re0	0000 0000 0000 0000 0000
pt0	•••• •••• •••• •••• ••••	re1	0000 0000 0000 0000 0000
pt1	•••• •••• •••• •••• ••••	sp	•••• •••• •••• •••• ••••
ptrap	•••• •••• •••• •••• ••••	vbase	0010 0000 0000 0001 0100
r0	•••• •••• •••• •••• ••••		

* **PC** resets to 0x20000 (first address of IROM) if the EXM pin is 0 at the time of reset. It resets to 0x80000 (first address of EROM) if the EXM pin is 1 at the time of reset.

Table 8.15-41 Core Register States After Reset—16-Bit Registers

Register	Bits[15:0]	Register	Bits[15:0]
alf	0000 00•• •000 0000	c1	•••• •••• •••• ••••
ar0	•••• •••• •••• ••••	c2	•••• •••• •••• ••••
ar1	•••• •••• •••• ••••	cloop	0000 0000 0000 0000
ar2	•••• •••• •••• ••••	cstate	0000 0000 0000 0000
ar3	•••• •••• •••• ••••	psw0	•••• 00•• •••• ••••
auc0	0000 0000 0000 0000	psw1	0000 •••• •••• ••••
auc1	0000 0000 0000 0000	vsw	0000 0000 0000 0000
c0	•••• •••• •••• ••••		

Table 8.15-42 Off-Core (Peripheral) Register Reset Values

Register	Bits[15:0]	Register	Bits[15:0]
accon	0000 0000 0000 0000	pllc	0001 1110 0000 0000
acstat	0000 0000 0000 0000	pllsac	0000 0000 0000 0000
ahcon	0000 0000 0000 0000	powerc	1000 0000 0000 0000
ahstat	0000 0000 0000 0000	sbit	0000 0000 •••• ••PP
cbit	•••• •••• 0000 0000	timer	0000 0000 0000 0000
ioc	0000 0000 0000 0000	timerc	0000 0000 0000 0000
jiob*	•••• •••• •••• •••• •••• •••• •••• ••••		
atest*	0000 0000 0000 0000 0000 0000 0000 0000		
patchc*	0000 0••• •••• 0000 0000 0000 0000 0000		

* The **jiob**, **patchc** and **atest** registers are the only peripheral registers that are 32 bits; therefore, the bit pattern shown is for bits 31—0.

8 Digital Signal Processor (DSP) Block (continued)

8.15 Registers (continued)

Table 8.15-43 Memory-Mapped Register Reset Values—16-Bit Registers

Register	Bits[15:0]
SSPCR0	0000 0000 0000 0000
SSPCR1	0000 0000 0000 0000
SSPDR	•••• •••• •••• ••••
SSPSR	0000 0000 0000 0011
SSPCPSR	0000 0000 0000 0000
SSPIMSC	0000 0000 0000 0000
SSPRIS	0000 0000 0000 1000
SSPMIS	0000 0000 0000 0000
SSPICR	0000 0000 0000 0000
ECON0	0000 1111 1111 1111

8.15.4 RB Field Encoding

Table 8.15-44 describes the encoding of the RB field. This information supplements the instruction set encoding information in the *DSP16000 Digital Signal Processor Core Instruction Set* Reference Manual.

Table 8.15-44 RB Field

RB*	Register	RB*	Register	RB*	Register	RB*	Register
000000	**a0g**	010000	Reserved	100000	**ioc**	110000	Reserved
000001	**a1g**	010001	**cloop**	100001	**powerc**	110001	**accon**
000010	**a2g**	010010	**cstate**	100010	**pllc**	110010	**acstat**
000011	**a3g**	010011	**csave**	100011	**pllsac**	110011	**ahcon**
000100	**a4g**	010100	**auc1**	100100	Reserved	110100	**ahstat**
000101	**a5g**	010101	**ptrap**	100101	**cbit**	110101	Reserved
000110	**a6g**	010110	**vsw**	100110	**sbit**	110110	Reserved
000111	**a7g**	010111	Reserved	100111	**timerc**	110111	**wpstatus**
001000	**a0_1h**	011000	**ar0**	101000	**timer**	111000	Reserved
001001	**inc1**	011001	**ar1**	101001	Reserved	111001	Reserved
001010	**a2_3h**	011010	**ar2**	101010	Reserved	111010	Reserved
001011	**inc0**	011011	**ar3**	101011	Reserved	111011	**patchc**
001100	**a4_5h**	011100	**vbase**	101100	Reserved	111100	Reserved
001101	**pi**	011101	**ins**	101101	Reserved	111101	**atest**
001110	**a6_7h**	011110	Reserved	101110	Reserved	111110	Reserved
001111	**psw1**	011111	Reserved	101111	Reserved	111111	**jiob**

* RB field specifies one of a secondary set of registers as the destination of a data move. Codes 000000 through 011111 correspond to core registers and codes 100000 through 111111 correspond to off-core (peripheral) registers.

9 ICP/IDP

9.1 Interprocessor Communication Port (ICP)

The ICP enables communication between the *ARM* and DSP16000 cores by means of a shared, dual-port 512 x 32-bit memory array and an interrupt-based communication protocol. This is a true dual-port array; accesses are made concurrently by both cores without incurring any additional wait-states. Interprocessor communication is controlled via two pairs of registers, one pair accessible by the DSP16000 core and the other pair accessible by the *ARM* core. Each pair of registers consists of a status and a control register. Specifically, the DSP16000 core reads and writes the ACCON register (see Table 9.1-3) to convey signaling information to the *ARM* core, and reads and writes the ACSTAT register (see Table 9.1-4) to monitor and acknowledge the signals from the *ARM* core.

Similarly, the *ARM* core reads and writes the DCCON register (see Table 9.1-1) to convey signaling information to the DSP16000 core, and reads and writes the DCSTAT register (see Table 9.1-2) to monitor and acknowledge the signals from the DSP16000 core.

Figure 9.1-1 shows the basic configuration of the ICP blocks.

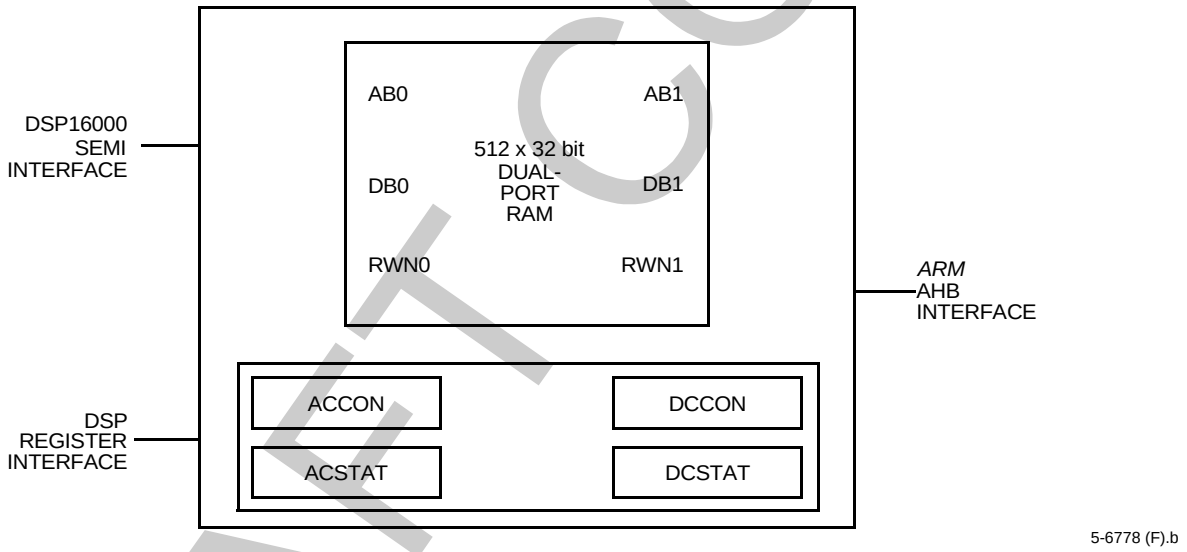


Figure 9.1-1 ICP Block

9.1.1 ICP Architecture

The 512 x 32-bit dual-port memory is accessible by the DSP16000 core through its SEMI system bus. It is mapped into address 0xF7000 through 0xF73FF. Accesses from the DSP core must be 16-bit or 32-bit aligned accesses; byte accesses are not supported.

9 ICP/IDP (continued)

9.1 Interprocessor Communication Port (ICP) (continued)

The 512 x 32-bit dual-port memory is also accessible by the *ARM* core through its AHB bus. The *ARM* core accesses this memory at locations 0xFC000000 through 0xFC0007FF. Accesses from the *ARM* core must be 16-bit or 32-bit aligned accesses; byte accesses are not supported.

Note: Due to the different endian-ness assumed by the *ARM* core and DSP core in memory organization, care must be taken when sharing data between the *ARM* and DSP cores. The *ARM* core assumes little endian, while the DSP core assumes big endian. Therefore, extra software work is needed to switch the high 16-bit word and the low 16-bit word, unless memory accesses from both sides are 32-bit aligned accesses.

The 512 x 32-bit dual-port memory is a true dual-port memory, and no additional wait-states are incurred due to access collisions if both processors access the memory at the same time.

WARNING: Software must prevent memory collisions.

The DCCON and DCSTAT registers within the ICP are accessible through the AHB of the *ARM* core. DCCON is mapped to address 0XFFFEFFF0, and DCSTAT is mapped to address 0XFFFEFFF4.

The DCCON and DCSTAT registers are accessed as 16-bit aligned values. The DCCON registers allow the *ARM* core to convey signaling information to the DSP16000 core. The DCSTAT registers allow the *ARM* core to monitor and acknowledge various signals from the DSP16000 core. The functions of the bits in the DCCON register are listed in Table 9.1-1 and the states indicated by the bits in DCSTAT are listed in Table 9.1-2.

Similar to the DCCON and DCSTAT registers, the ICP also has two 16-bit registers, ACCON and ACSTAT, that are accessible by the DSP16000 through its external register interface. The ACCON and ACSTAT registers are mapped into the peripheral register space of the DSP16000 core and have the following RAB-field register IDs: ACCON—0x71 and ACSTAT—0x72. The control register ACCON allows the DSP16000 core to convey signaling information to the *ARM* core. The status register ACSTAT allows the DSP16000 core to monitor and acknowledge various signals from the *ARM* core. The functions of the bits in the ACCON register are listed in Table 9.1-3 and the states indicated by the bits in ACSTAT are listed in Table 9.1-4.

The ICP state is initialized by T8307 external active-low RESETN pin reset.

Bits[7:0] of the DCCON registers allow the *ARM* core to interrupt the DSP16000 core via its INTn interrupt input and specify one (or more) of eight independently handshaked operations. If any of the DIRQ[7:0] bits are set by the *ARM* core, the ICP interrupt line to the DSP16000 core is asserted. The values of these bits are also synchronized with the DSP16000 core and reflected in bits 0—7 of the ACSTAT register, and the INTn interrupt handler in the DSP16000 core reads these bits to determine the desired operation(s). Once an operation is completed, the DSP16000 core signals completion by writing a one to the appropriate bit(s) of the DIRQ[0:7] field of the ACSTAT register, which clears the request in both the DCCON and ACSTAT registers. The *ARM* core verifies completion of an operation by monitoring the status of the DIRQ[0:7] field in the DCCON register.

Since all 8 bits share one vectored interrupt to the DSP16000, it is up to the INTn interrupt handler software to determine which actual operation is desired by examining the DIRQ[0:7] field of the ACSTAT register.

Similarly, bits 0—7 of the ACCON register allow the DSP16000 core to interrupt the *ARM* core via its IRQ27 interrupt input and specify one (or more) of eight independently handshaken operations. If any of the AIRQ[0:7] bits are set by the DSP16000 core, the IRQ27 line to the *ARM* core is asserted. The values of these bits are also reflected in bits 0—7 of the DCSTAT register, and the interrupt handler in the *ARM* core reads these bits to determine the desired operations. Handshaking and acknowledgment proceed in the same manner as described previously for the DSP16000 core.

The upper 6 bits of the DCCON register (bits 10—15) contain handshake bits that are writable by the *ARM* core and readable by the DSP16000 core via the corresponding bits of the ACSTAT register. Similarly, the upper 6 bits of the ACCON register (bits 10—15) contain handshake bits that are writable by the DSP16000 core and readable by the *ARM* core via the corresponding bits of the DCSTAT register. These bits are available to the user to aid in interprocessor communication and synchronization.

The DCCON registers also contain one additional control bit. The DRESETN signal (bit 8) of DCCON allows the *ARM* core to force a reset of the entire DSP subsystem. Note that this bit is negative-assertion, so the DSP comes up by default in the reset state. When written to a one, the DSP16000 core and its peripherals begin execution at the boot address. If written back to a zero, the DSP subsystem reenters the reset state.

9 ICP/IDP (continued)

9.1 Interprocessor Communication Port (ICP) (continued)

Table 9.1-1 DCCON Control Register in ICP (Controlled by *ARM* Core)

Bit	15	14	13	12	11	10	9	8	7—0
Name	ACSIG5	ACSIG4	ACSIG3	ACSIG2	ACSIG1	ACSIG0	RSVD	DRESETN	DIRQ[0:7]

Bit	Name	Description*
15	ACSIG5	User handshake signal 5 from *ARM* core to the DSP16000 core—readable by the DSP16000 core via ACSTAT register bit 15.
14	ACSIG4	User handshake signal 4 from *ARM* core to the DSP16000 core—readable by the DSP16000 core via ACSTAT register bit 14.
13	ACSIG3	User handshake signal 3 from *ARM* core to the DSP16000 core—readable by the DSP16000 core via ACSTAT register bit 13.
12	ACSIG2	User handshake signal 2 from *ARM* core to the DSP16000 core—readable by the DSP16000 core via ACSTAT register bit 12.
11	ACSIG1	User handshake signal 1 from *ARM* core to the DSP16000 core—readable by the DSP16000 core via ACSTAT register bit 11.
10	ACSIG0	User handshake signal 0 from *ARM* core to the DSP16000 core—readable by the DSP16000 core via ACSTAT register bit 10.
9	RSVD	Reserved.
8	DRESETN	When zero, the DDSP16000 core and its associated peripherals are held in the reset state (combined with external RESET pin). When written to a one by the *ARM* core, the DSP16000 begins execution.
7—0	DIRQ[0:7]	Interrupt request 0 through 7 to the DSP16000 core—readable and clearable by the DSP16000 core via ACSTAT register bits 0—7.

* All reset values are 0.

Table 9.1-2 DCSTAT Status Register in ICP (Controlled by *ARM* Core)

Bit	15	14	13	12	11	10	9—8	7—0
Name	DCSIG5	DCSIG4	DCSIG3	DCSIG2	DCSIG1	DCSIG0	RSVD	AIRQ[0:7]

Bit	Name	Description*
15	DCSIG5	User handshake signal 5 from the DSP16000 core to *ARM* core—reflects value in bit 15 of ACCON register.
14	DCSIG4	User handshake signal 4 from the DSP16000 core to *ARM* core—reflects value in bit 14 of ACCON register.
13	DCSIG3	User handshake signal 3 from the DSP16000 core to *ARM* core—reflects value in bit 13 of ACCON register.
12	DCSIG2	User handshake signal 2 from the DSP16000 core to *ARM* core—reflects value in bit 12 of ACCON register.
11	DCSIG1	User handshake signal 1 from the DSP16000 core to *ARM* core—reflects value in bit 11 of ACCON register.
10	DCSIG0	User handshake signal 0 from the DSP16000 core to *ARM* core—reflects value in bit 10 of ACCON register.
9—8	RSVD	Reserved.
7—0	AIRQ[0:7]	Interrupt request 0 through 7 to *ARM* core from the DSP16000 core. Writing a one to each bit acknowledges the interrupt, clearing AIRQ[n] in both ACCON and DCSTAT registers. Writing a zero to each bit leaves the value unchanged.

* All reset values are 0.

9 ICP/IDP (continued)

9.1 Interprocessor Communication Port (ICP) (continued)

Table 9.1-3 ACCON Control Register in ICP (Controlled by the DSP16000 Core)

Bit	15	14	13	12	11	10	9—8	7—0
Name	DCSIG5	DCSIG4	DCSIG3	DCSIG2	DCSIG1	DCSIG0	RSVD	AIRQ[0:7]

Bit	Name	Description*
15	DCSIG5	User handshake signal 5 from the DSP16000 core to *ARM* core—readable by *ARM* core via DCSTAT register bit 15.
14	DCSIG4	User handshake signal 4 from the DSP16000 core to *ARM* core—readable by *ARM* core via DCSTAT register bit 14.
13	DCSIG3	User handshake signal 3 from the DSP16000 core to *ARM* core—readable by *ARM* core via DCSTAT register bit 13.
12	DCSIG2	User handshake signal 2 from the DSP16000 core to *ARM* core—readable by *ARM* core via DCSTAT register bit 12.
11	DCSIG1	User handshake signal 1 from the DSP16000 core to *ARM* core—readable by *ARM* core via DCSTAT register bit 11.
10	DCSIG0	User handshake signal 0 from the DSP16000 core to *ARM* core—readable by *ARM* core via DCSTAT register bit 10.
9—8	RSVD	Reserved.
7—0	AIRQ[0:7]	Interrupt request 0 through 7 to *ARM* core—readable and clearable by *ARM* core via DCSTAT register bits 0—7.

* All reset values are 0.

Table 9.1-4 ACSTAT Status Register in ICP (Controlled by the DSP16000 Core)

Bit	15	14	13	12	11	10	9—8	7—0
Name	ACSIG5	ACSIG4	ACSIG3	ACSIG2	ACSIG1	ACSIG0	RSVD	DIRQ[0:7]

Bit	Name	Description*
15	ACSIG5	User handshake signal 5 from *ARM* core to the DSP16000 core—reflects value in bit 15 of DCCON register.
14	ACSIG4	User handshake signal 4 from *ARM* core to the DSP16000 core—reflects value in bit 14 of DCCON register.
13	ACSIG3	User handshake signal 3 from *ARM* core to the DSP16000 core—reflects value in bit 13 of DCCON register.
12	ACSIG2	User handshake signal 2 from *ARM* core to the DSP16000 core—reflects value in bit 12 of DCCON register.
11	ACSIG1	User handshake signal 1 from *ARM* core to the DSP16000 core—reflects value in bit 11 of DCCON register.
10	ACSIG0	User handshake signal 0 from *ARM* core to the DSP16000 core—reflects value in bit 10 of DCCON register.
9—8	RSVD	Reserved.
7—0	DIRQ[0:7]	Interrupt request 0 through 7 to the DSP16000 core from *ARM* core. Writing a one to each bit acknowledges the interrupt, clearing DIRQ[n] in both DCCON and ACSTAT registers. Writing a zero to each bit leaves the value unchanged.

* All reset values are 0.

9 ICP/IDP (continued)

9.2 Interprocessor Debug Port (IDP)

The IDP block implement interprocessor breakpointing and debugging features in T8307. This module contains two pairs of registers, one pair accessible by the DSP16000 core and the other pair accessible by the *ARM* core. Each pair of registers consists of a status and a control register. Specifically, the DSP core reads and writes the AHCON register to enable specific debugging options in the *ARM* core, and can read the AHSTAT register to monitor the debugging status of the *ARM* core.

Similarly, the *ARM* core reads and writes the corresponding DHCON register to enable specific debugging options in the DSP core, and reads the DHSTAT registers to monitor the debugging status of DSP core. Figure 9.2-1 shows the basic configuration of the IDP block.

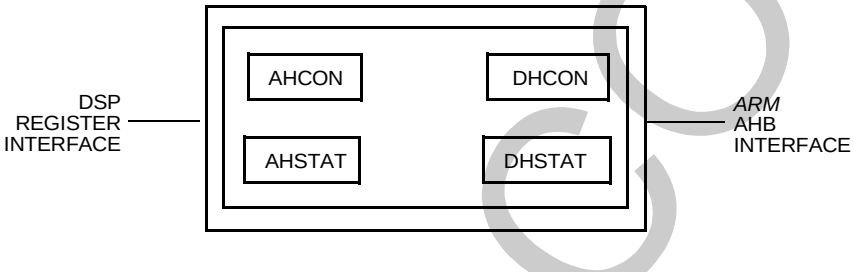


Figure 9.2-1 IDP Block

9.2.1 IDP Architecture

The DHCON and DHSTAT registers within the IDP are accessible through the AHB of the *ARM* core. DHCON is mapped to address 0xFFFEFFF8, and DHSTAT is mapped to address 0xFFFEFFFC. DHCON and DHSTAT are accessed as 16-bit aligned values. The control register DHCON allows the *ARM* core to enable and configure various debug options in the DSP16000. The status register DHSTAT contains information about the debug state of the DSP16000 core, accessible by the *ARM* core. The functions of the bits in the DHCON register and the states indicated by the bits in DHSTAT are listed in Table 9.2-2.

Similar to the DHCON and DHSTAT registers, the IDP also has two 16-bit registers, AHCON and AHSTAT, that are accessible by the DSP16000 through its external register interface. The AHCON and AHSTAT registers are mapped into the peripheral register space of the DSP16000 and have the following RAB-field register IDs: AHCON—0x73 and AHSTAT—0x74. The bits of AHCON are set or reset by the DSP16000 core, and enable and configure various debug options in the *ARM* core. The bits of AHSTAT reflect the debug state of the *ARM* core so that they can be monitored by the DSP16000 core. The functions of the bits in the AHCON register are listed in Table 9.2-3, and the states indicated by the bits in AHSTAT are listed in Table 9.2-4.

9 ICP/IDP (continued)

9.2 Interprocessor Debug Port (IDP) (continued)

The IDP state is initialized by T8307 external active-low RESETN pin.

Each of the lower 8 bits of the AHCON and DHCON registers enable a specific debugging feature in either the *ARM* core or the DSP16000 core. These bits in the two registers are matched; the DSP16000 core and the *ARM* core have their corresponding bits set in order to enable the feature.

Bits[1:0] of AHCON/DHCON allow a breakpoint in one processor to also force a breakpoint in the other processor. These bits enable simultaneous halting of execution on the occurrence of a breakpoint in either processor. Once halted, the diagnostic routine resident in the memories of both processors gain control, thus allowing concurrent monitoring of their states. This arrangement is especially useful during debugging of the control code that handles interprocessor communication. In this mode, it is not possible to synchronously resume execution in both processors once halted, since the *ARM* and the DSP16000 cores reside on separate JTAG scan chains.

Bits[3:2] of AHCON/DHCON allow either processor to immediately force a breakpoint in the other. This allows either processor to force the other to halt execution and enter debug mode. Figure 9.2-2 illustrates the behavior of bits[3:0] of the AHCON and DHCON registers.

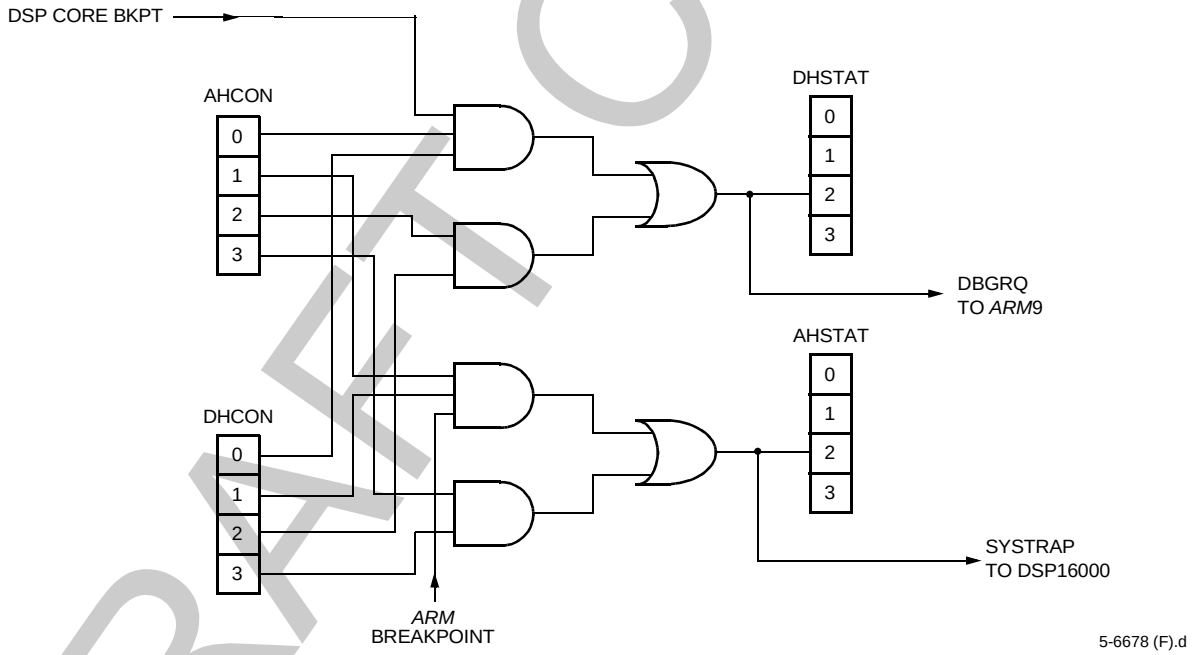


Figure 9.2-2 Behavior of AHCON and DHCON Bits[3:0]

Bits[6:4] of AHCON/DHCON allow either processor to temporarily stop execution of the other processor. The *ARM* core is stopped by stopping its clock. Any timers controlled by the *ARM* core stop at their current values while the stop request is in effect. The *ARM* debugging tools cannot access any *ARM*-side modules, including the *ARM* core, during this period.

The DSP16000 core is stopped by freezing the clocks to the core. All operations stop until the stop request is cleared. Any timers controlled by the DSP16000 core stop at their current values while the stop request is in effect. The DSP16000 debugging tools are locked out during this period.

9 ICP/IDP (continued)

9.2 Interprocessor Debug Port (IDP) (continued)

Bit 4 of AHCON/DHCON allows the DSP16000 core to immediately stop the *ARM* core if both processors write a 1 to their respective bits. Bit 5 of AHCON/DHCON allows the *ARM* core to immediately stop the DSP16000 core. When the appropriate bit is cleared, each processor resumes from the point at which it was stopped.

Bit 6 of AHCON allows the DSP16000 core to automatically stop the *ARM* core whenever the DSP16000 core is in debug mode. Bit 4 of DHCON must also have been set by the *ARM* core to enable this feature. Bit 3 of DHSTAT indicates whether the DSP16000 core is currently trying to stop the *ARM* core, either through bit 4 or bit 6 of AHCON.

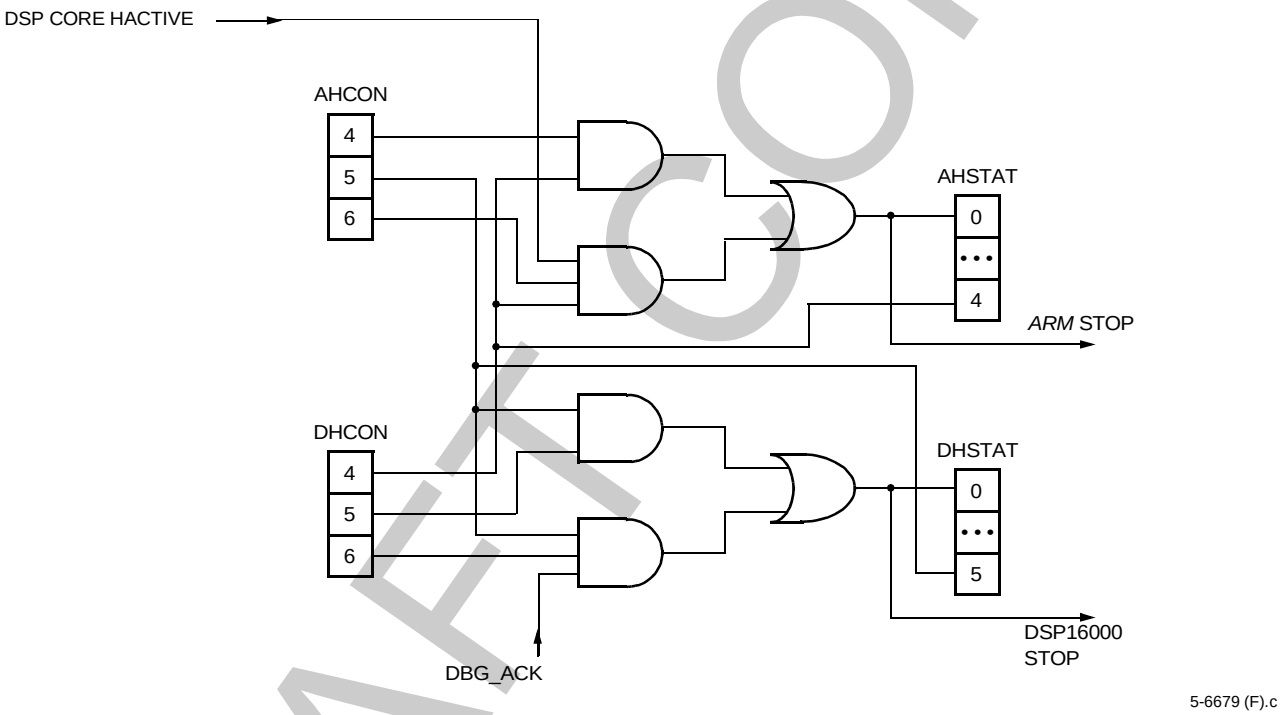


Figure 9.2-3 Behavior of AHCON and DHCON Bits[6:4]

Bit 6 of DHCON allows the *ARM* core to automatically stop the DSP16000 core whenever *ARM* is in debug mode. Bit 5 of AHCON must also have been set by the DSP16000 core to enable this feature. Bit 3 of AHSTAT indicates whether the *ARM* core is currently trying to stop the DSP16000 core, either through bit 5 or bit 6 of DHCON. Figure 9.2-3 illustrates the behavior of bits 3—6 of the AHCON and DHCON registers.

Bit 7 of AHCON/DHCON enables a feature that stops the *ARM* core on exit from its DBGRQ handler until the DSP16000 core also exits from its own HDS trap handler. Similarly, bit 8 of AHCON/DHCON enables a feature that stops the DSP16000 core on exit from its HDS trap handler until the *ARM* core also exits from its DBGRQ handler. These bits will be used by the debuggers to synchronize resuming execution on the two cores on a simultaneous breakpoint.

The upper 2 bits of the DHCON register (bits 14 and 15) contain handshake bits that are writable by the *ARM* core and readable by the DSP16000 core via the corresponding bits of the AHSTAT register. Similarly, the upper 2 bits of the AHCON register (bits 14 and 15) contain handshake bits that are writable by the DSP16000 core and readable by the *ARM* core via the corresponding bits of the DHSTAT register. These bits are reserved for use by the HDS software to aid in synchronization of the two processors while entering and exiting debug mode.

9 ICP/IDP (continued)

9.2 Interprocessor Debug Port (IDP) (continued)

Table 9.2-1 DHCON Control Register in IDP (Controlled by *ARM* Core)

Bit	15	14	13—9	8	7	6	5	4	3	2	1	0
Name	AHSIG1	AHSIG0	RSVD	DDHWAIT	DAHWAIT	DDHSLP	DDSLP	DASLP	DSYTRAP	DEXTERN	DA_DBK	DD_ABK

Bit	Name	Description*
15	AHSIG1	HDS handshake signal 1 from *ARM* core to the DSP16000 core; readable by the DSP16000 core via AHSTAT register bit 15.
14	AHSIG0	HDS handshake signal 0 from *ARM* core to the DSP16000 core; readable by the DSP16000 core via AHSTAT register bit 14.
13—9	RSVD	Reserved.
8	DDHWAIT	If set, stop the DSP16000 core on exit from HDS trap handler until *ARM* core also exits from DBGRQ handler.
7	DAHWAIT	If set, stop the *ARM* core on exit from DBGRQ handler until the DSP16000 core also exits from HDS trap handler.
6	DDHSLP	If set, stop the DSP16000 core automatically whenever the *ARM* core is executing in debug mode (based on value of DBG_ACK signal).
5	DDSLP	If set and AHCON[5] = 1, stop the DSP16000 core until DDSLP is subsequently cleared by the *ARM* core.
4	DASLP	If set, allow the DSP16000 to stop the *ARM* core by setting AHCON[4] = 1 or by setting AHCON[6] = 1 and entering debug mode.
3	DSYTRAP	If set and AHCON[3] = 1, generate an immediate SYSTRAP request to the DSP16000.
2	DEXTERN	If set, allow the DSP16000 to assert DBGRQ to the *ARM* ICE module by setting AHCON[2] = 1.
1	DA_DBK	If set and AHCON[1] = 1, the detection of a breakpoint in the *ARM* ICE module will also assert SYSTRAP to the DSP16000.
0	DD_ABK	If set and AHCON[0] = 1, the assertion of HTRAP to the DSP16000 HDS will also assert DBGRQ to the *ARM* ICE module.

* All reset values are 0.

Table 9.2-2 DHSTAT Status Register in IDP (Readable by *ARM* Core)

Bit	15	14	13—6	5	4	3	2	1	0
Name	DHSIG1	DHSIG0	RSVD	DSLRDY	RSVD	DSREQ	DDREQ	DDEBUG	DSLEEP

Bit	Name	Description*
15	DHSIG1	Handshake signal 1 from the DSP16000 core to *ARM* core; reflects value in bit 15 of AHCON register.
14	DHSIG0	Handshake signal 0 from the DSP16000 core to *ARM* core; reflects value in bit 14 of AHCON register.
13—6	RSVD	Reserved.
5	DSLRDY	If 1, the DSP16000 core will allow itself to be stopped by the *ARM* core (reflects the status of AHCON[5]). The *ARM* core can then stop the DSP16000 core by setting DHCON[5].
4	RSVD	Reserved.
3	DSREQ	If 1, the DSP16000 core is trying to stop the *ARM* core, due to HACTIVE and AHCON[6] = 1, or if AHCON[4] = 1.
2	DDREQ	If 1, the DSP16000 is trying to assert DBGRQ to the *ARM* ICE module, due to HTRAP and AHCON[0] = 1, or if AHCON[2] = 1.
1	DDEBUG	If 1, the DSP16000 core is in the HTRAP service routine (reflects HACTIVE signal).
0	DSLEEP	If 1, the DSP16000 core has been stopped via DHCON bit 5 or bit 6 (DDSLP/DDHSLP).

* All reset values are 0.

9 ICP/IDP (continued)

9.2 Interprocessor Debug Port (IDP) (continued)

Table 9.2-3 AHCON Control Register in IDP (Controlled by the DSP16000 Core)

Bit	15	14	13—9	8	7	6	5	4	3	2	1	0
Name	DHSIG1	DHSIG0	RSVD	ADHWAIT	AAHWAIT	AAHSLP	ADSLP	AASLP	ASYTRAP	AEXTERN	AA_DBK	AD_ABK

Bit	Name	Description*
15	DHSIG1	HDS handshake signal 1 from the DSP16000 core to *ARM* core; readable by the DSP16000 core via DHSTAT register bit 15.
14	DHSIG0	HDS handshake signal 0 from the DSP16000 core to *ARM* core; readable by the DSP16000 core via DHSTAT register bit 14.
13—9	RSVD	Reserved.
8	ADHWAIT	If set, stop the DSP16000 core on exit from HDS trap handler until *ARM* core also exits from DBGRQ handler.
7	AAHWAIT	If set, stop the *ARM* core on exit from DBGRQ handler until the DSP16000 core also exits from HDS trap handler.
6	AAHSLP	If set, stop the *ARM* core automatically whenever the DSP16000 core is executing in debug mode (based on value of HACTIVE signal).
5	ADSLP	If set, allow the *ARM* core to stop the DSP16000 core either by setting DHCON[5] = 1 or by setting DHCON[6] = 1 and entering debug mode.
4	AASLP	If set and DHCON[4] = 1, stop the *ARM* core until AASLP is subsequently cleared by the DSP16000 core.
3	ASYTRAP	If set, allow *ARM* core to assert SYSTRAP by setting DHCON[3] = 1.
2	AEXTERN	If set and DHCON[2] = 1, generate an immediate DBGRQ request to the *ARM* ICE module.
1	AA_DBK	If set and DHCON[1] = 1, the detection of a breakpoint to the *ARM* ICE module will also assert SYSTRAP to the DSP16000.
0	AD_ABK	If set and DHCON[0] = 1, the assertion of HTRAP to the DSP16000 will also assert DBGRQ to the *ARM* ICE module.

* All reset values are 0.

Table 9.2-4 AHSTAT Status Register in IDP (Readable by the DSP16000 Core)

Bit	15	14	13—5	4	3	2	1	0
Name	AHSIG1	AHSIG0	RSVD	ASLRDY	ASREQ	ADREQ	ADEBUG	ASLEEP

Bit	Name	Description*
15	AHSIG1	HDS handshake signal 1 from *ARM* core to the DSP16000 core; reflects value in bit 15 of DHCON register.
14	AHSIG0	HDS handshake signal 0 from *ARM* core to the DSP16000 core; reflects value in bit 14 of DHCON register.
13—5	RSVD	Reserved.
4	ASLRDY	If 1, the *ARM* core will allow itself to be stopped by the DSP16000 core (reflects the status of DHCON[4]). The DSP16000 core can then stop the *ARM* core by setting AHCON[4].
3	ASREQ	If 1, the *ARM* core is trying to stop the DSP16000 core, due to DBG_ACK and DHCON[6] = 1, or if DHCON[5] = 1.
2	ADREQ	If 1, the *ARM* core is trying to assert SYSTRAP to the DSP16000, due to DBGRQ and DHCON[1] = 1, or if DHCON[3] = 1.
1	ADEBUG	If 1, the *ARM* core is currently executing in debug mode (reflects the value of DBG_ACK).
0	ASLEEP	If 1, the *ARM* core has been stopped via AHCON bit 4 or bit 6 (AASLP/AAHSLP).

* All reset values are 0.

10 Device Characteristics

10.1 Absolute Maximum Ratings

Stresses in excess of the absolute maximum ratings can cause permanent damage to the device. These are absolute stress ratings only. Functional operation of the device is not implied at these or any other conditions in excess of those given in Section 10.3. Exposure to absolute maximum ratings for extended periods can adversely affect device reliability.

External leads can be bonded and soldered safely at temperatures of up to 300 °C in approximately 10 s.

Table 10.1-1 Absolute Maximum Ratings

Parameter	Min	Max	Unit
Voltage Range on VDD_CORE with Respect to Ground	–0.5	2.0	V
Voltage Range on VDD_IO_1P8 with Respect to Ground	–0.5	2.3	V
Voltage Range on VDDA_D or VDDA_U with Respect to Ground	–0.5	2.0	V
Voltage Range on VRTC with Respect to Ground	–0.5	2.0	V
Voltage Range on Signal Pin (VDD_IO_IP8)	–0.3	VDD_IO_IP8 + 0.3 V*	V
Voltage Range on Signal Pin (VDDA_D)	–0.3	VDDA_D + 0.3 V	V
Voltage Range on Signal Pin (VRTC)	–0.3	VRTC + 0.3 V	V
Power Dissipation			W
Junction Temperature (TJ)	–40	125	°C
Storage Temperature Range	–65	150	°C

* Not to exceed 2.3 V.

10.2 Handling Precautions

Although electrostatic discharge (ESD) protection circuitry has been designed into this device, proper precautions must be taken to avoid exposure to ESD and electrical overstress (EOS) during all handling, assembly, and test operations. Agere employs both a human-body model (HBM) and a charged-device model (CDM) qualification requirement in order to determine ESD-susceptibility limits and protection design evaluation. ESD voltage thresholds are dependent on the circuit parameters used in each of the models, as defined by JEDEC's JESD22-A114 (HBM) and JESD22-C101 (CDM) standards.

Table 10.2-1 Handling Precautions

Device	Minimum HBM Threshold	Minimum CDM Threshold
T8307	—	—

10.3 Recommended Operating Conditions

Table 10.3-1 Recommended Operating Conditions

Parameter	Signal	Min	Max	Unit
Power Supply Voltage (CORE)	VDD_CORE	1.43	1.57	V
Power Supply Voltage (1.8 V I/O)	VDD_IO_1P8	1.71	1.89	V
Power Supply Voltage (analog)	VDDA_D or VDDA_U	1.43	1.57	V
Power Supply Voltage (RTC)	VRTC	1.43	1.57	V
Power Supply Ground	VSS	0	0	V
Ambient Temperature	TA	–40	85	°C

11 Electrical Characteristics

The following electrical characteristics are preliminary and are subject to change. Electrical characteristics refer to the behavior of the device under specified conditions. Electrical requirements refer to conditions imposed on the user for proper operation of the device. The parameters below are valid for the conditions described in Section 10.3.

Table 11.1 Electrical Characteristics for 1.8 V I/O Pins

Parameter	Symbol	Min	Typical	Max	Unit
Input Voltage: Low (V_{IL} = 0 V) High (V_{IH} = 1.8 V)	 V_{IL} V_{IH}	 –0.3 0.70 x V_{DD}_CORE	 — —	 0.30 x V_{DD}_CORE V_{DD}_IO_1P8 + 0.3	 V V
Input Current (no pull-up): Low High	 I_{IL} I_{IH}	 — —	 — —	 1.0 1.0	 ∝A ∝A
Output Low Voltage: I_{OL} = 10 mA (Group 1), 4 mA (Group 2), 2 mA (Group 3)*	V_{OL}	—	—	0.25 x V_{DD}_IO_1P8	V
Output High Voltage: I_{OH} = –10 mA (Group 1), –4 mA (Group 2), –2 mA (Group 3)*	V_{OH}	0.75 x V_{DD}_IO_1P8	—	—	V
Output 3-State Current: Low (V_{IL} = 0 V) High (V_{IH} = 1.8 V)	 I_{OZL} I_{OZH}	 — —	 — —	 10 10	 ∝A ∝A
Input Capacitance	C_I	—	2.0	—	pF
Pull-up Resistance (where applicable)	R_{PU}	—	200	—	k.

* Group 1: *ARM* EMI output pins. Group 2: DSP SEMI output pins, *ARM* chip selects, USB pins, SD/MMC pins. Group 3: other output pins.

Table 11.2 Electrical Requirements and Characteristics for Small-Signal Input Clock Buffer

Parameter	Symbol	Min	Typ	Max	Unit
Small-signal Peak-to-peak Voltage (on CKI)	Vp-p	0.4	—	1.5	V
Small-signal Maximum Input Voltage	V_{max}	—	—	V_{DD}A_D	V
Small-signal Minimum Input Voltage	V_{min}	0	—	—	V
Input Resistance into CKI	R_{ss}	—	25	—	k.
Input Capacitance into CKI	C_{ss}	—	1.5	—	pF
Small-signal Input Duty Cycle	DCyc	30	—	70	%
Small-signal Buffer Frequency Range	f_{ss}	10	—	30	MHz
Start-up Time*	t_{ss}	—	20	50	∝s
Small-signal Buffer Supply Current (enabled)	I_{ss}	—	<100	—	∝A
Small-signal Buffer Supply Current (disabled)	I_{off}	—	—	1	∝A

* This specification assumes that the input clock is running and is stable before the buffer is enabled.

11 Electrical Characteristics (continued)

11.1 Typical Current Measurements

Current measurements during normal program execution are dependent on the distribution of instruction types executed, peripheral activity, the ratio of internal to external activity, and the number of wait-states used for external accesses. Only in-system measurements should be considered meaningful.

The following T8307 typical current measurement results are obtained under the condition of V_{DD_CORE} = 1.5 V, $V_{DD\_IO\_1P8}$ = 1.8 V, T_A = 20 °C. The numbers shown below are average values.

Table 11.3 T8307 Average Current Consumption (TBD)

TBD	TBD			
	TBD		TBD	
	TBD	TBD	TBD	TBD
TBD	TBD	TBD	TBD	TBD
TBD	TBD	TBD	TBD	TBD

11.2 Real-Time Clock (RTC) Circuit Electrical Characteristics

Table 11.4 lists the electrical specifications of the crystal oscillator circuit as shown in Figure 7.11-2.

Table 11.4 32.768 kHz Crystal Oscillator Electrical Characteristics

Parameter	Min	Typ	Max	Unit
Supply Voltage	1.35	1.5	1.65	V
Start-up Time	—	—	5	s
Supply Current (oscillator enabled)	—	10	15	∝A
Supply Current (oscillator disabled)	—	1	—	∝A

12 Timing Characteristics and Requirements

The following timing characteristics and requirements are preliminary information and are subject to change. Timing characteristics refer to the behavior of the device under specified conditions. Timing requirements refer to conditions imposed on the user for proper operation of the device. All timing data is valid for the following conditions:

T_A = –40 °C to +85 °C (see Section 10.3).

V_{DD}_IO_1P8 = 1.8 V ± 0.09 V, V_{SS} = 0 V (see Section 10.3).

Capacitance load on outputs (C_L) = 50 pF.

Output characteristics can be derated as a function of load capacitance (C_L).

All outputs except CKO_IACK: 0.025 ns/pF = dt/dC_L = 0.07 ns/pF for 10 = C_L = 100 pF.

CKO_IACK: 0.01 ns/pF = dt/dC_L = 0.025 ns/pF for 10 = C_L = 100 pF.

at V_{IH} for rising edge and at V_{IL} for falling edge.

For example, if the actual load capacitance on a pin other than CKO_IACK is 30 pF instead of 50 pF, the maximum derating for a rising edge is (30 – 50) pF x 0.07 ns/pF = 1.4 ns **less** than the specified rise time or delay that includes a rise time. The minimum derating for the same 30 pF load would be (30 – 50) pF x 0.025 ns/pF = 0.5 ns.

Test conditions for inputs:

- Rise and fall times of 4 ns or less.
- Timing reference levels for delays = V_{IH}, V_{IL}.

Test conditions for outputs (unless noted otherwise):

- C_{LOAD} = 40 pF.
- Timing reference levels for delays = V_{IH}, V_{IL}.
- 3-state delays measured to the high-impedance state of the output driver.

Unless otherwise noted, CKO_IACK in the timing diagrams is the free-running CKO.

12 Timing Characteristics and Requirements (continued)

12.1 CP and DSP Reset Circuit

The device has two external reset pins: RESETN and TRSTN. When a device reset is required, RESETN and TRSTN signals must be asserted simultaneously to initialize the device. During the power supply voltage ramping period, the external reset circuit is responsible for holding the RESETN and TRSTN input low. Figure 12.1-1 shows two separate events:

1. Device reset at initial powerup.
2. Device reset following a drop in power supply.

Note: The TRSTN pin must be asserted even if the JTAG controller is not used by the application.

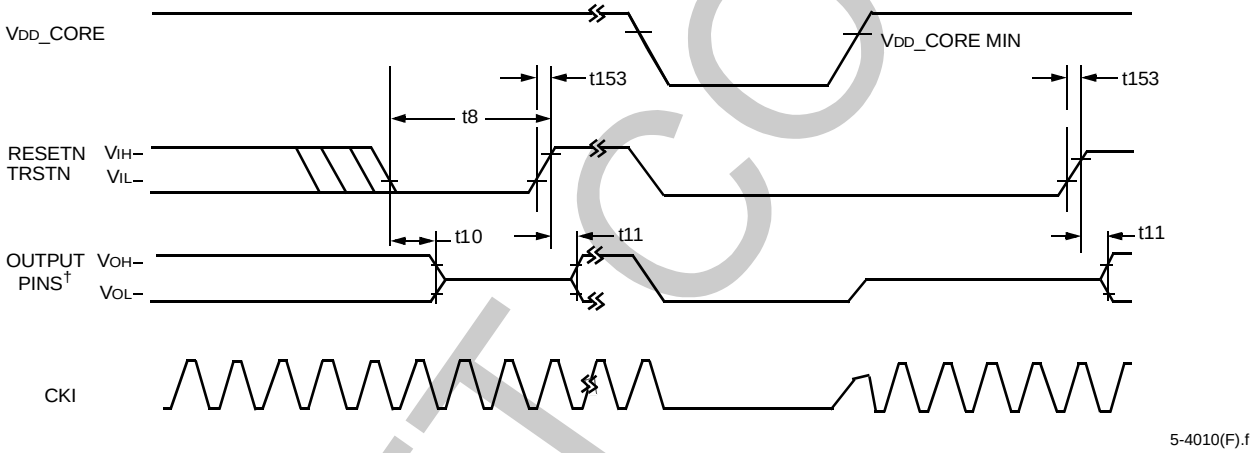


Figure 12.1-1 Powerup Reset and Device Reset Timing Diagram

Table 12.1-1 Timing Requirements for Powerup Reset and Device Reset

Abbreviated Reference	Parameter	Min	Max	Unit
t8	RESETN and TRSTN Reset Pulse (low to high)	6T*	—	ns
t153	RESETN and TRSTN Rise (low to high)	—	115†	ns

* T = internal clock period (CLK).
† Assume 20 pF capacitor on the RESETN input pin.

Table 12.1-2 Timing Characteristics for Powerup Reset and Device Reset

Abbreviated Reference	Parameter	Min	Max	Unit
t10	RESETN Disable Time (low to 3-state)	—	100	ns
t11	RESETN Enable Time (high to valid)	—	100	ns

Note: The device needs to be clocked for at least six CKI cycles during external reset. Otherwise, high currents flow.

12 Timing Characteristics and Requirements (continued)

12.2 DSP JTAG

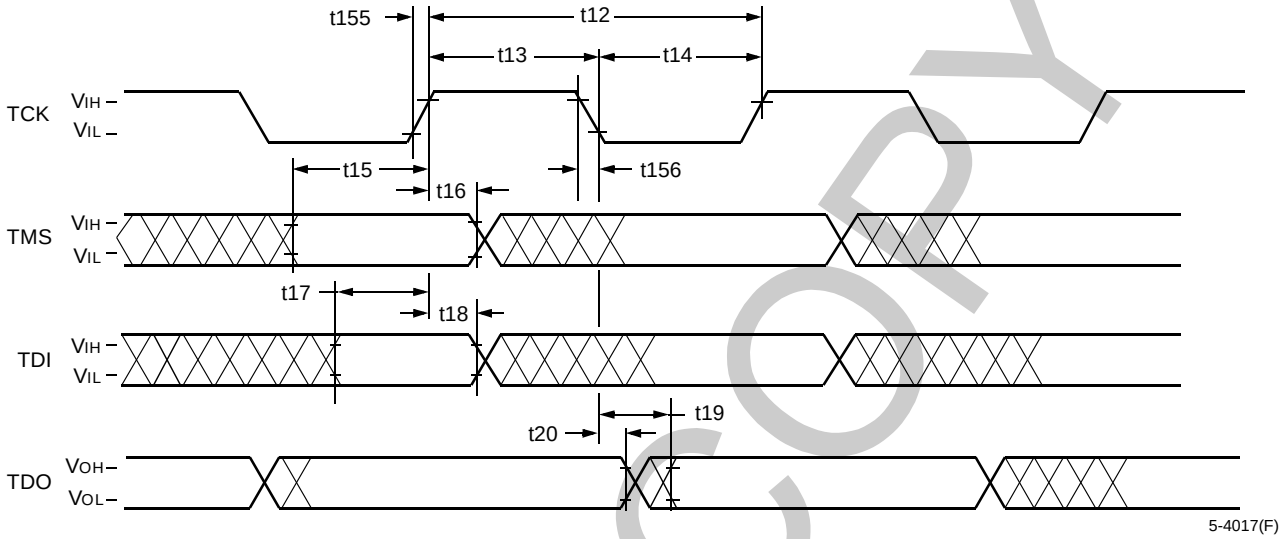


Figure 12.2-1 JTAG I/O Timing Diagram

Table 12.2-1 Timing Requirements for JTAG I/O

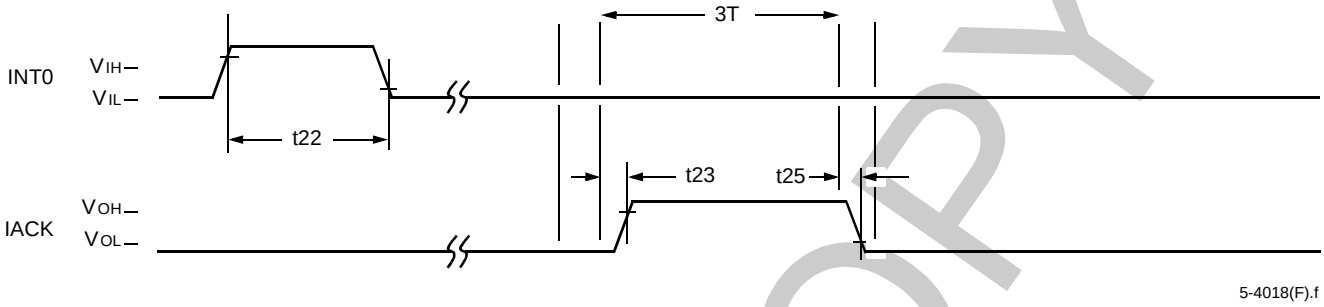
Abbreviated Reference	Parameter	Min	Max	Unit
t12	TCK Period (high to high)	50	—	ns
t13	TCK High Time (high to low)	22.5	—	ns
t14	TCK Low Time (low to high)	22.5	—	ns
t155	TCK Rise Transition Time (low to high)	0.6	—	V/ns
t156	TCK Fall Transition Time (high to low)	0.6	—	V/ns
t15	TMS Setup Time (valid to high)	7.5	—	ns
t16	TMS Hold Time (high to invalid)	5.0	—	ns
t17	TDI Setup Time (valid to high)	7.5	—	ns
t18	TDI Hold Time (high to invalid)	5.0	—	ns

Table 12.2-2 Timing Characteristics for JTAG I/O

Abbreviated Reference	Parameter	Min	Max	Unit
t19	TDO Delay (low to valid)	—	15	ns
t20	TDO Hold (low to invalid)	0	—	ns

12 Timing Characteristics and Requirements (continued)

12.3 DSP Interrupt



† CKO is free-running.

Figure 12.3-1 Interrupt and Trap Timing Diagram

Table 12.3-1 Timing Requirements for Interrupt

Abbreviated Reference	Parameter	Min	Max	Unit
t22	INT Assertion Time (high to low)	2T*	—	ns

* T = internal clock period (CLK).

Note: Interrupt is asserted during an interruptible instruction and no other pending interrupts.

Table 12.3-2 Timing Characteristics for Interrupt

Abbreviated Reference	Parameter	Min	Max	Unit
t23	IACK Valid Time (low to high)	—	10	ns
t25	IACK Invalid Time (low to low)	—	10	ns

Note: Interrupt is asserted during an interruptible instruction and no other pending interrupts.

12 Timing Characteristics and Requirements

12.4 DSP Bit I/O

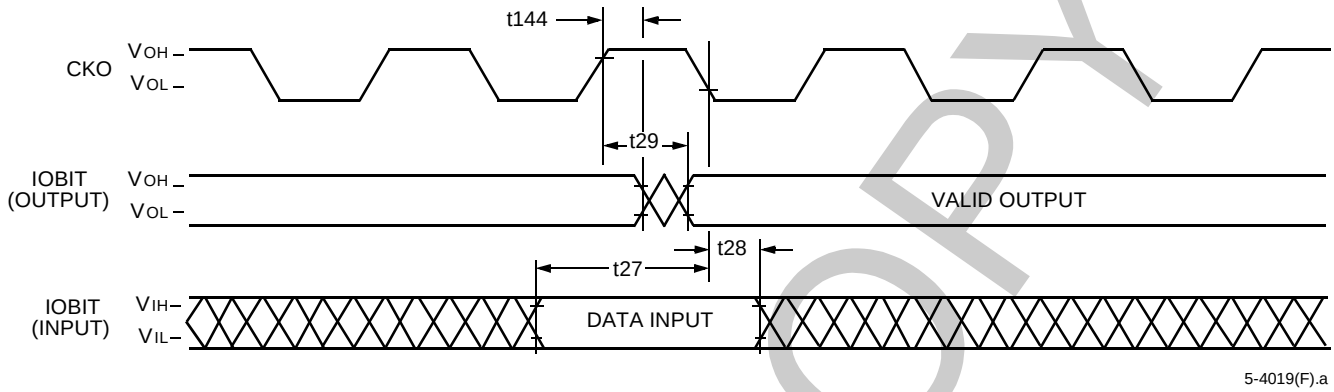


Figure 12.4-1 Write Outputs Followed by Read Inputs (cbit = IMMEDIATE; a1 = sbit)

Table 12.4-1 Timing Requirements for BIO Input Read

Abbreviated Reference	Parameter	Min	Max	Unit
t27	IOBIT Input Setup Time (valid to low)	10	—	ns
t28	IOBIT Input Hold Time (low to invalid)	0	—	ns

Table 12.4-2 Timing Characteristics for BIO Output

Abbreviated Reference	Parameter	Min	Max	Unit
t29	IOBIT Output Valid Time (high to valid)	—	9	ns
t144	IOBIT Output Hold Time (high to invalid)	1	—	ns

12 Timing Characteristics and Requirements (continued)

12.5 DSP System and External Memory Interface (SEMI)

12.5.1 Asynchronous Interface

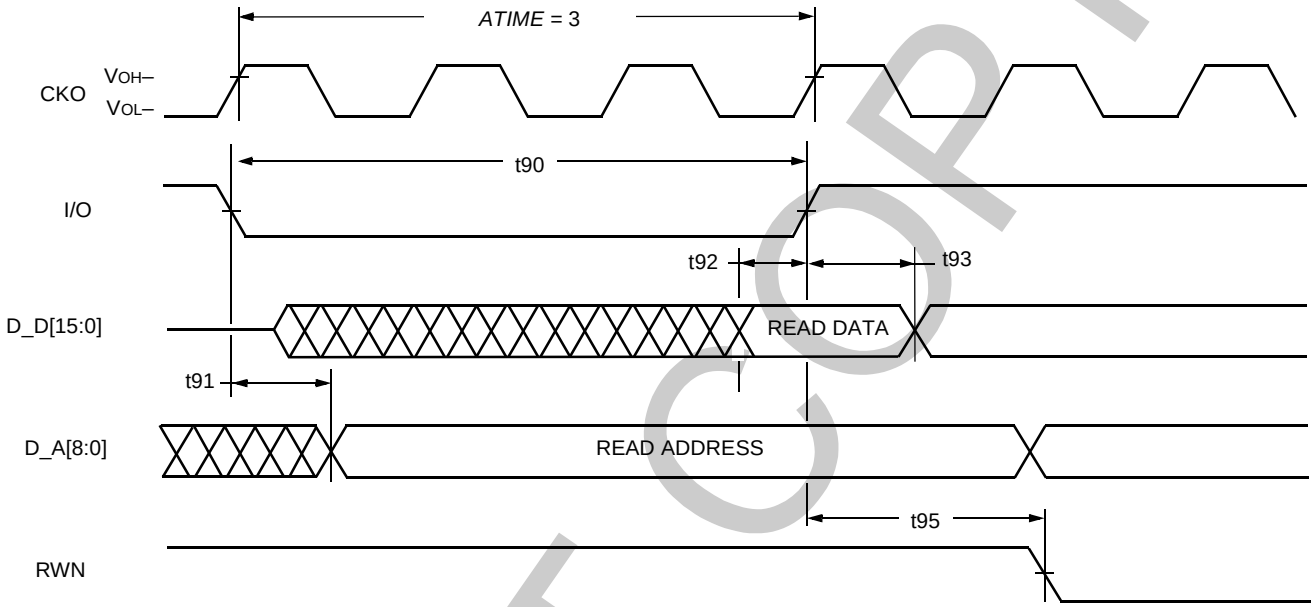


Figure 12.5-1 Asynchronous Read Timing Diagram (RHOLD = 0 and RSETUP = 0)

Table 12.5-1 Timing Requirements for Asynchronous Memory Read Operations

Abbreviated Reference	Parameter	Min	Max	Unit
t92	Read Data Setup (valid to I/O high)	5	—	ns
t93	Read Data Hold (I/O high to invalid)	0	—	ns

Table 12.5-2 Timing Characteristics for Asynchronous Memory Read Operations

Abbreviated Reference	Parameter	Min	Max	Unit
t90	I/O Width (low to high)	(T^{*} · ATIME) – 3	—	ns
t91	Address Delay (I/O low to valid)	—	2 – ($T^{\dagger}$ · $RSETUP^{\dagger}$)	ns
t95	RWN Activation (I/O high to RWN low)	$T^{\dagger}$ · (1 + $RHOLD^{\ddagger}$ + $WSETUP^{\S}$) – 3	—	—

* T = internal clock period (CLK).

† RSETUP = **ECON0**[12].

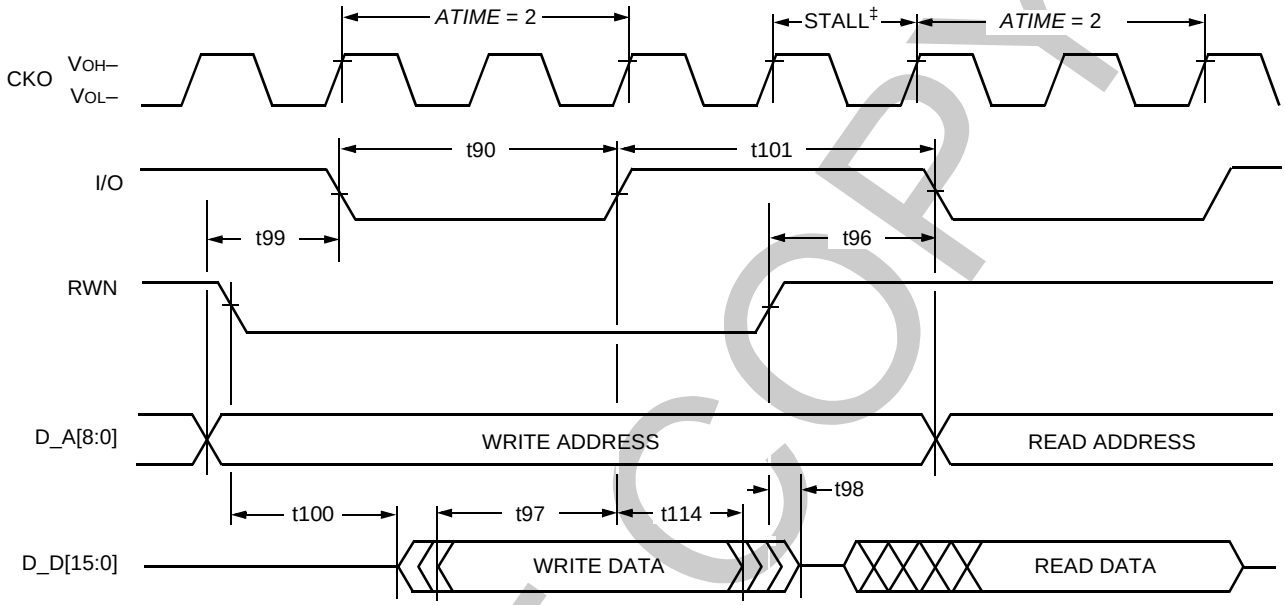
‡ RHOLD = **ECON0**[14].

§ WSETUP = **ECON0**[13].

Note: The external memory access time from the asserting of *I/O* can be calculated as t90 – (t91 + t92).

12 Timing Characteristics and Requirements (continued)

12.5 DSP System and External Memory Interface (continued)



† CKO_IACK reflects CLK, i.e., **ECON1**[1:0] = 1.
‡ The stall cycle is caused by the read following the write.

Figure 12.5-2 Asynchronous Write Timing Diagram (WHOLD = 0, WSETUP = 0)

Table 12.5-3 Timing Characteristics for Asynchronous Memory Write Operations

Abbreviated Reference	Parameter	Min	Max	Unit
t90	I/O Width (low to high)	(T* · ATIME) – 3	—	ns
t96	Enable Delay (RWN high to I/O low)	T* · (1 + WHOLD† + RSETUP‡) – 3	—	ns
t97	Write Data Setup (valid to I/O high)	(T* · ATIME) – 3	—	ns
t98	Write Data Deactivation (RWN high to 3-state)	—	3	ns
t99	Write Address Setup (valid to I/O low)	T* · (1 + WSETUP§) – 3	—	ns
t100	Write Data Activation (RWN low to low-Z)	T* – 2	—	ns
t101	Address Hold Time (I/O high to invalid)	T* · (1 + WHOLD‡) – 3	—	ns
t114	Write Data Hold Time (I/O high to invalid)	T – 3	—	ns

* T = internal clock period (CLK).

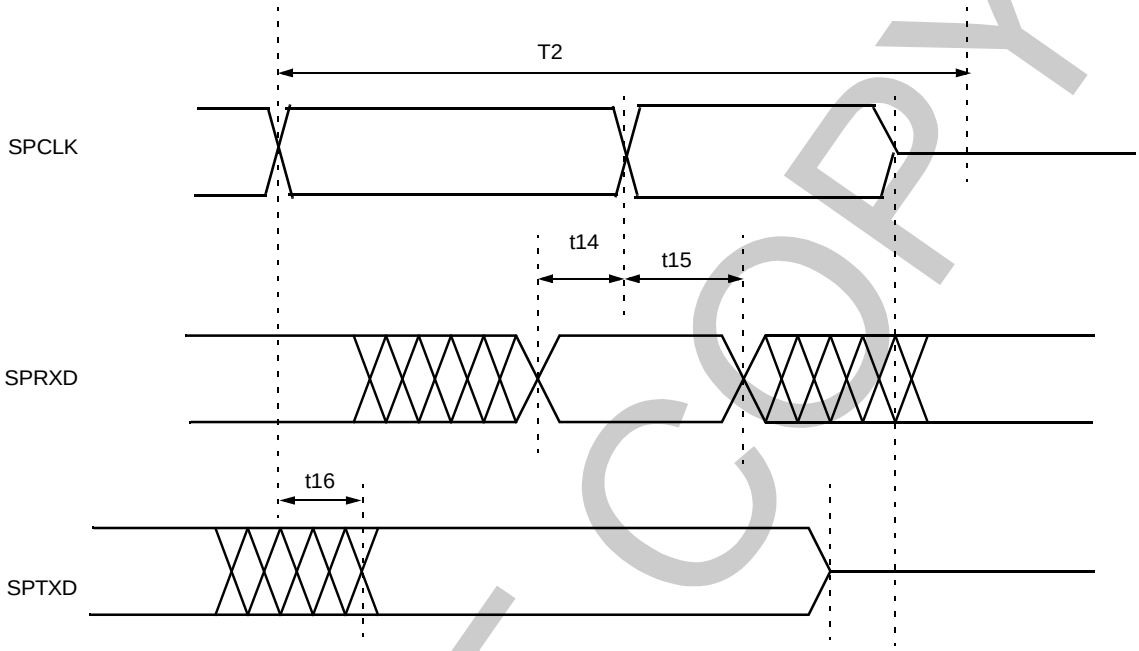
† WHOLD = **ECON0**[15].

‡ RSETUP = **ECON0**[12].

§ WSETUP = **ECON0**[13].

12 Timing Characteristics and Requirements (continued)

12.6 CP-Side and DSP-Side SSP



Note: T2: Clock period of SPCLK, an external clock generated by a clock divider with programmable polarity.

Figure 12.6-1 SSP Interface Timing Diagram as Master

Table 12.6-1 SSP Interface Timing Table as Master

Symbol	Signal	Type	Reference	Min	Max	Unit
t14	SPRXD	Input Setup	SPCLK Rise	0	—	ns
t15	SPRXD	Input Hold	SPCLK Rise	2	—	ns
t16	SPTXD	Output Valid	SPCLK Rise	7	24	ns

12 Timing Characteristics and Requirements (continued)

12.7 CP-Side and DSP-Side I²S

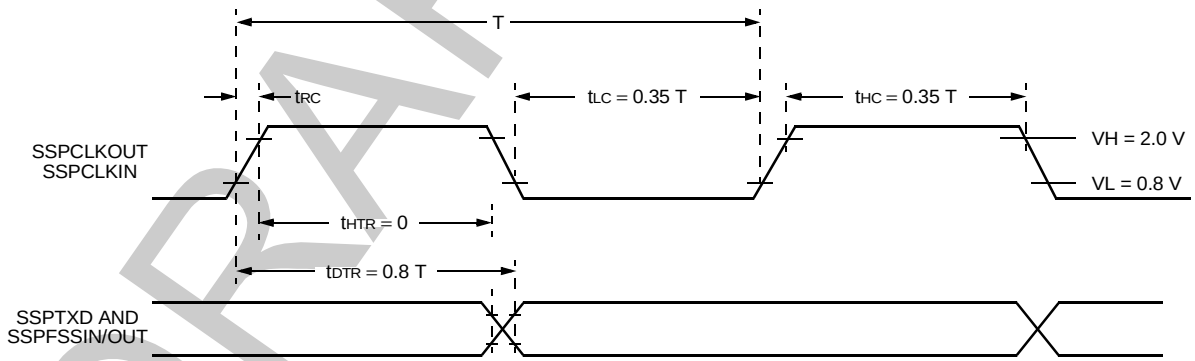
The receiver samples SSPRXD signal on the leading edge of SSPCLKIN serial clock and latches the data. SSPFSSIN is also sampled on rising edge of input clock. The transmitter samples SSPTXD signal on the leading edge of SSPCLKOUT serial clock and latches the data. SSPFSSOUT is synchronized to the falling edge of output clock.

Taking into account the propagation delays between master clock and the data and word select signals, that the total delay is sum of:

- The delay between the external (master) clock and the slave's internal clock; and
- The delay between the internal clock and the data and/or word select signals.

For data and word-select inputs, the external to internal clock delay is of no consequence because it only lengthens the effective setup time (see Figure 12.7-1). The major part of the time margin is to accommodate the difference between the propagation delay of the transmitter, and the time required to setup the receiver. All timings are specified relative to the clock period or to the minimum allowed clock period of a device.

- The system clock period T must be greater than T_{tr} and T_r because both transmitter and receiver have to be able to handle the data transfer rate.
- At all data rates in the master mode, the transmitter or receiver generates a clock signal with a fixed mark/space ratio. For this reason, t_{HC} and t_{LC} are specified with respect to T.
- In the slave mode, the transmitter and receiver need a clock signal with minimum high and low periods so that they can detect the signal. So long as the minimum periods are greater than 0.35 T_r, any clock that meets the requirements can be used (see Figure 12.7-2).
- Because the delay (tdtr) and the maximum transmitter speed (defined by T_{tr}) are related, a fast transmitter driven by a slow clock edge can result in tdtr not exceeding t_{RC}, which means thtr becomes zero or negative. Therefore, the transmitter has to guarantee that thtr is greater than or equal to zero, so long as the clock rise time t_{RC} is not more than t_{RC}max, where t_{RC}max is not less than 0.15 T_{tr}.
- To allow data to be checked out on a falling edge, the delay is specified with respect to the rising edge of the clock signal and T, always giving the receiver sufficient setup time.
- The data setup and hold time must not be less than the specified receiver setup and hold time.



Notes:
T = Clock period.

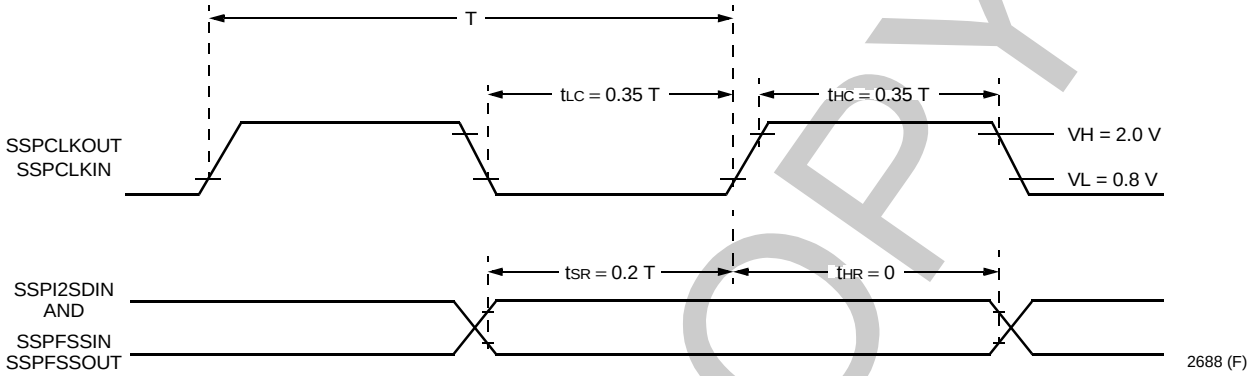
t_{RC} is only relevant for transmitters in slave mode.

t_R is the minimum allowed clock period for the transmitter.

Figure 12.7-1 Timing for I²S Transmitter

12 Timing Characteristics and Requirements (continued)

12.7 CP-Side and DSP-Side I²S (continued)



Note: T = Clock period.

Figure 12.7-2 Timing for I²S Receiver

Table 12.7-1 Example: Master Transmitter with Data Rate of 2.5 MHz (±10%) (in ns)

Name	Min	Typ	Max	Condition
Clock Period T	360	400	440	Ttr = 360
Clock High t_{HC}	160	—	—	min > 0.35 T = 140
Clock Low t_{LC}	160	—	—	min > 0.35 T = 140
Delay tdtr	—	—	300	max < 0.80 T = 320
Hold Time thtr	100	—	—	min > 0
Clock Rise Time t_{RC}	—	—	60	max > 0.15 T

Table 12.7-2 Slave Receiver

Name	Min	Typ	Max	Condition
Clock Period T	360	400	440	Ttr
Clock High t_{HC}	110	—	—	min < 0.35 T = 126
Clock Low t_{LC}	110	—	—	min < 0.35 T = 126
Setup Time tsr	60	—	—	min < 0.20 T = 72
Hold Time thtr	0	—	—	min < 0

12 Timing Characteristics and Requirements (continued)

12.8 CP Block External Memory Interface (SMC)

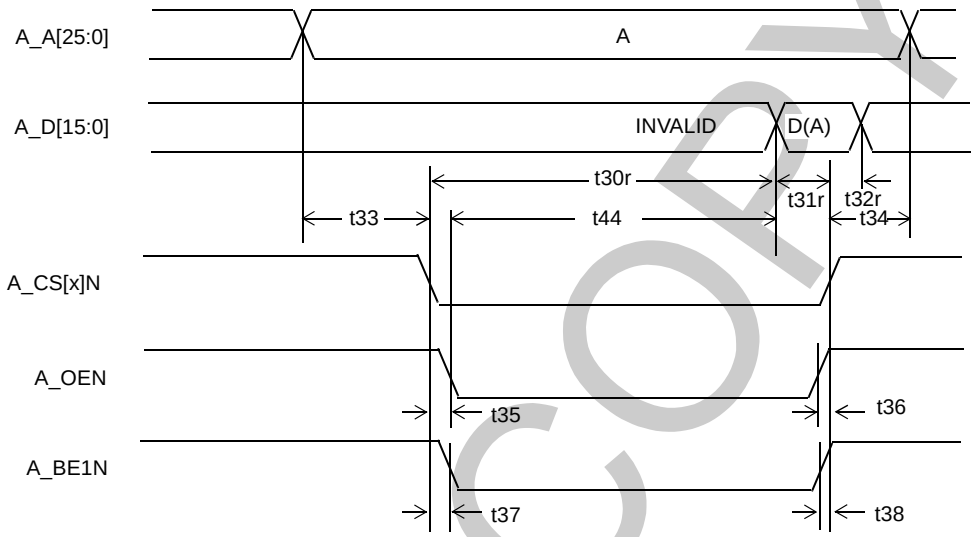


Figure 12.8-1 External Memory Read Timing Diagram

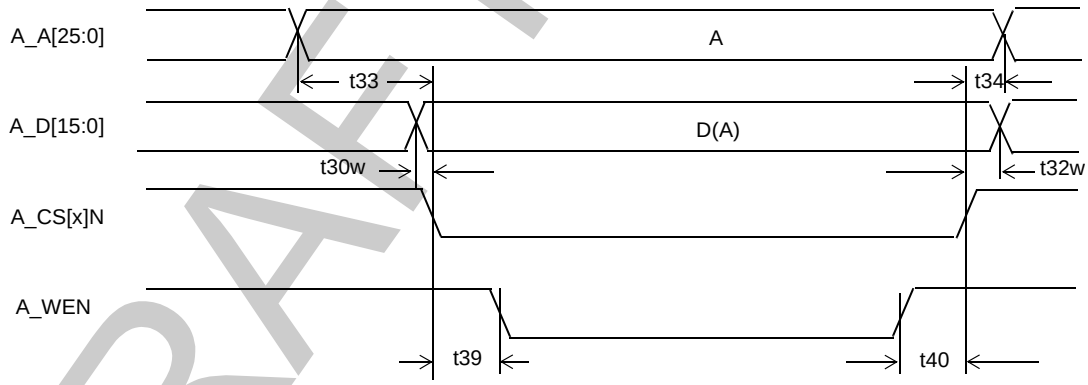


Figure 12.8-2 External Memory Write Timing Diagram

12 Timing Characteristics and Requirements (continued)

12.8 CP Block External Memory Interface (SMC) (continued)

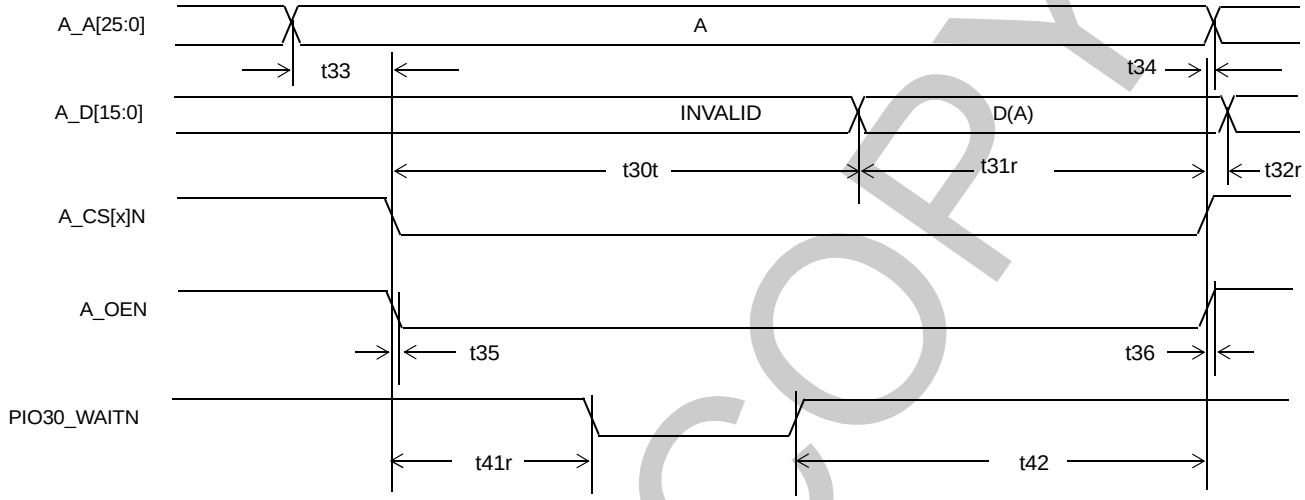


Figure 12.8-3 External Wait-Timed Read Transfer

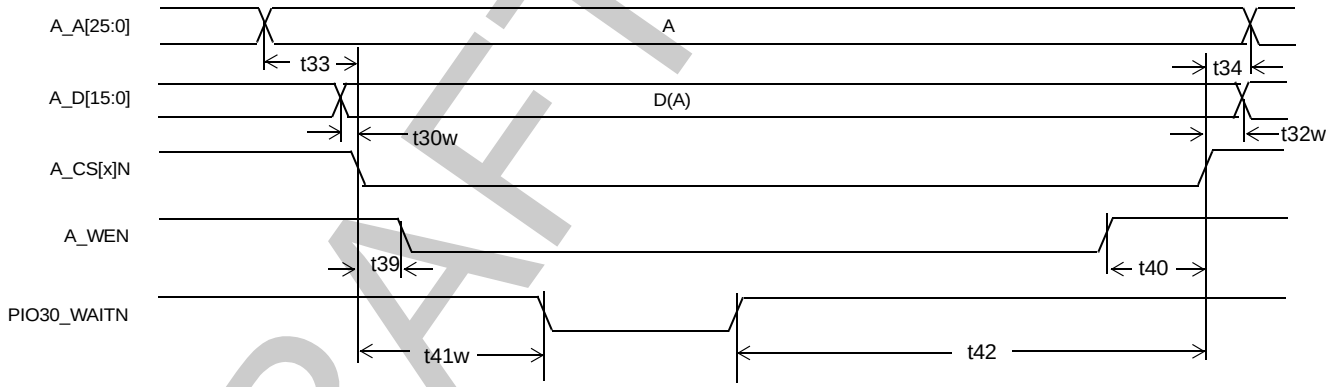


Figure 12.8-4 External Wait-Timed Write Transfer

12 Timing Characteristics and Requirements (continued)

12.8 CP Block External Memory Interface (SMC) (continued)

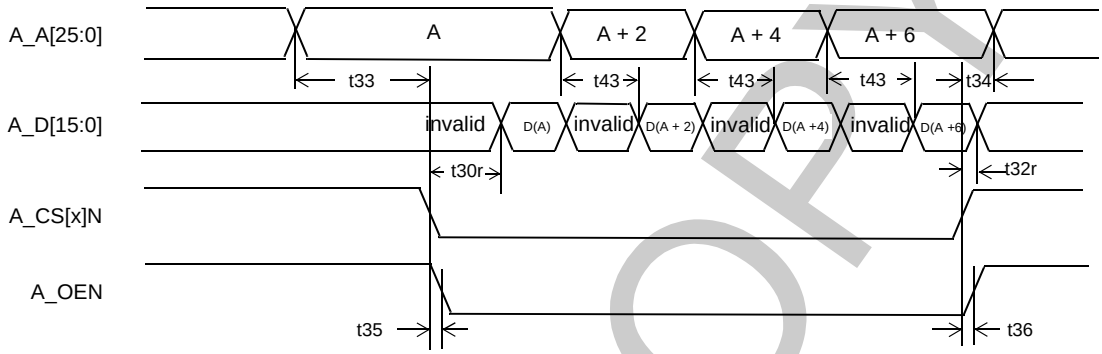


Figure 12.8-5 External Memory Zero Wait-State Fixed-Length Read Timing Diagram

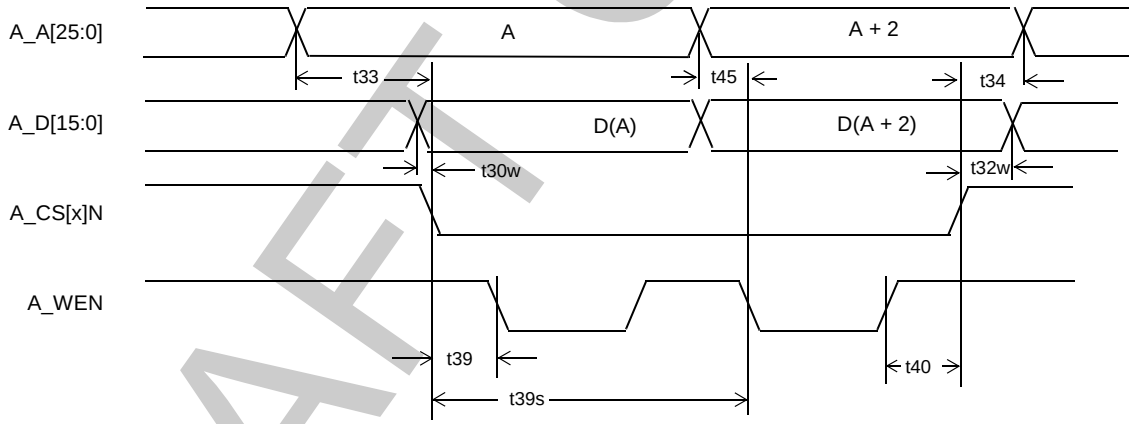


Figure 12.8-6 External Memory Two Zero Wait-Writes Timing Diagram

12 Timing Characteristics and Requirements (continued)

12.8 CP Block External Memory Interface (SMC) (continued)

Table 12.8-1 Timing Characteristics for SMC Asynchronous Memory Read and Write Operations

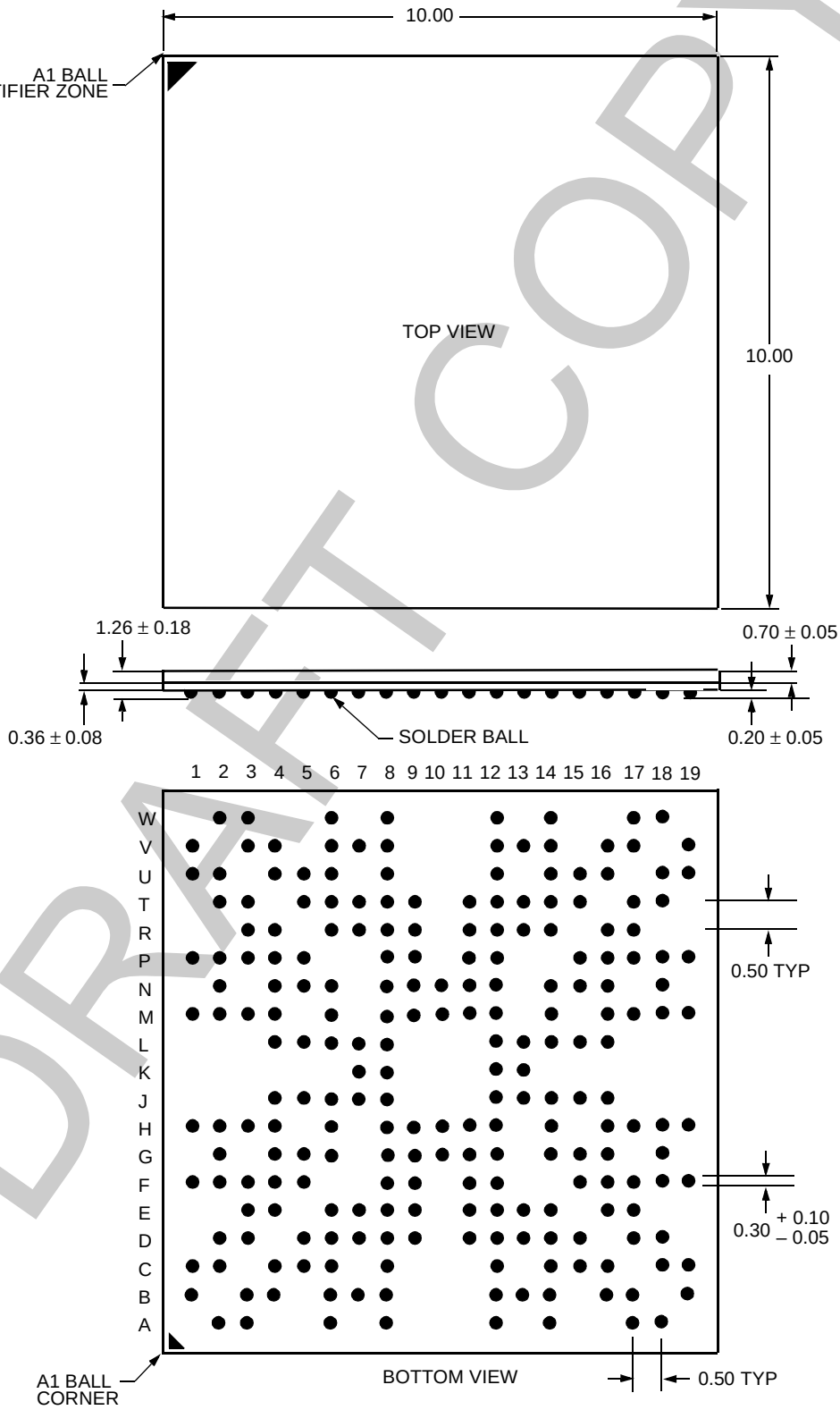
Abbreviated Reference	Parameter	Min	Max	Unit
t30r	CS[x]N Low to A_D Valid in Read Operation	—	(WST1* + 1) · tHCLK† – 7	ns
t30w	CS[x]N Low to A_D Valid in Write Operation	—	4	ns
t30t	CS[x]N Low to A_D Setup in Wait-Timed Read Operation	—	(WST1 + 2) · tHCLK – 7 + Ext. wait assert delay + Ext. wait deassert delay + Ext. wait sync delay	ns
t31r	A_D Valid to CS[x]N High in Read Operation	8	—	ns
t32r	A_D Hold after CS[x]N High in Read Operation	0	—	ns
t32w	A_D Hold after CS[x]N High in Write Operation	0	—	ns
t33	A_A Setup to CS[x]N low	—	tHCLK +5	ns
t34	CS[x]N High to A_A Hold	0	—	ns
t35	CS[x]N Low to A_OEN Low	WSTOEN‡ · t_{HCLK} – 4	WSTOEN · tHCLK +4	ns
t36	A_OEN High to CS[x]N High	—	WST2OEN · tHCLK +4	ns
t37	BE[x]N Low to CS[x]N Low	—	4	ns
t38	CS[x]N High to BE[x]N High	—	4	ns
t39	CS[x]N Low to A_WEN Low	(WSTWEN§+0.5) · tHCLK – 6	(WSTWEN + 0.5) · tHCLK + 3	ns
t39s	CS[x]N Low to Second A_WEN Low	(WST1 + WST2WEN + WSTWEN + 2.5) · tHCLK + 1	(WST1 + WST2WEN + WSTWEN + 2.5) · tHCLK + 3	ns
t40	A_WEN High to CS[x]N High	(WST2WEN+0.5) · tHCLK – 3	—	ns
t41r	CS[x]N Low to PIO30_WAITN Low in Read Operation	WST1 · tHCLK – 12 + Ext. wait assert delay	WST1 · tHCLK – 7 + Ext. wait assert delay	ns
t41w	CS[x]N Low to PIO30_WAITN Low in Write Operation	WST2** · tHCLK – 12 + Ext. wait assert delay	WST2 · tHCLK – 7 + Ext. wait assert delay	ns
t42	PIO30_WAITN High to CS[x]N High	2 · tHCLK + 1+ Ext. wait sync delay	2 · tHCLK + 9 + Ext. wait sync delay	ns
t43	Burst Read Access Time	(WST2+1) · tHCLK – 13	(WST2 + 1) · tHCLK – 7	ns
t44	A_OEN Low to A_D Setup	—	(WST1 + 1 – WSTOEN) · tHCLK – 8	ns
t45	Address Change to Second A_WEN Low	—	WSTWEN x tHCLK + 0.5 x tHCLK + 3	ns

* WST1: SMBWST1R<0—7>[4:0].
† tHCLK: Clock period of the ARM-side system clock.
‡ WSTOEN: SMBWSTOENR<0—7>[3:0].
§ WSTWEN: SMBWSTWENR<0—7>[3:0].
** WST2: SMBWST2R<0—7>[4:0].

13 Outline Diagram

13.1 224-Pin FSBGAC

All dimensions are in millimeters.



14 Change History

Table 14.1-1 Change History

Date	Description
01/30/04	1. Created document.

Notes

Bluetooth is a registered trademark of Bluetooth SIG, Inc.
Wi-Fi is a registered trademark of Wireless Ethernet Compatibility Alliance Inc.
ARM and *PrimeCell* are registered trademarks of Advanced RISC Machines Limited.
Motorola is a registered trademark of Motorola, Inc.
Texas Instruments is a registered trademark and *SSI* is a trademark of Texas Instruments Inc.
MICROWIRE is a registered trademark of Advanced Interconnection Technology, Inc.
Philips is a registered trademark of Philips Electronics N.V.
Epson is a registered trademark of Seiko Epson Corporation.
Micrel is a trademark of Micrel Semiconductor, Inc.
IEEE is a registered trademark of The Institute of Electrical and Electronics Engineers, Inc.
National Semiconductor is a registered trademark of National Semiconductor Corporation.

For additional information, contact your Agere Systems Account Manager or the following:
INTERNET: **http://www.agere.com**
E-MAIL: **docmaster@agere.com**
N. AMERICA: Agere Systems Inc., Lehigh Valley Central Campus, Room 10A-301C, 1110 American Parkway NE, Allentown, PA 18109-9138
1-800-372-2447, FAX 610-712-4106 (In CANADA: **1-800-553-2448**, FAX 610-712-4106)
ASIA: Agere Systems Hong Kong Ltd., Suites 3201 & 3210-12, 32/F, Tower 2, The Gateway, Harbour City, Kowloon
Tel. (852) 3129-2000, FAX (852) 3129-2020
CHINA: **(86) 21-5047-1212** (Shanghai), **(86) 755-25881122** (Shenzhen)
JAPAN: **(81) 3-5421-1600** (Tokyo), KOREA: **(82) 2-767-1850** (Seoul), SINGAPORE: **(65) 6778-8833**, TAIWAN: **(886) 2-2725-5858** (Taipei)
EUROPE: **Tel. (44) 1344 296 400**

January 30, 2004
DS03-156IPT

agere systems