Intel® StrongARM® SA-1100 Microprocessor

Developer's Manual

April 1999

Order Number: 278088-003

Contents

Figures

Tables

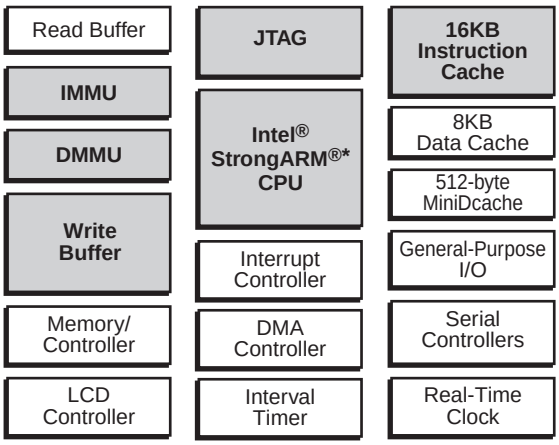
Introduction 1

1.1 Intel® StrongARM® SA-1100 Microprocessor

The Intel® StrongARM® SA-1100 Microprocessor (SA-1100) is the second member of the StrongARM® family. It is a highly integrated communications microcontroller that incorporates a 32-bit StrongARM® RISC processor core, system support logic, multiple communication channels, an LCD controller, a PCMCIA controller, and general-purpose I/O ports.

As does the Intel® StrongARM® SA-110 Microprocessor (SA-110), the first member of the StrongARM family, the SA-1100 provides power efficiency, low cost, and high performance. Figure 1-1 shows the features of the SA-1100. The shaded boxes are features that have carried over with few or no changes from the SA-110. The nonshaded boxes are new or updated features for the SA-1100.

Figure 1-1. SA-1100 Features



* StrongARM is a registered trademark of ARM Limited.

A6830-01

Table 1-1. Features of the SA-1100 CPU for AA and EA Parts

- High Performance
  - — 150 Dhrystone 2.1 MIPS @ 133 MHz
  - — 220 Dhrystone 2.1 MIPS @ 190 MHz
- Low power (normal mode)†
  - — <230 mW @1.5 V/133 MHz
  - — <330 mW @ 1.5 V/200 MHz
- Integrated clock generation
  - — Internal phase-locked loop (PLL)
  - — 3.686 MHz oscillator
  - — 32.768 kHz oscillator
- Power-management features
  - — Normal (full-on) mode
  - — Idle (power-down) mode
  - — Sleep (power-down) mode
- Big and little endian operating modes
- 3.3 V I/O interface
- 208-pin thin quad flat pack (LQFP)††
- 256 mini-ball grid array (mBGA)
- 32-way set-associative caches
  - — 16 Kbyte instruction cache
  - — 8 Kbyte write-back data cache
- 32-entry memory-management units
  - — Maps 4 Kbyte, 8 Kbyte, or 1 Mbyte
- Write buffer
  - — 8-entry, between 1 and 16 bytes each
- Read buffer
  - — 4-entry, 1, 4, or 8 words
- Memory bus
  - — Interfaces to ROM, Flash, SRAM, and DRAM
  - — Supports two PCMCIA sockets

† Power dissipation, particularly in idle mode, is strongly dependent on the details of the system design.

†† Package nomenclature has been modified due to industry standardization of packages. LQFP is 1.4mm thick, thin quad flat pack. Please note that no modification has been made to the package itself.

Table 1-2. Features of the SA-1100 CPU for CA and DA Parts

- High Performance
  - — 180 Dhrystone 2.1 MIPS @ 160 MHZ
  - — 250 Dhrystone 2.1 MIPS @ 220 MHz
- Low power (normal mode)†
  - — <430 mW @ 2.0-V/160-MHz
  - — <550 mW @ 2.0-V/220-MHz
- Integrated clock generation
  - — Internal phase-locked loop (PLL)
  - — 3.686-MHz oscillator
  - — 32.768-kHz oscillator
- Big and little endian operating modes
- 3.3-V I/O interface
- 208-pin thin quad flat pack (LQFP)††
- 256 mini-ball grid array (mBGA)
- 32-way set-associative caches
  - — 16 Kbyte instruction cache
  - — 8 Kbyte write-back data cache
- 32-entry memory-management units
  - — Maps 4 Kbyte, 8 Kbyte, or 1 Mbyte
- Write buffer
  - — 8-entry, between 1 and 16 bytes each
- Read buffer
  - — 4-entry, 1, 4, or 8 words
- Memory bus
  - — Interfaces to ROM, Flash, SRAM, and DRAM
  - — Supports two PCMCIA sockets

† Power dissipation, particularly in idle mode, is strongly dependent on the details of the system design.

†† Package nomenclature has been modified due to industry standardization of packages. LQFP is 1.4mm thick, thin quad flat pack. Please note that no modification has been made to the package itself.

Table 1-3. Changes to the SA-1100 Core from the SA-110

- Data cache reduced from 16 Kbyte to 8 Kbyte
- Interrupt vector address adjust capability
- Read buffer (nonblocking)
- Minicache for alternate data caching
- Hardware breakpoints
- Memory-management unit (MMU) enhancements
- Process ID mapping

Table 1-4. Additional Features Built into SA-1100 Chipset

- Memory controller supporting ROM, Flash, EDO, standard DRAM, and SRAM
- LCD controller
  - 1-, 2-, or 4-bit gray-scale levels
  - 8-, 12-, or 16-bit color levels
- Serial communications module supporting SDLC
- 230-Kbps UART
- Touch-screen, audio, telecom port
- Infrared data (IrDA) serial port
  - 115 Kbps, 4 Mbps
- Six-channel DMA controller
- Integrated two-slot PCMCIA controller
- Twenty-eight general-purpose I/O ports
- Real-time clock with interrupt capability
- On-chip oscillators for clock sources
- Interrupt controller
- Power-management features
  - Normal (full-on) mode
  - Idle (power-down) mode
  - Sleep (power-down) mode
- Four general-purpose interruptible timers
- 12-Mbps USB device controller
- Synchronous serial port (UCB1100, UCB1200, SPI, TI, Wire)

Table 1-5. SA-1100 Tool Chains and Operating Systems

Manufacturer	Operating System	Tool Chain
Accelerated Technology Incorporated	Nucleus PLUS*	Nucleus UDB*; ARM SDT* (ARM Limited); C, C++, MULTI* (Green Hills Software, Inc.); C/C++* compilers (MetaWare Incorporated)
Express Logic, Inc.	ThreadX*	ARM SDT* (ARM Limited); C, C++, MULTI* (Green Hills Software, Inc.); C/C++* compilers (MetaWare Incorporated)
Freewear	µCOS*	ARM SDT* (ARM Limited); C, C++, MULTI* (Green Hills Software, Inc.); C/C++* compilers (MetaWare Incorporated)
Integrated Systems, Inc.	pSOS*	pRISM+*; ARM SDT* (ARM Limited); C, C++, MULTI* (Green Hills Software, Inc.); C/C++* compilers (MetaWare Incorporated)
JMI Software Systems, Inc.	C EXECUTIVE*	ARM SDT* (ARM Limited); C, C++, MULTI* (Green Hills Software, Inc.); C/C++* compilers (MetaWare Incorporated)
KADAK Products Ltd.	AMX*	ARM SDT* (ARM Limited); C, C++, MULTI* (Green Hills Software, Inc.); C/C++* compilers (MetaWare Incorporated)
Lucent Technologies, Inc.	Inferno*	Inferno Toolkit*
Microsoft Corporation	Windows CE*	Platform Builder*
Microware Systems Corporation	OS-9*	FasTrac*
Precise Software Technologies, Inc.	Precise/MQX*	ARM SDT* (ARM Limited); C, C++, MULTI* (Green Hills Software, Inc.); C/C++* compilers (MetaWare Incorporated)
Sun Microsystems, Inc.	Chorus OS*	Sun Embedded Workshop Toolset*
Sun Microsystems, Inc.	Java OS for Consumer*	Java Development Tools*
Symbian	EPOC32*	C++ SDK
Wind River Systems, Inc.	VxWorks/Tornado*	Tornado*

1.2 Overview

The SA-1100 Microprocessor (SA-1100) is a general-purpose, 32-bit RISC microprocessor with a 16 Kbyte instruction cache, an 8 Kbyte write-back data cache, a minicache, a write buffer, a read buffer, and a memory management unit (MMU) combined in a single chip. The SA-1100 is software compatible with the ARM™ V4 architecture processor family and can be used with ARM support chips such as I/O, memory, and video. The core of the SA-1100 is derived from the core of the SA-110 Microprocessor (SA-110), with the following changes:

- Reduction in size of the data cache from 16 Kbyte to 8 Kbyte
- Addition of a 512-byte mini data cache that allocates data based on MMU settings
- Addition of debug support in the form of address and data breakpoints
- Addition of a four-entry read buffer to facilitate software-controlled data prefetching
- Addition of vector address adjust capability
- Addition of a process ID register

The logic outside the core and caches is grouped into the following three modules:

- Memory and PCMCIA control module (MPCM)
  – Memory interface supporting ROM, Flash, DRAM, SRAM and PCMCIA control signals
- System control module (SCM)
  – Twenty-eight general-purpose interruptible I/O ports
  – Real-time clock, watchdog, and interval timers
  – Power management controller
  – Interrupt controller
  – Reset controller
  – Two on-chip oscillators for connection to 3.686 MHz and 32.768 kHz crystals
- Peripheral control module (PCM)
  – Six-channel DMA controller
  – Gray/color, active/passive LCD controller
  – 230 Kbps SDLC controller
  – 16550-compatible UART
  – IrDA serial port (115 Kbps, 4 Mbps)
  – Synchronous serial port (UCB1100, UCB1200, SPI, TI, µWire)
  – Universal serial bus (USB) device controller

The instruction set comprises eight basic instruction types:

- Two make use of on-chip arithmetic logic unit, barrel shifter, and multiplier to perform high-speed operations on data in a bank of 16 logical registers (31 physical registers), each 32 bits wide.
- Three classes of instructions control data transfer between memory and the registers: one optimized for flexibility of addressing, one for rapid context switching, and one for swapping data.
- Two instructions control the flow and privilege level of execution.
- One class is used to access the privileged state of the CPU.

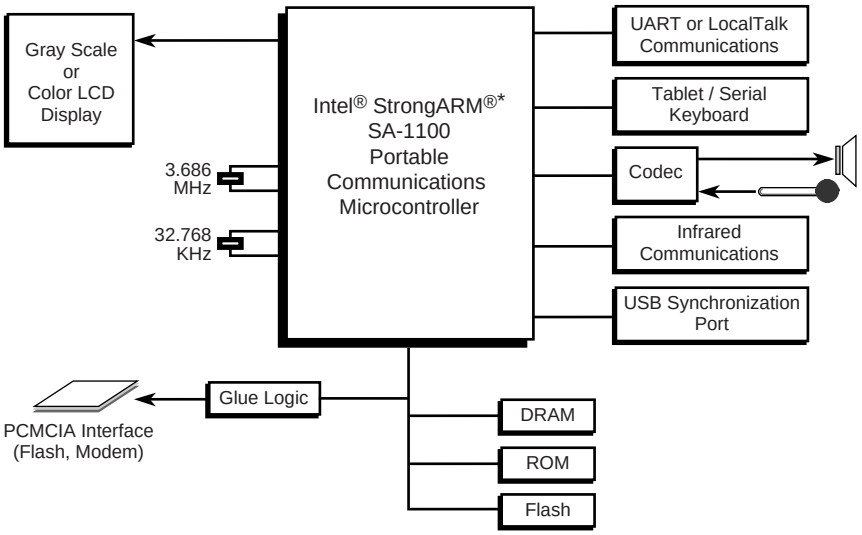
The ARM instruction set is a good target for compilers of many different high-level languages. Where required for critical code segments, assembly code programming is also straightforward, unlike some RISC processors that need sophisticated compiler technology to manage complicated instruction interdependencies.

The SA-1100 is a static part and has been designed to run at a reduced voltage to minimize its power requirements. This makes it a good choice for portable applications where both of these features are essential.

1.3 Example System

Figure 1-2 shows how the SA-1100 can be used in a hand-held computing device.

Figure 1-2. SA-1100 Example System



* StrongARM is a registered trademark of ARM Limited.

A6870-01

1.4 ARM™ Architecture

The SA-1100 implements the ARM V4 architecture as defined in the *ARM Architecture Reference*, 28-July-1995, with the following options:

1.4.1 26-Bit Mode

The SA-1100 supports 26-bit mode but all exceptions are initiated in 32-bit mode. The P and D bits do not affect the operation of SA-1100; they are always read as ones and writes to them are ignored.

1.4.2 Coprocessors

The SA-1100 supports MCR and MRC access to coprocessor number 15. These instructions are used to access the memory-management, configuration, and cache control registers. In addition, coprocessor 15 provides control for read buffer fills and flushes, and hardware breakpoints. All other coprocessor instructions cause an undefined instruction exception. No support for external coprocessors is provided.

1.4.3 Memory Management

Memory management exceptions preserve the base address registers so that no code is required to restore state. Separate translation lookaside buffers (TLBs) are implemented for the instruction and data streams. Each TLB has 32 entries that can each map a segment, a large page, or a small page. The TLB replacement algorithm is round robin. The data TLBs support both the flush-all and flush-single-entry operations, while the instruction TLBs support only the flush-all operation.

1.4.4 Instruction Cache

The SA-1100 has a 16 Kbyte instruction cache (Icache) with 32-byte blocks and 32-way associativity. The cache supports the flush-all function. Replacement is round robin within a set. The Icache can be enabled while memory management is disabled. When memory management is disabled, all memory is considered cacheable by the Icache.

1.4.5 Data Cache

The SA-1100 has an 8 Kbyte data cache (Dcache) with 32-byte blocks and 32-way associativity. The cache supports the flush-all, flush-entry, and copyback-entry functions. The copyback-all function is not supported in hardware. This function can be provided by software. The cache is read allocate with round-robin replacement.

The Dcache has been augmented with a 16-entry, two-way set associative minicache that allocates when the MMU **b** and **c** bits are 0 and 1, respectively. This cache is accessed in parallel with the main Dcache. Replacement victims in this cache are replaced based on a least-recently-used (LRU) algorithm. This cache is useful for applications that access large data structures and would normally thrash the main Dcache. Instead, these data structures can be mapped so that they allocate into the minicache and only replace data from the same structure.

1.4.6 Write Buffer

The SA-1100 has an eight-entry write buffer with each entry able to contain 1 to 16 bytes. A drain write buffer operation is supported.

1.4.7 Read Buffer

The SA-1100 has a four-entry read buffer capable of loading 1, 4, or 8 words of data per entry. This facility permits software to preload data into the buffer for use at a later time without blocking the operation of the processor. Software can flush either a single entry or the entire buffer (four entries). The read buffer is controlled through system control coprocessor 15 and can be enabled for use in user mode.

Functional Description 2

This chapter provides a functional description of the Intel® StrongARM® SA-1100 Microprocessor (SA-1100). It describes the basic building blocks within the processor, lists and describes the pins, and explains the memory map.

2.1 Block Diagram

The SA-1100 consists of the following functional blocks:

- **Processor**

The processor is the ARM™ SA-1 core with a 16 Kbyte instruction and 8 Kbyte data cache (Dcache). The instruction (I) and data (D) streams are translated through independent memory-management units (MMUs). Stores are made using a four-line write buffer. The performance of specialized load routines is enhanced with the four-entry read buffer that can be used to prefetch data for use at a later time. A 16-entry minicache provides a smaller and logically separate data cache that can be used to enhance caching performance when dealing with large data structures.

- **Memory and PCMCIA controller**

The memory and PCMCIA control module (MPCM) supports four banks of standard or EDO DRAM on a 32-bit data width. ROM (standard and burst), Flash memory, and SRAM are also supported. ROM and Flash can be either 16 or 32 bits wide. SRAM width is limited to 32 bits. Expansion devices are supported through PCMCIA control signals that share the memory bus data and address lines to complete the card interface. Some external glue logic (buffers and transceivers) is necessary to implement the interface. Control is provided to permit two card slots with hot-swap capability.

- **Peripherals**

The peripheral control module (PCM) contains a number of serial control devices, an LCD controller as well as a six-channel DMA controller to provide service to these devices:

  – An LCD controller with support for passive or active displays
  – A universal serial bus (USB) endpoint controller
  – An SDLC communications controller
  – A serial controller with supporting 115 Kbps and 4 Mbps IrDA protocols
  – A 16550-like UART supporting 230 Kbps
  – A CODEC interface supporting SPI, µWire, TI, UCB1100, and UCB1200

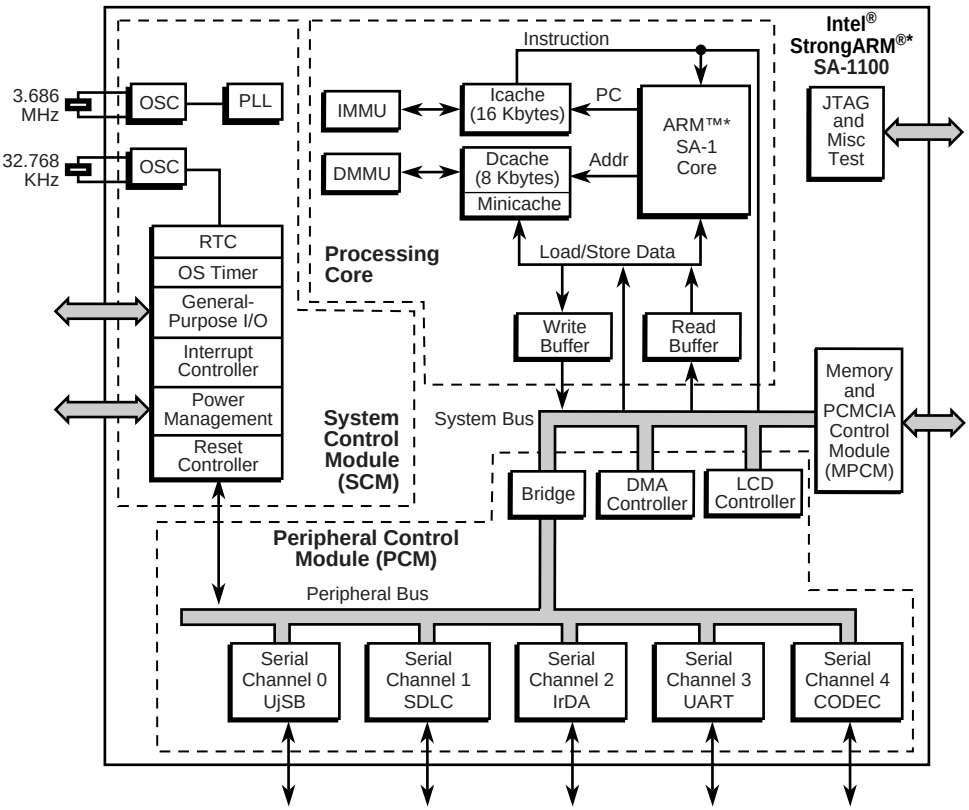
- **General system control functions**

The system control module (SCM) is also connected to the peripheral bus. It contains five blocks used for general system functions:

  – A real-time clock (RTC) clocked from an independent 32.768 kHz oscillator
  – An operating system timer (OST) for general system timer functions as well as a watchdog mode
  – Twenty-eight general-purpose I/Os (GPIO)
  – An interrupt controller
  – A power-management controller that handles the transitions in and out of sleep and idle modes
  – A reset controller that handles the various reset sources on the processor

Figure 2-1 shows the functional blocks contained in the SA-1100 integrated processor. Figure 2-2 is a functional diagram of the SA-1100.

Figure 2-1. SA-1100 Block Diagram

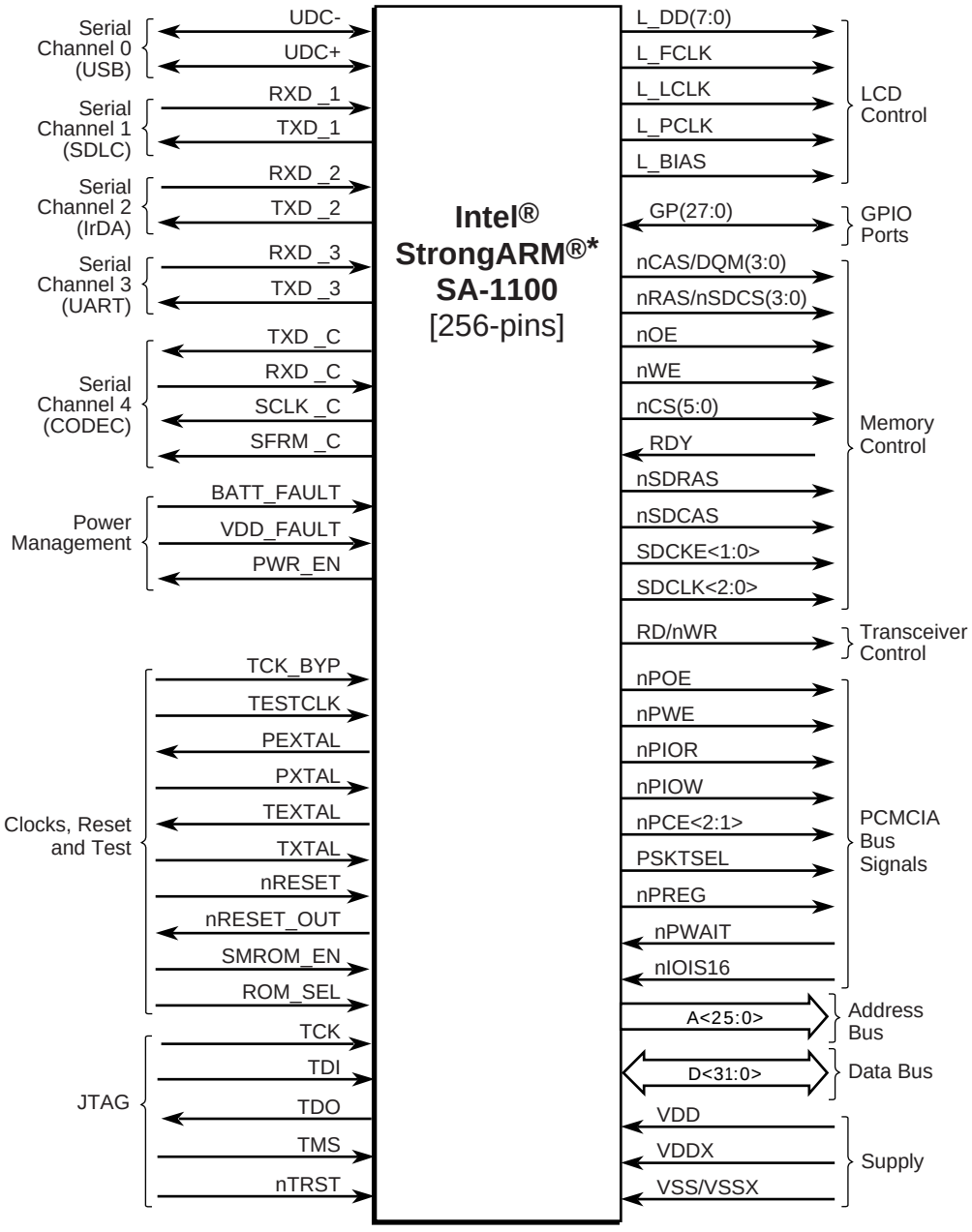


* ARM is a trademark and StrongARM is a registered trademark of ARM Limited.

A6832-01

2.2 Inputs/Outputs

Figure 2-2. SA-1100 Functional Diagram



* StrongARM is a registered trademark of ARM Limited.

A6849-01

2.3 Signal Description

The following table describes the signals.

Key to Signal Types: **n** – Active low signal
IC – Input, CMOS threshold
ICOCZ – Input, CMOS threshold, output CMOS levels, tristatable
OCZ – Output, CMOS levels, tristatable

Table 2-1. Signal Descriptions (Sheet 1 of 3)

Name	Type	Description
A<25:0>	OCZ	Memory address bus. This bus signals the address requested for memory accesses. Bits 21..10 carry the 12-bit DRAM address, the static memory devices, and the expansion bus receive address bits 25..0.
D<31:0>	ICOCZ	Memory data bus.
nCS<3:0>	OCZ	Static chip selects. These signals are chip selects to static memory devices such as ROM and Flash. They are individually programmable in the memory configuration registers.
nOE	OCZ	Memory output enable. This signal should be connected to the output enables to begin driving data onto the data bus.
nWE	OCZ	DRAM write enable. This signal should be connected to the DRAM write enables to perform writes. This signal is used in conjunction with CAS<3:0> to perform byte writes.
nRAS<3:0>	OCZ	DRAM RAS. These signals should be connected to the DRAM row address strobe (RAS) pin.
nCAS<3:0>	OCZ	DRAM CAS. These signals should be connected to the DRAM column address strobe (CAS) pins.
nPOE	OCZ	PCMCIA output enable. This PCMCIA signal is an output and is used to perform reads from memory and attribute space.
nPWE	OCZ	PCMCIA write enable. This signal is an output and is used to perform writes to memory and attribute space.
nPIOW	OCZ	PCMCIA I/O write. This signal is an output and is used to perform write transactions to the PCMCIA I/O space.
nPIOR	OCZ	PCMCIA I/O read. This signal is an output and is used to perform read transactions from the PCMCIA I/O space.
nPCE<2:1>	OCZ	PCMCIA card enable. These signals are output and are used to select a PCMCIA card. Bit one enables the high-byte lane and bit zero enables the low-byte lane.
nIOIS16	IC	I/O Select 16. This signal is an input and is an acknowledgment from the PCMCIA card that the current address is a valid 16-bit wide I/O address.
nPWAIT	IC	PCMCIA wait. This signal is an input and is driven low by the PCMCIA card to extend the length of the transfers to/from the SA-1100.
PSKTSEL	OCZ	PCMCIA socket select. This signal is an output and is used by external steering logic to route control, address, and data signals to one of the PCMCIA sockets. When PSKTSEL is low, socket zero is selected. When PSKTSEL is high, socket one is selected. This signal has the same timing as the address lines.
nPREG	OCZ	PCMCIA register select. This signal is an output and indicates that, on a memory transaction, the target address is attribute space. This signal has the same timing as address.
L_DD<7:0>	OCZ	LCD controller display data.

Table 2-1. Signal Descriptions (Sheet 2 of 3)

Name	Type	Description
L_FCLK	OCZ	LCD frame clock.
L_LCLK	OCZ	LCD line clock.
L_PCLK	OCZ	LCD pixel clock.
L_BIAS	OCZ	LCD ac bias drive.
TXD_C	OCZ	CODEC transmit.
RXD_C	IC	CODEC receive.
SCLK_C	OCZ	CODEC clock.
SFRM_C	OCZ	CODEC frame signal.
UDC+	OCZ	Serial port zero transmit pin (UDC).
UDC-	IC	Serial port zero receive pin (UDC).
TXD_1	OCZ	Serial port one transmit pin (SDLC).
RXD_1	IC	Serial port one receive pin (SDLC).
TXD_2	OCZ	Serial port two transmit pin (IrDA).
RXD_2	IC	Serial port two receive pin (IrDA).
TXD_3	OCZ	Serial port three transmit pin (UART).
RXD_3	IC	Serial port three receive pin (UART).
GP<27:0>	ICOCZ	General-purpose input output.
ROM_SEL	IC	ROM select. This pin is used to configure the ROM width. It is either grounded or pulled high. If ROM_SEL is grounded, the ROM width is 16 bits. If ROM_SEL is pulled up, the ROM width is 32 bits.
PXTAL	IC	Input connection for 3.686-MHz crystal.
PEXTAL	OCZ	Output connection for 3.686-MHz crystal.
TXTAL	IC	Input connection for 32.768-kHz crystal.
TEXTAL	OCZ	Output connection for 32.768-kHz crystal.
PWR_EN	OCZ	Power enable. Active high. **PWR_EN** enables the external power supply. Negating it signals the power supply that the system is going into sleep mode and that the **VDD** power supply should be removed.
BATT_FAULT	IC	Battery fault. Signals the SA-1100 that the main power source is going away (battery is low or has been removed from the system). The assertion of BATT_FAULT causes the SA-1100 to enter sleep mode. The SA-1100 will not recognize a wake-up event while this signal is asserted.
VDD_FAULT	IC	VDD fault. Signals the SA-1100 that the main power supply is going out of regulation (shorted card is inserted). VDD_FAULT will cause the SA-1100 to enter sleep mode. VDD_FAULT is ignored after a wake-up event until the poser supply timer completes (approximately 10 ms).
nRESET	IC	Hard reset. This active low signal is a level-sensitive input used to start the processor from a known address. A low level will cause the current instruction to terminate abnormally, and the on-chip caches, MMU, and write buffer to be disabled. When nRESET is driven high, the processor will restart from address 0. nRESET must remain low until the power supply is stable and the internal 3.686-MHz oscillator has come up to speed. While nRESET is low, the processor will perform idle cycles.

Table 2-1. Signal Descriptions (Sheet 3 of 3)

Name	Type	Description
nRESET_OUT	OCZ	Reset out. This signal is asserted when nRESET is asserted and deasserts when the processor has completed resetting. nRESET_OUT is also asserted for "soft" reset events (sleep and watchdog).
nTRST	IC	Test interface reset. Note this pin has an internal pull-down resistor and must be driven high to enable the JTAG circuitry. If left unconnected, this pin is pulled low and disables JTAG operation.
TDI	IC	JTAG test interface data input. Note this pin has an internal pull-up resistor.
TDO	OCZ	JTAG test interface data output. Note this pin does *not* have an internal pull-up resistor.
TMS	IC	JTAG test interface mode select. Note this pin has an internal pull-up resistor.
TCK	IC	JTAG test interface reference clock. This times all the transfers on the JTAG test interface. Note this pin has an internal pull-down resistor.
TCK_BYP	IC	Test clock PLL bypass. When TCK_BYP is high, the TESTCLK is used as the core clock in place of the PLL clock; when low, the internal PLL output is used. This signal has no relation to the JTAG TCK pin.
TESTCLK	IC	Test clock. TESTCLK is used to provide the core clock when TCK_BYP is high. It should be tied low if TCK_BYP is low. This pin should be used for test purposes only. An end user should ground this pin.
VDD	—	Positive supply for the core. Nine pins are allocated to this supply; eight pins are labeled VDD. The ninth pin, labeled VDDP is dedicated to the PLL supply and should be tied directly to the VDD power plane with the other eight VDD pins.
VDDX	—	Positive supply for the pins. Twenty pins are allocated to VDDX, labeled VDDX1, VDDX2 and VDDX3. All of these pins should be tied directly to the VDDX power plane.
VSS	—	Ground supply. Nine pins are allocated to VSS, including one for the PLL.
VSSX	—	Ground supply for the I/O pins. Eighteen pins are allocated to VSSX.

2.4 Memory Map

Figure 2-3 shows the SA-1100 memory map. The map is divided into four main partitions of 1 Gbyte each.

The bottom partition is dedicated to static memory devices (ROM, SRAM, and Flash) and to the PCMCIA expansion bus area. It occupies addresses 0h0000 0000 through 0h3FFF FFFF. This space is divided into four 128 Mbyte blocks for static memory devices and two 256 Mbyte blocks for PCMCIA.

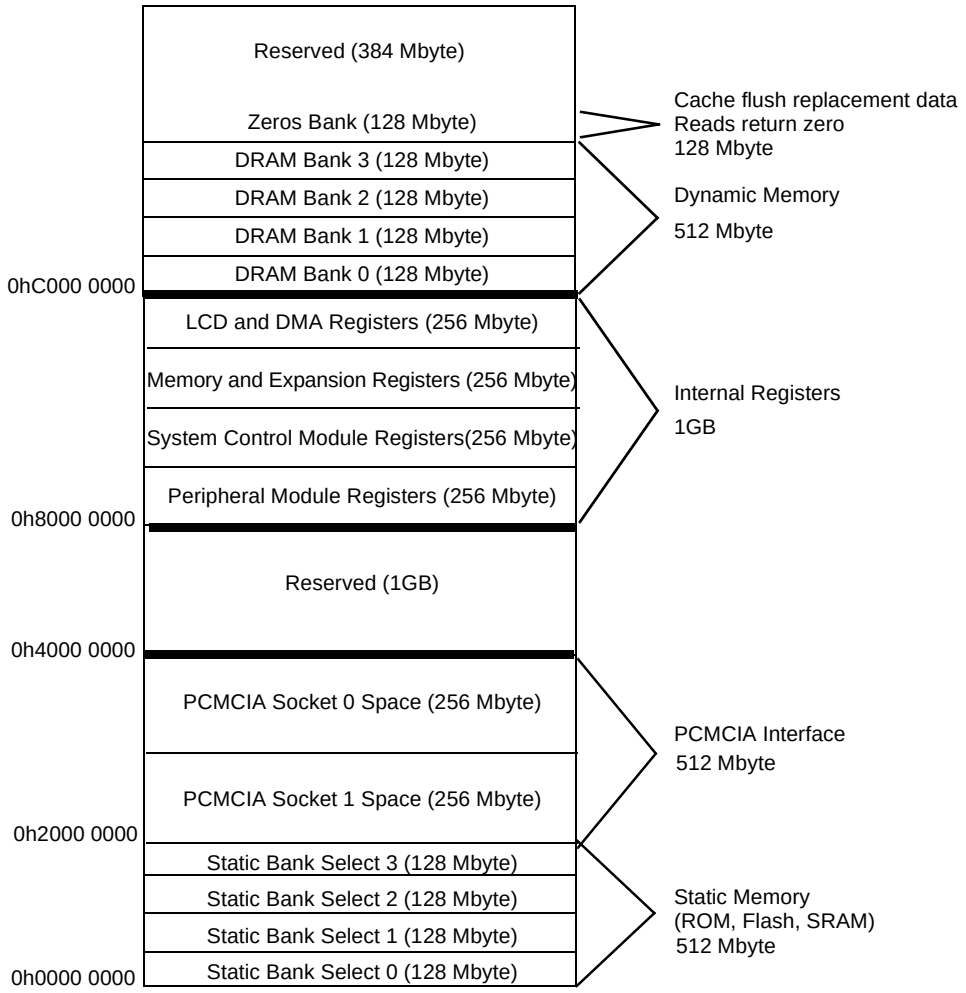
The static memory space is intended for ROM, SRAM, and Flash memory. The bottom partition (at 0h0000 0000) is assumed to be ROM at boot time. The width of the boot ROM is determined by the state of the ROMSEL pin. The PCMCIA interface is divided into Socket 0 and Socket 1 space. These partitions are further subdivided into I/O, memory and attribute space.

The next partition (0h4000 0000 to 0h7FFF FFFF) is reserved. Accessing this reserved space results in a data abort exception.

The third partition (0h8000 0000 to 0hBFFF FFFF) contains all on-chip registers (except those specified by the ARM V4 architecture). This block is further subdivided into four blocks of 256 Mbyte each. They contain control registers for the major functional blocks within the processor (MECM, SCM, PCM). The LCD and DMA controllers are separate from the rest of the PCM and occupy the top 256 Mbyte partition.

The fourth partition (0hC000 0000 to 0hFFFF FFFF) contains DRAM memory. The bank sizes for DRAM are fixed at 128 Mbyte each. With multiple banks implemented, there probably will be gaps in the map that should be mapped through the memory-management unit. The next 128 Mbyte block in this partition is mapped within the memory controller and returns zeros when read. This function is intended to facilitate rapid cache flushing by not requiring an external memory access to load data into the cache. This space is burstable. Writes to this space have no effect. The top 384 Mbyte of this partition is reserved. Accessing this space causes a data abort exception.

Figure 2-3. SA-1100 Memory Map



ARM™ Implementation Options 3

The following sections describe ARM™ architecture options that are implemented by the Intel® StrongARM® SA-1100 Microprocessor (SA-1100).

3.1 Big and Little Endian

The big endian bit in the control register sets whether the SA-1100 treats words stored in memory as being stored in big endian or little endian format. Memory is viewed as a linear collection of bytes numbered upwards from 0. Bytes 0 to 3 hold the first stored word, bytes 4 to 7 hold the second, and so on.

In the little endian scheme, the lowest numbered byte in a word is considered to be the least significant byte of the word and the highest numbered byte is the most significant. Byte 0 of the memory system should be connected to data lines 7 through 0 (D<7:0>) in this scheme.

In the big endian scheme, the most significant byte of a word is stored at the lowest numbered byte and the least significant byte is stored at the highest numbered byte. Therefore, byte 0 of the memory system should be connected to data lines 31 through 24 (D<31:24>).

The state of the big endian bit changes the location of the bytes only within a 32-bit word. The accessed bytes are changed for the load byte, store byte, load halfword, and store halfword instructions only. Instruction fetches and word load and stores are not changed by the state of the big endian bit.

These conventions are identical to those of the SA-110. In addition, the SA-1100 DMA controller is programmable by channel as to the endian format of the transfer. For DMA transfers, all memory accesses are words. Then the data is buffered and transferred to/from the device as halfwords or bytes. When the words are assembled or disassembled, the endian format of the channel is observed. For details on how DMA data is transferred relative to the endian format of the channel, see the Section 11.6, “DMA Controller” on page 11-7 in Chapter 11, “Peripheral Control Module”.

3.2 Exceptions

Exceptions arise whenever there is a need for the normal flow of program execution to be broken; for example, so that the processor can be diverted to handle an interrupt from a peripheral. The processor state just prior to handling the exception must be preserved so that the original program resumes when the exception routine has completed. Many exceptions may arise at the same time.

The SA-1100 handles exceptions by making use of banked registers to save state. The contents of PC and CPSR are copied into the appropriate R14 and SPSR, and the PC and mode bits in the CPSR bits are forced to a value that depends on the exception. Interrupt disable flags are set where required to prevent otherwise unmanageable nestings of exceptions. In the case of a reentrant interrupt handler, R14 and the SPSR should be saved onto a stack in main memory before reenabling the interrupt; when transferring the SPSR register to and from a stack, it is important to transfer the whole 32-bit value, and not just the flag or control fields. When multiple exceptions

arise simultaneously, a fixed priority determines the order in which they are handled. The priorities are listed later in this chapter. Most exceptions are fully defined in the *ARM Architectural Reference*. The following sections specify the exceptions where the SA-1100 implementation differs from the *ARM Architectural Reference*.

SA-1100 initiates all exceptions in 32-bit mode. When an exception occurs while running in 26-bit mode, the SA-1100 saves only the PC in R14 and the CPSR in the SPSR of the exception mode. The 32-bit handler must merge the condition codes, the interrupt enables, and the mode from the SPSR into R14 if a handler is to run in 26-bit mode.

3.2.1 Power-Up Reset

When the nRESET signal is low, SA-1100 stops executing instructions, asserts the nRESET_OUT pin, and then performs idle cycles on the bus.

When nRESET is high again, SA-1100 does the following:

1. Overwrites R14_svc and SPSR_svc by copying the current values of the PC and CPSR into them. The values of the saved PC and CPSR are not defined.
2. Forces M<4:0>=10011 (32-bit supervisor mode) and sets the I and F bits in the CPSR.
3. Forces the PC to fetch the next instruction from address 0x0000 0000.
4. Based on the state of the ROM_SEL pin, fetches this first instruction from either 16-bit (ROM_SEL low) or 32-bit (ROM_SEL high) space. The SA-1100 memory controller assembles the data into words in the case of a 16-bit wide ROM.

At the end of the reset sequence, the MMU, Icache, Dcache, and write buffer are disabled. Alignment faults are also disabled, and little-endian mode is enabled. During power-up, nRESET must be negated no earlier than 150 milliseconds after VDD and VDDx are stable to allow the internal 3.686-MHz oscillator to stabilize. After the negation of nRESET, the PLL begins its internally timed locking sequence. Note that the assertion of nRESET is destructive because the state of the real-time clock and the contents of DRAM are lost.

The SA-1100 has three types of reset. See Section 16.2, “Reset” on page 16-2 in the Boundary-Scan Test Interface for details.

3.2.2 ROM Size Select

The ROM width may be selected using the ROM_SEL pin. This pin is sampled during the assertion of nRESET. The value is stored in the memory controller for use during ROM accesses. If this signal is high during RESET, then the ROM is selected to be 32 bits wide. If it is low during RESET, then the ROM width is 16 bits. There is no provision for 8-bit ROMs in the SA-1100.

3.2.3 Abort

An abort can be signalled by the internal memory-management unit, through a data breakpoint, or by a reference to reserved memory. An abort indicates that the current memory access cannot be completed or that a prespecified breakpoint address and (optionally) data pattern has been reached. For instance, in a virtual memory system, the data corresponding to the current address may have been moved out of memory onto a disk, and considerable processor activity may be required to recover the data before the access can be performed successfully. The SA-1100 checks for an abort during memory access cycles. When aborted, the SA-1100 responds in one of two ways:

1. If the abort occurred during an instruction prefetch (a *prefetch abort*), the prefetched instruction is marked as invalid but the abort exception does not occur immediately. If the instruction is not executed, for example, as a result of a branch being taken while it is in the pipeline, no abort will occur. An abort will take place if the instruction reaches the head of the pipeline and is about to be executed.
2. If the abort occurred during a data access (a *data abort*), the action depends on the instruction type.
    a. Single data transfer instructions (LDR, STR) will abort with no registers modified.
    b. The swap instruction (SWP) is aborted as though it had not executed, though externally the read access may take place.
    c. Block data transfer instructions (LDM, STM) abort on the first access that cannot complete. If write-back is set, the base is **NOT** updated. If the instruction would normally have overwritten the base with data (for example, an LDM instruction with the base in the transfer list), the original value in the base register is restored.

When either a prefetch or data abort occurs, the SA-1100 performs the following:

1. Saves the address of the aborted instruction plus 4 (for prefetch aborts) or 8 (for data aborts) in R14_abt; saves CPSR in SPSR_abt.
2. Forces M<4:0>=10111 (abort mode) and sets the I bit in the CPSR.
3. Forces the PC to fetch the next instruction from either address 0x0C (prefetch abort) or address 0x10 (data abort).

To return after fixing the reason for the abort, use SUBS PC,R14_abt,#4 (for a prefetch abort) or SUBS PC,R14_abt,#8 (for a data abort). This will restore both the PC and the CPSR, and retry the aborted instruction.

The abort mechanism allows a *demand paged virtual memory system* to be implemented when suitable memory management software is available. The processor is allowed to generate arbitrary addresses, and when the data at an address is unavailable, the MMU signals an abort. The processor traps into system software, which must work out the cause of the abort, make the requested data available, and retry the aborted instruction. The application program needs no knowledge of the amount of memory available to it, nor is its state in any way affected by the abort.

3.2.4 Vector Summary

Table 3-1 lists byte addresses, and they normally contain branch instructions pointing to the relevant routines. These addresses (except the reset vector) can be changed (to 0xFFFF xxxx) through the vector adjust facility (bit 13, register 1, coprocessor 15). The vector adjust is cleared at reset and cannot modify the reset vector.

Table 3-1. Vector Summary

Address	Exception	Mode on Entry
0x00000000	Reset	Supervisor
0x00000004	Undefined instruction	Undefined
0x00000008	Software interrupt	Supervisor
0x0000000C	Abort (prefetch)	Abort
0x00000010	Abort (data)	Abort
0x00000014	Not used	—
0x00000018	IRQ	IRQ
0x0000001C	FIQ	FIQ

3.2.5 Exception Priorities

When multiple exceptions arise at the same time, a fixed priority system determines the order in which they will be handled:

1. Reset (highest priority)
2. Data abort
3. FIQ
4. IRQ
5. Prefetch abort
6. Undefined instruction, software interrupt (lowest priority)

Note that not all exceptions can occur at once. Undefined instructions and software interrupts are mutually exclusive because they correspond to particular (nonoverlapping) decodings of the current instruction.

If a data abort occurs at the same time as a FIQ, and FIQs are enabled (that is, the F flag in the CPSR is clear), the SA-1100 will enter the data abort handler and then immediately proceed to the FIQ vector. A normal return from FIQ will cause the data abort handler to resume execution. Placing data abort at a higher priority than FIQ is necessary to ensure that the transfer error does not escape detection; the time for this exception entry should be added to worst-case FIQ latency calculations.

3.2.6 Interrupt Latencies and Enable Timing

The ability to recognize an IRQ or FIQ interrupt is, in part, determined by the I and F bits of the CPSR. To ensure that a pending interrupt is taken, an interrupt-enabling write to CPSR (msr instruction) must be separated from an interrupt-disabling write to the CPSR by at least two instructions.

3.3 Coprocessors

The SA-1100 has no external coprocessor bus, so it is not possible to add external coprocessors to this device.

The SA-1100 uses the internal coprocessor designated 15 for control of the on-chip MMU, caches, clocks, and breakpoints. Coprocessor 15 is also used for read-buffer fills and flushes. If a coprocessor other than 15 is used, then the SA-1100 will take the undefined instruction trap. The coprocessor load, store, and data operation instructions also take the undefined instruction trap. Permissions are set so that access to coprocessor 15 is privileged except where protection is programmable with respect to the read buffer operations.

Instruction Set 4

This section describes the instruction timing for the Intel® StrongARM® SA-1100 Microprocessor (SA-1100).

4.1 Instruction Set

The SA-1100 implements the ARM™ V4 architecture as defined in the *ARM Architecture Reference*, 28-July-1995, with previously noted options and additions.

4.2 Instruction Timings

Table 4-1 lists the instruction timing for the SA-1100. The result delay is the number of cycles that the next sequential instruction would stall if it used the result as an input. The issue cycles are the number of cycles that this instruction takes to issue. For most instructions, the result delay is zero and the issue cycles is one. For load and stores, the timing is for cache hits.

Table 4-1. Instruction Timings

Instruction Group	Result Delay	Issue Cycles
Data processing	0	1
Mul or Mul/Add giving 32-bit result	1..3	1
Mul or Mul/Add giving 64-bit result	1..3	2
Load single – write-back of base	0	1
Load single – load data zero extended	1	1
Load single – load data sign extended	2	1
Store single – write-back of base	0	1
Load multiple (delay for last register)	1	MAX (2, number of registers loaded)
Store multiple – write-back of base	0	MAX (2, number of registers loaded)
Branch or branch and link	0	1
MCR	2	1
MRC	1	1
MSR to control	0	3
MRS	0	1
Swap	2	2

Coprocessors 5

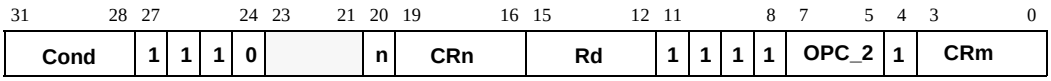
The operation and configuration of the Intel® StrongARM® SA-1100 Microprocessor (SA-1100) is controlled with coprocessor instructions, configuration pins, and memory-management page tables. The coprocessor 15 instructions manipulate on-chip registers that control the configuration of the cache, write buffer, MMU, read buffer, breakpoints, and other configuration options.

Note: The gray areas in the register and translation diagrams are reserved and should be programmed 0 for future compatibility.

5.1 Internal Coprocessor Instructions

The on-chip cache, MMU, write buffer, and read buffers are controlled using MRC instructions and MCR instructions. These operations to coprocessor 15 are allowed only in nonuser modes except when read-buffer operations are explicitly enabled. The undefined instruction trap is taken if accesses are attempted in user mode. Figure 5-1 shows the format of internal coprocessor instructions MRC and MCR.

Figure 5-1. Format of Internal Coprocessor Instructions MRC and MCR



Cond	ARM™ condition codes
n	1 MRC register read
	0 MCR register write
CRn	SA-1100 register
Rd	ARM register
OPC_2	Function bits for some MRC/MCR instructions
CRm	Function bits for some MRC/MCR instructions

5.2 Coprocessor 15 Definition

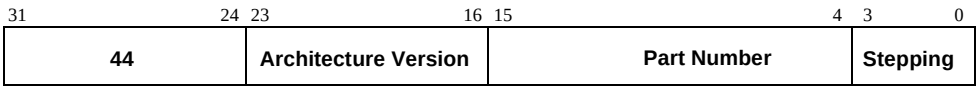
The SA-1100 coprocessor 15 contains registers that control the cache, MMU, and write buffer operation as well as some clocking functions. These registers are accessed using CPRT instructions to coprocessor 15 with the processor in any privileged mode. Only some of registers 0–15 are valid; the result of an access to an invalid register is unpredictable. Table 5-1 lists the coprocessor 15 control registers.

Table 5-1. Cache and MMU Control Registers (Coprocessor 15)

Register	Register Reads	Register Writes
0	ID	RESERVED
1	Control	Control
2	Translation table base	Translation table base
3	Domain access control	Domain access control
4	RESERVED	RESERVED
5	Fault status	Fault status
6	Fault address	Fault address
7	RESERVED	Cache operations
8	RESERVED	TLB operations
9	RESERVED	Read buffer operations
10..12	RESERVED	RESERVED
13	Read process ID (PID)	Write process ID (PID)
14	Read breakpoint	Write breakpoint
15	RESERVED	Test, clock, and idle

5.2.1 Register 0 – ID

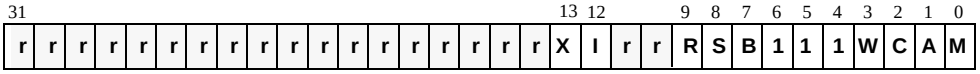
Register 0 is a read-only register that returns an architecture and implementation-defined identification for the device.identification for the device.



Architecture Version	ARM architecture version 01 = Version 4
Part Number	Part number A11 = SA1100
Stepping	Stepping revision of SA-1100 1 = B stepping 2 = C stepping 8 = D stepping 9 = E stepping 11 = G stepping

5.2.2 Register 1 – Control

Register 1 is a read/write register containing control bits. All writable bits in this register are forced low by reset. The shaded bits (also labeled r) are reserved and are not readable or writable.



M bit 0 **Enable/disable**
0 – On-chip memory-management unit disabled
1 – On-chip memory-management unit enabled

A bit 1 **Address fault enable/disable**
0 – Alignment fault disabled
1 – Alignment fault enabled

C bit 2 **Data cache enable/disable**
0 – Data cache disabled
1 – Data cache enabled

W bit 3 **Write buffer enable/disable**
0 – Write buffer disabled
1 – Write buffer enabled

B bit 7 **Big/little endian**
0 – Little endian operation
1 – Big endian operation

S bit 8 **System**
This bit selects the access checks performed by the memory-management unit. See the *ARM Architecture Reference* for more information.

R bit 9 **ROM**
This bit selects the access checks performed by the memory-management unit. See the *ARM Architecture Reference* for more information.

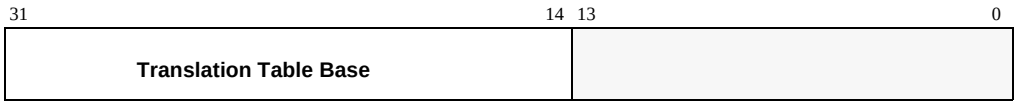
I bit 12 **Instruction cache enable/disable**
0 – Instruction cache disabled
1 – Instruction cache enabled

X bit 13 **Virtual interrupt vector adjust**
0 – Base address of interrupt vectors is 0h0000 0000
1 – Base address of interrupt vectors is 0hFFFF 0000

Bits 14..31 Unused.
Undefined on Read. Writes ignored.

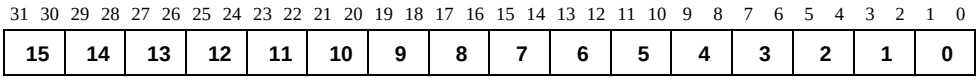
5.2.3 Register 2 – Translation Table Base

Register 2 is a read/write register that holds the base of the currently active level 1 page table. Bits <13:0> are undefined on read, ignored on write.



5.2.4 Register 3 – Domain Access Control

Register 3 is a read/write register that holds the current access control for domains 0 to 15. Refer to the *ARM Architecture Reference* for a description of the domain structure

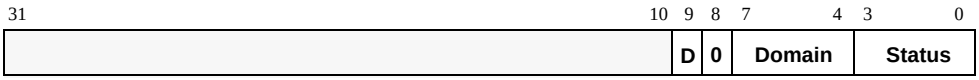


5.2.5 Register 4 – RESERVED

Register 4 is reserved. Accessing this register yields unpredictable results.

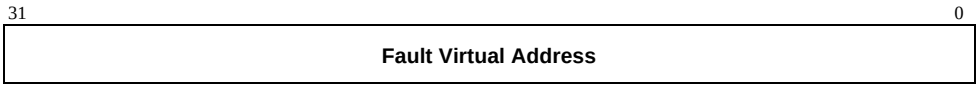
5.2.6 Register 5 – Fault Status

Reading register 5 returns the current contents of the fault status register (FSR). The FSR is written when a data memory fault occurs or can be written by an MCR to the FSR. It is not updated for a prefetch fault. See Chapter 7, “Memory-Management Unit (MMU)” for more details. Bits <31:10> are undefined on read, ignored on write. Bit 9 is set when a data breakpoint is taken and can be cleared by an MCR operation. Bit 8 is ignored on write and is always returned as zero. Refer to the *ARM Architecture Reference* for a description of the domain and status fields.



5.2.7 Register 6 – Fault Address

Reading register 6 returns the current contents of the fault address register (FAR). The FAR is written when a data memory fault occurs with the virtual address of the data fault or can be written by an MCR to the FAR.



5.2.8 Register 7 – Cache Control Operations

Register 7 is a write-only register. The CRm and OPC_2 fields are used to encode the cache control operations. Operation for all other values for OPC_2 and CRm is unpredictable.

Function	OPC_2	CRm	Data
Flush I+D	0b000	0b0111	Ignored
Flush I	0b000	0b0101	Ignored
Flush D	0b000	0b0110	Ignored
Flush D single entry	0b001	0b0110	Virtual address
Clean Dcache entry	0b001	0b1010	Virtual address
Drain write buffer	0b100	0b1010	Ignored

5.2.9 Register 8 – TLB Operations

Register 8 is a write-only register. The CRm and OPC_2 fields are used to encode the following TLB flush operations. Operation for all other values of OPC_2 and CRm is unpredictable.

Function	OPC_2	CRm	Data
Flush I+D	0b000	0b0111	Ignored
Flush I	0b000	0b0101	Ignored
Flush D	0b000	0b0110	Ignored
Flush D single entry	0b001	0b0110	Virtual address

5.2.10 Register 9 – Read-Buffer Operations

The read buffer is controlled and accessed through register 9 of coprocessor 15. The functions supported are: flush-all buffers, flush-a-single entry, load-an-entry (1, 4 or 8 words), and enable/disable user mode access.

The CRm and OPC_2 fields are used to encode these control operations. All other values for OPC_2 and CRm are undefined and the results of using them are unpredictable.

Function	OPC_2	CRm	Data
Flush all entries	0b000	0b0000	Ignored
Flush Buffer 0	0b001	0b0000	Ignored
Flush Buffer 1	0b001	0b0001	Ignored
Flush Buffer 2	0b001	0b0010	Ignored
Flush Buffer 3	0b001	0b0011	Ignored
Load Buffer 0 with one word	0b010	0b0000	Virtual address
Load Buffer 0 with four words	0b010	0b0100	Virtual address
Load Buffer 0 with eight words	0b010	0b1000	Virtual address
Load Buffer 1 with one word	0b010	0b0001	Virtual address
Load Buffer 1 with four words	0b010	0b0101	Virtual address
Load Buffer 1 with eight words	0b010	0b1001	Virtual address
Load Buffer 2 with one word	0b010	0b0010	Virtual address
Load Buffer 2 with four words	0b010	0b0110	Virtual address
Load Buffer 2 with eight words	0b010	0b1010	Virtual address
Load Buffer 3 with one word	0b010	0b0011	Virtual address
Load Buffer 3 with four words	0b010	0b0111	Virtual address
Load Buffer 3 with eight words	0b010	0b1011	Virtual address
Disable user-mode MCR access	0b100	0b0000	Ignored
Enable user-mode MCR access	0b101	0b0000	Ignored

See Chapter 6, “Caches, Write Buffer, and Read Buffer” for details on the use and operation of the read buffer.

5.2.11 Registers 10 – 12 RESERVED

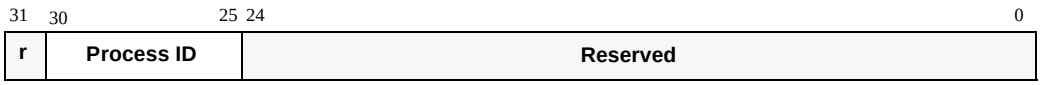
Accessing any of these registers yields unpredictable results.

5.2.12 Register 13 – Process ID Virtual Address Mapping

The SA-1100 supports the remapping of virtual addresses through a process ID (PID) register. The 6-bit PID value is OR'ed with bits 30..25 of the virtual address when bits 31..25 of the virtual address are zero. This effectively remaps the address to one of 64 "slots" in the lower 2 Gbyte address space. The following table shows the OPC_2 and CRm field encodings used to access the process ID register. This register is zero at reset and if left unmodified, effectively disables the remapping function. As such, no explicit enable or disable function is necessary. Reserved bits read as zero and must be written as zero. This register is readable and writable.

Function	OPC_2	CRm
Access process ID register	0b000	0b0000

The following figure shows the format of the process ID register.



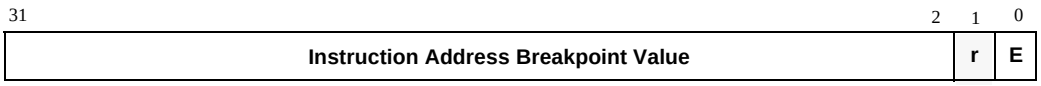
5.2.13 Register 14 – Debug Support (Breakpoints)

The SA-1100 supports address and data breakpoints through register 14 of coprocessor 15. The instruction formats follow. For a description of the breakpoint operation, see Chapter 15, "Debug Support". The following table shows the OPC_2 and CRm field encodings used to access the address and data breakpoints.

Function	OPC_2	CRm
Access data breakpoint address register (DBAR).	0b000	0b0000
Access data breakpoint value register (DBVR).	0b000	0b0001
Access data breakpoint mask register (DBMR).	0b000	0b0010
Load data breakpoint control register (DBCR). --- **DBCR Bit** **Action** --- lw 0 = Disable load watch 1 = Enable load watch saw 0 = Disable store address watch 1 = Enable store address watch sdw 0 = Disable store data watch 1 = Enable store data watch	0b000	0b0011
Write instruction breakpoint address and control register (IBCR). Low-order address bit is the address break enable/disable bit. Register not readable.	0b000	0b1000

The DBCR register is a 3-bit register used to control the enabling and disabling of the data breakpoints. Bits 0..2 are valid and positioned as shown below. Bits 3..31 are reserved. These bits read as zeros and writes have no effect.

The IBCR is a write-only register used to load an address breakpoint address and to set an enable bit for the function. If an address is loaded with bit 0 (E) set, then the address is enabled as a breakpoint. If bit zero is cleared, then the breakpoint is disabled. Bit 1 is reserved and should be written to zero.



5.2.14 Register 15 – Test, Clock, and Idle Control

Register 15 is a write-only register. The CRm and OPC_2 fields are used to encode the following control operations. Operation for all other values of OPC_2 and CRm is unpredictable.

Function	OPC_2	CRm
Enable odd-word loading of the linear feedback shift register (LFSR)	0b001	0b0001
Enable even-word loading of LFSR	0b001	0b0010
Clear LFSR	0b001	0b0100
Move LFSR to R14.abort	0b001	0b1000
Enable clock switching	0b010	0b0001
Disable clock switching	0b010	0b0010
RESERVED	0b010	0b0100
Wait for interrupt	0b010	0b1000

Caches, Write Buffer, and Read Buffer 6

To reduce effective memory access time, the Intel® StrongARM® SA-1100 Microprocessor (SA-1100) has an instruction cache, a data cache, a write buffer, and a read buffer. All except the read buffer are transparent to program execution. The following sections describe each of these units and give all necessary programming information.

6.1 Instruction Cache (Icache)

The SA-1100 contains a 16 Kbyte instruction cache (Icache). The Icache has 512 lines of 32 bytes (8 words), arranged as a 32-way set associative cache, and uses the virtual addresses generated by the processor core. The Icache is always reloaded a line at a time (8 words). It may be enabled or disabled via the SA-1100 control register, and is disabled on the assertion of nRESET or through a software or sleep reset sequence. (See Section 9, "System Control Module" on page 9-1 in the System Control Module for details.) The operation of the cache, when memory management is enabled, is further controlled by the *cacheable* or C bit stored in the memory-management page table. If memory management is disabled, all addresses are marked as cacheable (C=1). When memory management is enabled, the C bit in each page table entry can disable caching for an area of virtual memory.

6.1.1 Icache Operation

In the SA-1100, the instruction cache is searched regardless of the state of the C bit; only reads that miss the cache are affected. If, on an Icache miss, the C bit is a one or the Memory Management Unit (MMU) is disabled, a linefetch of 8 words is performed and it is placed in a cache bank with a round-robin replacement algorithm. If, on a miss, the MMU is enabled and the C bit is a zero for the given virtual address, an external memory access for a single word is performed and the cache is not written.The Icache should be enabled as soon as possible after reset for best performance.

6.1.2 Icache Validity

The Icache operates with virtual addresses, so care must be taken to ensure that its contents remain consistent with the virtual-to-physical mappings performed by the memory management unit. If the memory mappings are changed, the Icache validity must be ensured. The Icache is not coherent with stores to memory, so programs that write cacheable instruction locations must ensure the Icache validity. Instruction fetches do not check the write buffer, so data must not only be pushed out of the cache but the write buffer must also be drained.

6.1.2.1 Software Icache Flush

The entire Icache can be invalidated by writing to the SA-1100 cache operations register (register 7). The cache is flushed immediately when the register is written, but note that the following instruction fetches may come from the cache before the register is written.

6.1.3 Icache Enable/Disable and Reset

The Icache is automatically disabled and flushed on the assertion of nRESET. Once enabled, cacheable read accesses cause lines to be placed in the cache. If the Icache is subsequently disabled, no new lines are placed in the cache, but the cache is still searched and if the data is found, it will be used by the processor. If the data in the cache must not be used, then the cache must be flushed.

6.1.3.1 Enabling the Icache

To enable the Icache, set bit 12 in the control register. The MMU and Icache may be enabled simultaneously with a single control register write.

6.1.3.2 Disabling the Icache

To disable the Icache, clear bit 12 in the control register.

6.2 Data Caches (Dcaches)

The SA-1100 contains two logically separate data caches: the main data cache and the mini data cache (or minicache). The main data cache, an 8 Kbyte write-back Dcache, has 256 lines of 32 bytes (8words) in a 32-way set-associative organization. It is intended for use during most data accesses. This cache allocates on loads to spaces marked B=1 and C=1. Replacements in the main data cache are selected according to a set of round-robin pointers. At reset, the pointer in each block of the Dcache points to way zero of each 32-way block. As lines are allocated, the pointers are incremented to the next way of the set. After way 31 is allocated, the next linefill replaces (and copies back to memory, if dirty) the data in way zero. The minicache is a 512-byte write-back cache. It has 16 lines of 32 bytes (8 words) in a two-way set-associative organization and provides an alternate caching structure for dealing with large data structures that could thrash the main data cache. This cache allocates on loads to spaces marked B=0 and C=1. Replacements in the minicache use the same round-robin pointer mechanism as in the main data cache. However, since this cache is only two-way set-associative, the replacement algorithm reduces to a simple least-recently-used (LRU) mechanism.

The Dcaches are accessed in parallel and the design ensures that a particular line entry will exist in only one of the two at any time. Both Dcaches use the virtual address generated by the processor and allocate only on loads (write misses never allocate in the cache). Each line entry contains the physical address of the line and two dirty bits. The dirty bits indicate the status of the first and the second halves of the line. When a store hits in the Dcaches, the dirty bit associated with it is set. When a line is evicted from the Dcaches, the dirty bits are used to decide if all, half, or none of the line will be written back to memory using the physical address stored with the line. The Dcaches are always reloaded a line at a time (8 words).

The Dcaches allocate only on loads and according to the settings of the B and C bits in the MMU. If B=0 and C=1, the memory access allocates into the minicache. If B=1 and C=1, the memory access allocates into the main data cache. The Dcaches should be flushed prior to changing the bufferable and/or cacheable state of the page table mapping.

The main data cache and the minicache are enabled and disabled via the SA-1100 control register, and are disabled on **nRESET** as well as software, sleep, and watchdog reset. The operation of the Dcaches is further controlled by the *cacheable* or C bit and the *bufferable* or B bit stored in the

memory-management page table. For this reason, in order to use the Dcaches, the MMU must be enabled. The two functions may be enabled simultaneously with a single write to the control register.

Note: The Dcaches operate with virtual addresses, so care must be taken to ensure that their contents remain consistent with the virtual-to-physical mappings performed by the memory-management unit. If the memory mappings are changed, the validity of the Dcaches must be ensured.

6.2.1 Cacheable Bit – C

The cacheable bit determines whether, on load misses, the data being read should be placed in one of the two data caches. Cache hits are not affected by the cacheable bit; if a data access hits in the cache, the data is assumed to be valid and the load or store is performed. Typically, main memory is marked as cacheable to improve system performance and I/O space as noncacheable to stop the data from being stored in SA-1100's cache. For example, if the processor is polling a hardware flag in I/O space, it is important that the processor is forced to read data from the external peripheral, and not a copy of initial data held in the cache.

6.2.1.1 Cacheable Reads – C = 1

A linefetch of 8 words will be performed and it will be placed in a cache bank with a round-robin replacement algorithm.

6.2.1.2 Noncacheable Reads – C = 0

An external memory access will be performed and the cache will not be written.

6.2.2 Bufferable Bit – B

The bufferable bit does not affect writes that hit the Dcaches. If a store hits in the Dcaches, the store is assumed to be bufferable. Write-backs of dirty lines are treated as bufferable writes. See the Section 6.3, "Write Buffer (WB)" on page 6-5 for more information on the B bit.

Table 6-1 summarizes the effects of the B and C bits on the Dcaches.

Table 6-1. Effects of the Cacheable and Bufferable Bits on the Data Caches

		Load		Store	
B	C	Cache Hit	Cache Miss	Cache Hit	Cache Miss
0	0	Deliver cache data.	Load from memory. – No allocate.	Store to either cache. – Mark line dirty.	Store to memory. – No allocate.
0	1	Deliver cache data.	Allocate to minicache.	Store to either cache. – Mark line dirty.	Store to memory. – No allocate.
1	0	Deliver cache data.	Load from memory. – No allocate.	Store to either cache. – Mark line dirty.	Store to memory. – No allocate.
1	1	Deliver cache data.	Allocate to main data cache.	Store to either cache. – Mark line dirty.	Store to memory. – No allocate.

6.2.3 Software Dcache Flush

The SA-1100 supports the flush and clean operations on single entries of the Dcaches by writes to the cache operations registers. The flush whole cache is also supported. Note that since this is a write-back cache, in order to prevent the loss of data, a flush whole must be preceded by a sequence of loads to cause the cache to write back any dirty entries. The memory controller in the SA-1100 provides an internally decoded memory space to perform coherent Dcache flushing. This space resides in the upper 512 megabytes of the memory map (starting at virtual address 0hE000 0000) and, when accessed, is detected by the memory controller, which then returns zeros without incurring an external memory latency.

The following code causes the main data cache to flush all dirty entries:

```
;+
;Call:
;    R0 points to the start of a 8192 byte region of readable data used
;        only for this cache flushing routine.
;    bl    writeBackDC
;Return:
;    R0, R1, R2 trashed
;    Data cache is clean
;-
    writeBackDC
    movr0, 0hE000000
    addr1, r0, #8192
l1
    ldr r2, <r0>, #32
    teqr1, r0
    bnel1
    mcrp15, 0, r0, c7, c6, 0
    movpc, r14
```

A similar routine may be written to flush the minicache. To perform this flush, the MMU B and C settings must be as described above. The invalidate-all operation also invalidates the minicache.

6.2.3.1 Doubly Mapped Space

Since the Dcaches work with virtual addresses, it is assumed that every virtual address maps to a different physical address. If the same physical location is accessed by more than one virtual address, the cache cannot maintain consistency, since each virtual address has a separate entry in the cache, and only one entry is updated on a processor write operation. To avoid any cache inconsistencies, doubly mapped virtual addresses should be marked as noncacheable.

6.2.4 Dcaches Enable/Disable and Reset

The Dcaches are automatically disabled and flushed on the assertion of nRESET. Once enabled, cacheable read accesses cause lines to be placed in the Dcaches. If subsequently disabled, no new lines are placed in the Dcaches, but they are still searched and if the data is found, it is used by the processor. Write operations continue to update the Dcaches, thus maintaining consistency with the external memory. If the data in the Dcaches must not be used, then the Dcaches must be flushed.

6.2.4.1 Enabling the Dcaches

To enable the Dcaches, make sure that the MMU is enabled first by setting bit 0 in the control register, then enable the Dcaches by setting bit 2 in the control register. The MMU and Dcaches can be enabled simultaneously with a single control register write.

6.2.4.2 Disabling the Dcaches

To disable the Dcache, clear bit 2 in the control register.

6.3 Write Buffer (WB)

The SA-1100 write buffer is used to improve system performance by buffering up to 8 blocks of data of 1 to 16 bytes, at independent addresses. It can be enabled or disabled via the W bit (bit 3) in the SA-1100 control register. The buffer is disabled and all entries are marked empty following reset. Operation of the write buffer is further controlled by the *cacheable* or C bit and the *bufferable* or B bit, which are stored in the memory-management page tables. For this reason, in order to use the write buffer, the MMU must be enabled. The two functions can be enabled simultaneously with a single write to the control register. For a write to use the write buffer, both the W bit in the control register and the B bit in the corresponding page table must be set. It is not possible to abort buffered writes externally. Stores will not merge with other data at the same line address in the write buffer with the exception of store multiples, which do merge.

6.3.1 Bufferable Bit

This bit controls whether a write operation may use the write buffer. Typically, main memory is bufferable and I/O space unbufferable.

6.3.2 Write Buffer Operation

When the CPU performs a store, the Dcaches are first checked. If one of the Dcaches hits on the store and the protection for the location and mode of the store allows the write, then the write completes in the Dcaches and the write buffer is not used. If the location misses in the Dcaches, then the translation entry for that address is inspected and the state of the B and C bits determines which of the three following actions are performed. If the write buffer is disabled via the SA-1100 control register, writes are treated as if the B bit is a zero.

6.3.2.1 Writes to a Bufferable and Cacheable Location (B=1,C=1)

If the write buffer is enabled and the processor performs a write to a bufferable and cacheable location, and the data is in one of the caches, then the data is written to that cache, and the cache line is marked dirty. If a write to a bufferable area misses in both data caches, the data is placed in the write buffer and the CPU continues execution. The write buffer performs the external write sometime later. If a write is performed and the write buffer is full, then the processor is stalled until there is sufficient space in the buffer. No write buffer merging is allowed in the SA-1100 except during store multiples.

6.3.2.2 Writes to a Bufferable and Noncacheable Location (B=1,C=0)

If the write buffer is enabled and the processor performs a write to a bufferable but noncacheable location and misses in the Dcaches, the data is placed in the write buffer and the CPU continues execution. As with the cacheable case, merging is allowed only on store multiples. The write buffer performs the external write sometime later.

6.3.2.3 Unbufferable Writes (B=0)

If the write buffer is disabled or the CPU performs a write to an unbufferable area, the processor is stalled until the write buffer empties and the write completes externally. This requires several external clock cycles.

6.3.3 Enabling the Write Buffer

To enable the write buffer, ensure that the MMU is enabled by setting bit 0 in the control register, then enable the write buffer by setting bit 3 in the control register. The MMU and write buffer can be enabled simultaneously with a single write to the control register.

6.3.3.1 Disabling the Write Buffer

To disable the write buffer, clear bit 3 in the control register. Any writes already in the write buffer will complete normally, but a drain write buffer needs to be done to force all writes out to memory.

Note: The write buffer is used for copy-backs from the Dcaches even when they are disabled.

6.4 Read Buffer (RB)

The SA-1100 contains a software-programmable read buffer that can increase the performance of critical loop code by prefetching data. The RB enables the preallocation of read-only data into one of four 32-byte buffers without stalling the pipe. For subsequent loads that hit in the RB, data is sourced from the buffer instead of the Dcaches at a rate of 1 word per core clock. Also, because the programmer specifies which entry of the RB is used, critical data can be “locked” in to eliminate bus latency.

The RB is controlled using coprocessor 15, register 9, and provides the capability to allocate 1 word, a half-line (4 words), or a full line (8 words) into one of four entries of the RB. (See Chapter 5, “Coprocessors” for a detailed RB coprocessor description.) Half-line loads are automatically aligned onto half-block boundaries (the lower four address bits are ignored). Full-line loads are automatically aligned onto line boundaries (the lower five address bits are ignored). For partial cache line RB loads, only the words actually fetched are marked valid and can be sourced from the buffer. A small queue is used to ensure that subsequent RB load instructions go out in order.

When an RB allocate instruction is executed, the virtual address is looked up in the TB to check for a translation hit and possible access violations. If the access misses in the TB, the pipe is stalled until the page is fetched through the normal hardware tablewalk mechanism. If an access violation occurs, the RB load is NOP’d. For example, an RB allocate instruction can generate a data abort. Once the RB allocate has received a TB hit and no access violations, a bus access is requested that fills the appropriate buffer without stalling the core pipeline. Subsequent load instructions to this virtual address result in an RB hit and data is sourced from the appropriate entry to the core.

Any two data words with the same virtual address may not be contained in the RB at the same time. If an RB allocate references a data word that is already contained in another RB entry, then the old RB entry is invalidated and the new allocation is performed. It is possible for a portion of a cache clock at a given virtual address to be contained in one RB entry while another portion of the same block is contained in another RB entry. However, a given word can not be in more than one entry at a time.

If a load instruction misses in the RB, then a normal cache fill is performed (provided the cache is enabled and the page is marked cacheable). It then presents the possibility of having a partial line resident in the RB as well as having the line present in one of the Dcaches. This presents coherency issues that must be managed by software. If this situation does occur and the addressed data is in both the Dcache and the RB, then the data is sourced from the RB. If an RB entry contains a partial cache block (1 or 4 words), then those words will be sourced from the RB while the remaining words are sourced from the data cache or memory.

RB allocate instructions are not affected by the cache enable bit (bit 2 in the control register) or by the C bit in the MMU. Any RB allocate to a valid RB entry causes that RB entry to be invalidated, followed by a new allocation for the desired data. This occurs regardless of the address of the data currently in the buffer. For example, back-to-back RB allocate instructions to the same entry at the same address will invalidate the entry caused by the first instruction prior to performing the second fill.

An RB allocate or a load instruction that is issued to an RB entry currently being filled will stall until the fill completes. If a data abort is signaled on a read buffer allocate, the fill completes. After that, if a load to that entry is attempted, a data abort exception is issued. The coprocessor 15 register provides the ability to invalidate individual entries in the RB or to invalidate the entire buffer in one operation. RB coherency must be managed in software. Writes to addresses present in the read buffer are not written into the buffer. Specific RB entries must be invalidated before writing to the addresses or changing the page tables of the entries. Coherency is not checked between the RB and the WB. The WB should be drained prior to performing an RB load.

Memory-Management Unit (MMU) 7

This chapter describes the memory-management functions.

7.1 Overview

The Intel® StrongARM® SA-1100 Microprocessor (SA-1100) implements the standard ARM™ memory-management functions using two 32-entry fully associative translation buffers (TBs). One is used for instruction accesses and the other for data accesses. On a TB miss, the translation table hardware is invoked to retrieve the translation and access permission information. Once retrieved, if the entry maps to a valid page or section, then the information is placed into the TB. The replacement algorithm in the TB is round robin. For an invalid page or section, an abort is generated and the entry is not placed in the TB.

7.1.1 MMU Registers

See Section 5.2, "Coprocessor 15 Definition" on page 5-2 for a description of the Memory Management Unit (MMU) coprocessor 15 registers supported by the SA-1100.

7.2 MMU Faults and CPU Aborts

The MMU generates four faults:

- Alignment fault
- Translation fault
- Domain fault
- Permission fault

Alignment faults are generated by word loads or stores with the low-order two address bits nonzero, and by load or store half words when the low-order address bit is a one. Translation faults are generated by access to pages marked invalid by the memory-management page tables. Domain faults and permission faults are generated by accesses to memory that are protected by the current mode, domain, and page protection. See the *ARM Architecture Reference* for more information. In addition, an external abort may be raised on external data accesses.

7.3 Data Aborts

The SA-1100 takes a data abort exception due to: MMU-generated exceptions, accessing reserved memory space, and assertion of the abort pin while accessing expansion memory. Writes to memory areas marked as bufferable ignore the external abort pin.

7.3.1 Cacheable Reads (Linefetches)

A linefetch can be safely aborted on any word in the transfer. If an abort occurs during the linefetch, the cache is purged so it will not contain invalid data. If the abort happens before the word that was requested by the access is returned, the load is aborted. If the abort happens after the word that was requested by the access is returned, the load completes and the fill is aborted (but no exception is generated).

7.3.2 Buffered Writes

Buffered writes cannot be externally aborted. Therefore, the system should be configured such that it does not perform buffered writes to areas of memory that are capable of flagging an external abort.

7.4 Interaction of the MMU, Icache, Dcache, and Write Buffer

The MMU, Icache, Dcache, and WB can be enabled or disabled independently. The Icache can be enabled with the MMU enabled or disabled. However, the Dcache and WB can only be enabled when the MMU is enabled. Because the write buffer is used to hold dirty copy-back cached lines from the Dcache, it must be enabled along with the Dcache. Therefore, only four of the eight combinations of the MMU, Dcache, and WB enables are valid. There are no hardware interlocks on these restrictions, so invalid combinations will cause undefined results.

Table 7-1. Valid MMU, Dcache, and Write Buffer Combinations

MMU	Dcache	Write Buffer
Off	Off	Off
On	Off	Off
On	Off	On
On	On	On

The following procedures must be observed.

To enable the MMU:

1. Program the translation table base and domain access control registers.
2. Program level 1 and level 2 page tables as required.
3. Enable the MMU by setting bit 0 in the control register.

Note: Care must be taken if the translated address differs from the untranslated address because the three instructions following the enabling of the MMU will have been fetched using “flat translation”, and enabling the MMU may be considered a branch with delayed execution. A similar situation occurs when the MMU is disabled. Consider the following code sequence:

```
MOV             R1, #0x1
MCR             15,0,R1,0,0          ; Enable MMU
Fetch nontranslated
Fetch nontranslated
Fetch nontranslated
Fetch Translated
```

To disable the MMU:

1. Disable the WB by clearing bit 3 in the control register.
2. Disable the Dcache by clearing bit 2 in the control register.
3. Disable the Icache by clearing bit 12 in the control register.
4. Disable the MMU by clearing bit 0 in the control register.

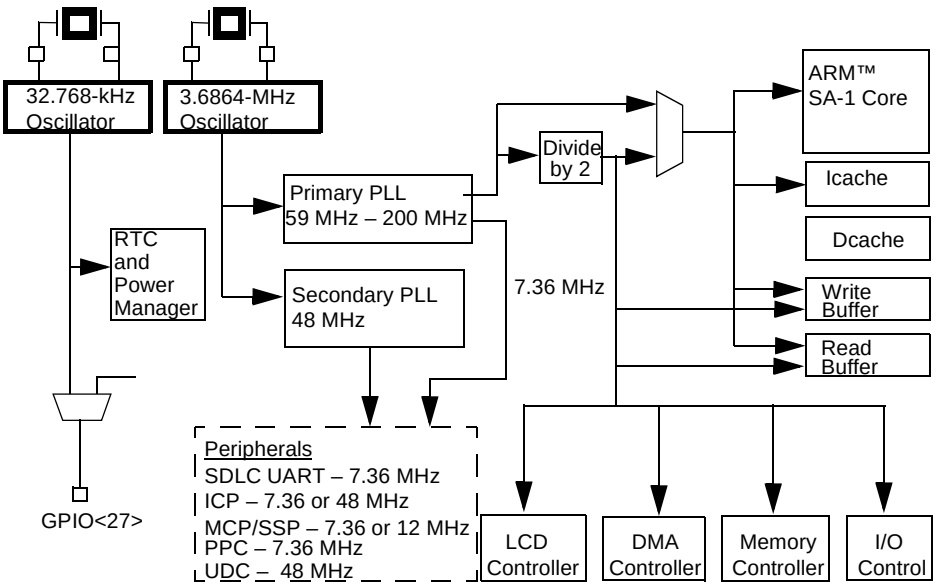
Note: If the MMU is disabled and subsequently reenabled, the contents of the TB is preserved. If the contents are now invalid, the TB should be flushed before reenabling the MMU.

7.5 Mini Data Cache

The mini data cache is a 16-entry, 2-way set-associative data cache. It is accessed in parallel with the main data cache. A data reference is allocated into the mini data cache if the B and C bits in the MMU are 0 and 1, respectively. A line of data can reside only in one of the two Dcaches at any one time. Both Dcaches must be flushed prior to any page table manipulation that could change the allocation policy.

Clocks 8

This section describes the Intel® StrongARM® SA-1100 Microprocessor (SA-1100) clocks. The following diagram shows the distribution of clocks in the SA-1100. The 3.6864-MHz oscillator feeds both PLLs. The primary PLL provides clocks for the core logic and a 7.36-MHz clock for several of the serial controllers. The core, Dcaches, and read and write buffers use either the full-speed core clock or the divided-down clock. The LCD controller, DMA, memory controller, and GPIO use the core clock divided by 2 (RCLK). The 32.768-kHz oscillator feeds the real-time clock (RTC) and the power manager logic. The secondary PLL provides the clock for the UDC, the ICP, and the MCP. The oscillators and PLLs are completely integrated with the SA-1100 and require no external devices other than the crystals for operation.



8.1 SA-1100 Crystal Oscillators

The SA-1100 clocks are derived from two crystals connected to onchip oscillators. The first clock source is a 3.6864-MHz crystal that feeds the CPU PLL and the 48-MHz PLL. The CPU PLL multiplies the oscillator output up to the core frequency. This frequency is then divided down to generate baud rates for the serial ports. If the UARTs are not being used or do not need standard baud rates, then the 3.6864 -Hz oscillator may be replaced with a 3.5795-MHz crystal to generate frequencies as shown in Table 8-1.

The second oscillator is connected to a 32.768-kHz crystal. The output of this oscillator clocks the power management controller and the real-time clock (RTC).

See Appendix B, “3.6864–MHz Oscillator Specifications” and Appendix C, “32.768–kHz Oscillator Specifications” for detailed specifications of the crystal oscillators.

8.2 Core Clock Configuration Register

The core clock frequency is configured by software through the core clock configuration field (CCF<4:0>) in the power manager phase-locked loop (PLL) configuration register (PPCR). This field should be programmed during the boot sequence for the desired full-speed operation. nRESET clears the field by selecting the lowest frequency operation.

See Section 9.5, “Power Manager” on page 9-26 for the physical address used to access this register.

Table 8-1 shows the core clock frequency as a function of the CCF setting.

Table 8-1. Core Clock Configurations

CCF<4:0>	Core Clock Frequency in MHz	
	3.6864-MHz Crystal Oscillator	3.5795-MHz Crystal Oscillator
00000	59.0	57.3
00001	73.7	71.6
00010	88.5	85.9
00011	103.2	100.2
00100	118.0	114.5
00101	132.7	128.9
00110	147.5	143.2
00111	162.2	157.5
01000	176.9	171.8
01001	191.7	186.1
01010	206.4	200.5
01011	221.2	214.8
01100– 11111	Not supported.	—

8.2.1 Restrictions on Changing the Core Clock Configuration

When the CPU writes to the PPCR, the core clock PLL and the 48-MHz PLL are stopped for a period of time to allow the core clock PLL to relock to the new frequency. When these PLLs are stopped, the core clock and all clocks derived from that clock are stopped. When this happens, certain units within the SA-1100 (the LCD controller, all serial controllers, the DMA controller, and the OS timer) will experience an interruption in operation for approximately 150 microseconds after the PPCR is written.

Because of these restrictions, it is recommended that the user not change the PPCR *except* immediately following a hard reset or immediately following wake-up from sleep mode. The LCD controller, all serial controllers (except the UDC), the DMA controller, and the OS timer are already disabled and are not affected by an interruption in their clock stream. In addition to these restrictions, the PPCR must be written prior to enabling clock switching. Note that the 32.768-kHz clock is not affected by any change in the PPCR and units using this clock (power management, RTC) do not see any interruption in service during the 150 microsecond period.

8.3 Driving SA-1100 Crystal Pins from an External Source

In most applications, a 3.6864-MHz crystal will be connected between the PXTAL and the PEXTAL pins. Similarly, a 32.768-kHz crystal will be connected between the TXTAL and TEXTAL pins. In some applications, supplying these clocks from an external source may be preferred. This is accommodated in the SA-1100 design by:

- Supplying the 32.768-kHz clock from an external source
  - — Only the TXTAL pin is driven. The TEXTAL pin must be left floating.
  - — The peak-to-peak voltage swing on TXTAL must be at least 0.6 V and the voltage on the pin must remain within the range of 0 V to 1 V, independent of the other power supply voltages applied to the processor.
- Supplying a 3.6864-MHz clock from an external source
  - — Both PXTAL and PEXTAL are driven with complementary signals.
  - — The peak-to-peak voltage swing on PXTAL and PEXTAL must be at least 0.6 V and the voltage on the pin must remain in the range of 0 V to 1 V, independent of the other power supply voltages applied to the processor.†
  - — When an external clock is being used, the pull-down path in the internal 3.6864 MHz oscillator is active. In order to limit the current into the internal oscillator, it is recommended that the minimum impedance to the positive supply be controlled. The maximum current sourced by the external clock source when the clock is at its maximum positive voltage should be about 1 mA.†
  - — The maximum impedance of the external clock source is set by the minimum slew rate at the PXTAL and PEXTAL pins, approximately 1 V per 100 ns.†

    †These constraints can be satisfied by the following suggestions:
- For applications in which a pulse generator is available, drive differential 1-V signals through series 1-K resistors (after the usual 50-ohm terminators-to-ground).
- To supply external clock signals from a 3.3-V supply, drive signals with open collector or tristatable drivers. Set high level with 3.3 K from 3.3 V to the output and 1.3 K from the output to ground.
- To supply external clock signals from a 1.5-V supply, drive signals with open collector or tristatable drivers. Set high level with 1.5 K from 1.5 V to the output and 2.7 K from output to ground. This solution may be preferred in portable applications that turn off the 1.5-V supply in sleep mode because this would eliminate the current through the resistors in sleep mode.

The two pairs of crystal pins are located close to each other on the processor. This arrangement is advantageous when there are crystals connected to the pins because the low signal swings and slow edges result in limited noise coupling between the pins. If one of the crystals is replaced by an independent signal source and the other is not, some degradation of the remaining crystal oscillator performance can result due to increased noise coupling. If only one crystal is being used, this effect can be reduced by limiting the speed of the edge rate on the pin driven by the independent source.

If the PXTAL or TXTAL pin is driven above the voltage indicated, there will be no permanent damage to the processor for pin voltages less than 2.5 V. However, ESD diodes on these pins will attempt to clamp the voltage at approximately 1.5 V. The clamping action results in significant noise injected into an internally generated supply used by several sensitive circuits on the processor. Consequently, driving this pin higher than the 1 V limit can result in unpredictable operation not obviously connected with the crystal pins. Users should refrain from driving the crystal pins higher than 1 V even if there is no obvious side effect.

Note: In every system, there must be a provision for both a 3.6864-MHz and a 32.768-kHz source either from an external oscillator or a crystal.

8.4 Clocking During Test

If TCK_BYP is high, then the PLLs and oscillators are not used and the high-speed core clock is supplied externally on the TESTCLK pin. This mode is for testing only and is not supported for standard operation.

System Control Module 9

This chapter describes the system control module that controls several processor-wide system functions. The units contained in the system control module are: the general-purpose I/O ports, the interrupt controller, the real-time clock, the operating system timer, the power manager, and the reset controller.

9.1 General-Purpose I/O

The Intel® StrongARM® SA-1100 Microprocessor (SA-1100) provides 28 general-purpose I/O (GPIO) port pins for use in generating and capturing application-specific input and output signals. Each pin is programmable as an input or output and as an interrupt source. All 28 pins are configured as inputs during the assertion of reset, and remain inputs until they are configured otherwise.

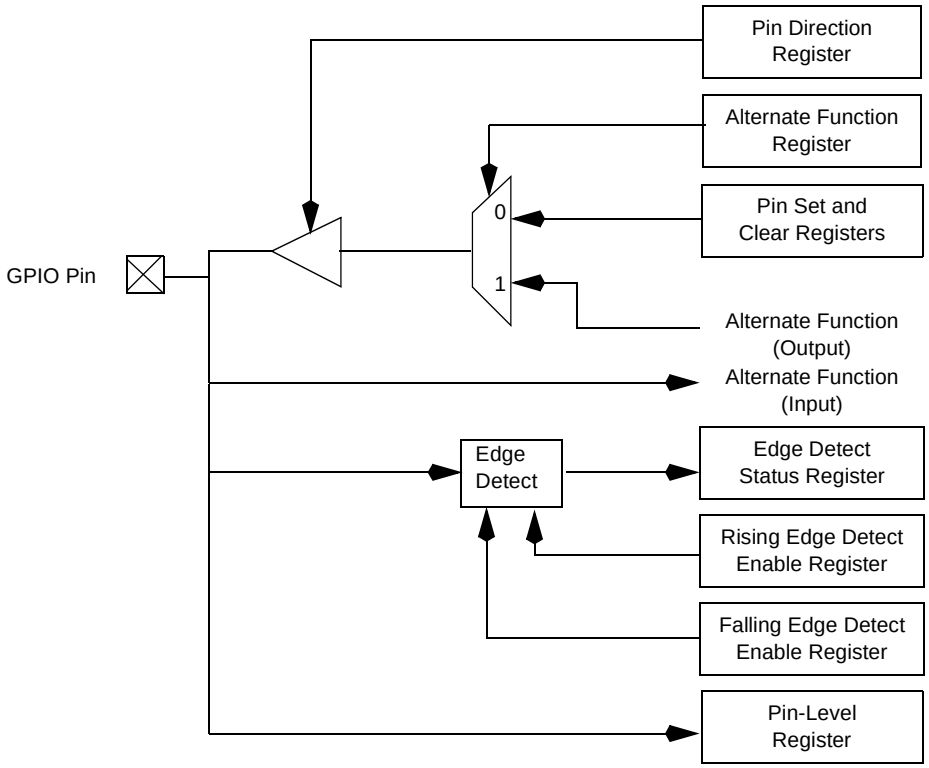
Each GPIO pin can be configured as an input or an output by programming the GPIO pin direction register (GPDR). When programmed as an output, the pin can be controlled by writing to the GPIO pin output set register (GPSR) and the GPIO pin output clear register (GPCR). Writing to these registers controls the output data register, which is not directly readable or writable. The set and clear registers can be written regardless of whether the pin is configured as an input or an output. The programmed output state will take effect when the pin is reconfigured as an output.

When programmed as an input, the current state of each GPIO pin can be read through the GPIO pin-level register (GPLR). This register can be read at any time and can be used to confirm the state of the pin when it is configured as an output. In addition, each GPIO pin can be programmed to detect a rising and/or falling edge through the GPIO rising-edge detect register (GRER) and GPIO falling-edge detect register (GFER). The state of the edge detect can be read through the GPIO edge detect status register (GEDR). These edge detects can be programmed to generate an interrupt (see the Section 9.2, “Interrupt Controller” on page 9-11) or to serve as a wake-up event to bring the SA-1100 out of sleep mode (see the Section 9.5, “Power Manager” on page 9-26).

When the SA-1100 enters sleep mode, the contents of the power manager sleep state register (PGSR) is loaded into the output data register. If the particular pin is programmed as an output, then the state in the PGSR will be driven onto the pin before entering sleep. When the SA-1100 exits sleep mode, these values remain until reprogrammed by writing to the GPSR and GPCR.

Some GPIO pins can also serve an alternate function within the SA-1100. Certain modes within the serial controllers and LCD controller require extra pins. These functions are hardwired into specific GPIO pins and their use is described in the following sections. Even though a GPIO pin has been taken over for an alternate function, the user must still program the proper direction of that pin through the GPDR. Details on alternate functions are provided in following sections. Figure 9-1 shows a block diagram of a single GPIO pin.

Figure 9-1. General-Purpose I/O Block Diagram



9.1.1 GPIO Register Definitions

There are a total of eight registers within the GPIO control block: one is used to monitor pin state; two are used to control pin state; one is used to control pin direction; two are used to specify a pin's edge type that should be detected; and one is used to flag when specified edge types are detected on pins. The last register indicates whether a pin is used as normal GPIO or whether it is taken over by the alternate function. Note that the pin direction register (GPDR) is the only register that is initialized by reset. The values in all other GPIO registers are unknown following reset and must be initialized by software.

9.1.1.1 GPIO Pin-Level Register (GPLR)

The state of each of the GPIO port pins is visible through the GPIO pin-level register (GPLR). Each bit number corresponds to the port pin number from bit 0 to bit 27. This is a read-only register that is used to determine the current level of a particular pin (regardless of the programmed pin direction).

The following table shows the locations of the 28 pin-level bits within the GPLR. This is a read-only register. For reserved bits, reads return zero; a question mark indicates that the values are unknown at reset.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read	Reserved				PL27	PL26	PL25	PL24	PL23	PL22	PL21	PL20	PL19	PL18	PL17	PL16
Reset	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?

-	-

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PL15	PL14	PL13	PL12	PL11	PL10	PL9	PL8	PL7	PL6	PL5	PL4	PL3	PL2	PL1	PL0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	Name	Description
{n}	PL{n}	GPIO port pin level n (where n = 0 through 27). 0 – Pin state is low. 1 – Pin state is high.
31.. 28	—	Reserved.

9.1.1.2 GPIO Pin Direction Register (GPDR)

Pin direction is controlled by programming the GPIO pin direction register (GPDR). The GPDR contains one direction control bit for each of the 28 port pins. If a direction bit is programmed to a one, the port is an output. If it is programmed to a zero, it is an input. At hardware reset, all bits in this register are cleared, configuring all GPIO pins as inputs. Soft resets and sleep reset have no effect on this register. For reserved bits, writes are ignored and reads return zero. The following table shows the location of each pin direction bit in the GPIO pin direction register.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	Reserved				PD27	PD26	PD25	PD24	PD23	PD22	PD21	PD20	PD19	PD18	PD17	PD16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
{n}	PD{n}	GPIO port pin direction n (where n = 0 through 27). 0 – Pin configured as an input. 1 – Pin configured as an output.
31..28	—	Reserved.

9.1.1.3 GPIO Pin Output Set Register (GPSR) and Pin Output Clear Register (GPCR)

When a port is configured as an output, the user controls the state of the pin by writing to either the GPIO pin output set register (GPSR) or the GPIO pin output clear register (GPCR). An output pin is set by writing a one to its corresponding bit within the GPSR. To clear an output pin, a one is written to the corresponding bit within the GPCR. These are write-only registers. Reads return unpredictable values. Writing a zero to any of the GPSR or GPCR bits has no effect. Writing a one to a GPSR or GPCR bit corresponding to a pin that is configured as an input has no effect. For reserved bits, writes are ignored. The following tables show the locations of the GPSR bits and the locations of the GPCR bits. These are write-only registers and reset values do not apply.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Write	Reserved				PS27	PS26	PS25	PS24	PS23	PS22	PS21	PS20	PS19	PS18	PS17	PS16
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	PS15	PS14	PS13	PS12	PS11	PS10	PS9	PS8	PS7	PS6	PS5	PS4	PS3	PS2	PS1	PS0
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit	Name	Description
{n}	PS{n}	GPIO output pin set n (where n = 0 through 27). 0 – Pin level unaffected. 1 – If pin configured as an output, set pin level high (one).
31..28	—	Reserved.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Write	Reserved				PC27	PC26	PC25	PC24	PC23	PC22	PC21	PC20	PC19	PC18	PC17	PC16
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit	Name	Description
{n}	PC{n}	GPIO output pin clear n (where n = 0 through 27). 0 – Pin level unaffected. 1 – If pin configured as an output, clear pin level low (zero).
31.. 28	—	Reserved.

The user can test a bit within the GPLR corresponding to a pin that is configured as an output after having set or cleared the pin state to determine if there is an external conflict on the pin. For example, if an off-chip device is driving a GPIO output pin high and the user has cleared the pin's state by writing a one to its GPCR bit, the user can read the GPLR, then compare the written value (zero) to the actual value (one) to detect the conflict.

9.1.1.4 GPIO Rising-Edge Detect Register (GRER) and Falling-Edge Detect Register (GFER)

Each GPIO port can also be programmed to detect a rising-edge, falling-edge, or either transition on a pin. When an edge is detected that matches the type of edge programmed for the pin, a status bit is set. The interrupt controller can be programmed to signal an interrupt to the CPU or wake up the SA-1100 from sleep mode when any one of these status bits is set.

The GPIO rising-edge and falling-edge detect registers (GRER and GFER, respectively) are used to select the type of transition on a GPIO pin that causes a bit within the GPIO edge detect status register (GEDR) to be set. For a given GPIO port pin, its corresponding GRER bit is set to cause a GEDR status bit to be set when the pin transitions from logic level zero (0) to one (1). Likewise, GFER is used to set the corresponding GEDR status bit when a transition from logic level one (1) to zero (0) occurs. When the corresponding bits are set in both registers, either a falling- or a rising-edge transition causes the corresponding GEDR status bit to be set.

The following table shows both the rising-edge and falling-edge enable bit locations corresponding to all 28 port pins. For reserved bits, writes are ignored and reads return zero; a question mark indicates that the values are unknown at reset.

GRER

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	Reserved				RE27	RE26	RE25	RE24	RE23	RE22	RE21	RE20	RE19	RE18	RE17	RE16
Reset	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	RE15	RE14	RE13	RE12	RE11	RE10	RE9	RE8	RE7	RE6	RE5	RE4	RE3	RE2	RE1	RE0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1

Bit	Name	Description: GPIO Rising-Edge Detect Register (GRER)
{n}	RE{n}	GPIO pin n rising-edge detect (where n = 0 through 27). 0 – Disable rising-edge detect. 1 – Set corresponding GEDR status bit when a rising edge is detected on the GPIO pin.
31.. 28	—	Reserved.

GFER

Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	Reserved				FE27	FE26	FE25	FE24	FE23	FE22	FE21	FE20	FE19	FE18	FE17	FE16
Reset	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	FE15	FE14	FE13	FE12	FE11	FE10	FE9	FE8	FE7	FE6	FE5	FE4	FE3	FE2	FE1	FE0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1

Bit	Name	Description: GPIO Falling-Edge Detect Register (GRER)
{n}	FE{n}	GPIO pin n falling-edge detect (where n = 0 through 27). 0 – Disable falling-edge detect. 1 – Set corresponding GEDR status bit when a falling edge is detected on the GPIO pin.
31..28	—	Reserved.

9.1.1.5 GPIO Edge Detect Status Register (GEDR)

The GPIO edge detect status register (GEDR) contains 28 status bits that correspond to the 28 GPIO port pins. When an edge detect occurs on a pin that matches the type of edge programmed in the GRER and/or GFER registers, the corresponding status bit is set in GEDR. Once a GEDR bit is set, the CPU must clear it. GEDR status bits are cleared by writing a one to them. Writing a zero to a GEDR status bit has no effect.

Each edge detect that sets the corresponding GEDR status bit for GPIO pins 0 – 27 can trigger an interrupt request. Pins 27 – 11 together form a group that can cause one interrupt request to be triggered when *any* one of the GEDR status bits 27 – 11 is set. Each of GPIO pins 10 – 0 causes an independent first-level interrupt. See the Section 9.2, “Interrupt Controller” on page 9-11 for a description of the programming of GPIO interrupts. The following table shows a summary of GEDR; a question mark indicates that the values are unknown at reset.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	Reserved				ED27	ED26	ED25	ED24	ED23	ED22	ED21	ED20	ED19	ED18	ED17	ED16
Reset	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	ED15	ED14	ED13	ED12	ED11	ED10	ED9	ED8	ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	Name	Description
{n}	ED{n}	GPIO edge detect status n (where n = 0 through 27). 0 – No edge detect has occurred on pin as specified in GRER and/or GFER. 1 – Edge detect has occurred on pin as specified in GRER and/or GFER.
31..28	—	Reserved.

9.1.1.6 GPIO Alternate Function Register (GAFR)

The GPIO alternate function register (GAFR) contains 28 control bits that correspond to the 28 GPIO port pins. When the processor sets a bit in the GAFR, the corresponding GPIO pin is switched over to that pin's alternate function. See the following section for details on alternate functions. This register is cleared to all zeros on all reset conditions.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	Reserved				AF27	AF26	AF25	AF24	AF23	AF22	AF21	AF20	AF19	AF18	AF17	AF16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	AF15	AF14	AF13	AF12	AF11	AF10	AF9	AF8	AF7	AF6	AF5	AF4	AF3	AF2	AF1	AF0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
{n}	AF{n}	GPIO alternate function bits (where n = 0 through 27). A bit set in this register indicates that the corresponding GPIO pin is to be used for its alternate function. A zero in this register indicates that the corresponding GPIO pin is to be used for its normal GPIO function.
31..28	—	Reserved.

9.1.2 GPIO Alternate Functions

Most GPIO pins have an alternate function that can be invoked to enable additional functionality within the SA-1100. If a GPIO is used for this alternate function, then it cannot be used as a GPIO at the same time. Pins 0 and 1 are reserved because of their special use during sleep mode and are not available for any alternate function. The following table shows each GPIO pin and its corresponding alternate function. For more details on an alternate function, see the section that corresponds to its name in the Unit column in the table.

Pin	Alternate Function	Direction	Unit	Signal Description
GP<27>	32KHZ_OUT	Output	Clocks	Raw 32.768-kHz oscillator output
GP<26>	RCLK_OUT	Output	Clocks	Internal clock/2
GP<25>	RTC clock	Output	RTC	Trimmed 1-Hz clock
GP<24>	Reserved	—	—	—
GP<23>	TREQB	Input	Test controller	TIC request B
GP<22>	TREQA/MBREQ	Input	Test controller	Either TIC request A or MBREQ
GP<21>	TIC_ACK/MBGNT	Output	Test controller	Either TIC acknowledge or MBGNT
GP<21>	MCP_CLK	Input	Serial port 4	MCP clock in
GP<20>	UART_SCLK3	Input	Serial port 3:UART	Sample clock input
GP<19>	SSP_CLK	Input	Serial port 2:SSP	Sample clock input
GP<18>	UART_SCLK1	Input	Serial port 1:UART	Sample clock input
GP<17>	SDLC_AAF	Output	Serial port 1:SDLC	Abort after frame control
GP<16>	SDLC_SCLK	I/O	Serial port 1:SDLC	Geoport clock out
GP<15>	UART_RXD	Input	Serial port 1:UART	UART receive
GP<14>	UART_TXD	Output	Serial port 1:UART	UART transmit
GP<13>	SSP_SFRM	Output	Serial Port 4:SSP	SSP frame clock
GP<12>	SSP_SCLK	Output	Serial port 4:SSP	SSP serial clock
GP<11>	SSP_RXD	Input	Serial port 4:SSP	SSP receive
GP<10>	SSP_TXD	Output	Serial port 4:SSP	SSP transmit
GP<2..9>	LDD<8..15>	Output	LCD controller	High-order data pins for split-screen color LCD support
GP<1>	Reserved	—	—	No alternate function
GP<0>	Reserved	—-	—	No alternate function

9.1.3 GPIO Register Locations

The following table shows the registers associated with the GPIO block and the physical addresses used to access them.

Address	Name	Description
0h 9004 0000	GPLR	GPIO pin-level register
0h 9004 0004	GPDR	GPIO pin direction register
0h 9004 0008	GPSR	GPIO pin output set register
0h 9004 000C	GPCR	GPIO pin output clear register
0h 9004 0010	GRER	GPIO rising-edge detect register
0h 9004 0014	GFER	GPIO falling-edge detect register
0h 9004 0018	GEDR	GPIO edge detect status register
0h 9004 001C	GAFR	GPIO alternate function register

9.2 Interrupt Controller

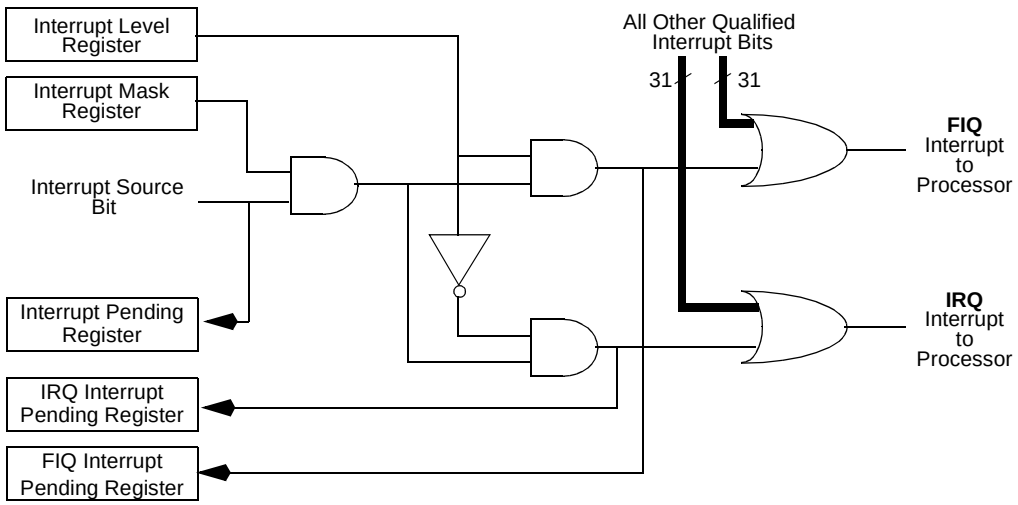
The SA-1100 interrupt controller provides masking capability for all interrupt sources and combines them into their final state, either an FIQ or IRQ processor interrupt. The interrupt hierarchy of the SA-1100 is a two-level structure.

The first level of the structure, represented by the interrupt controller IRQ pending register (ICIP) and the interrupt controller FIQ pending register (ICFP) contain the all-enabled and unmasked interrupt sources. Interrupts are enabled at their source and unmasked in the interrupt controller mask register (ICMR). The ICIP contains the interrupts that are programmed to generate an IRQ interrupt. The ICFP contains all valid interrupts that are programmed to generate an FIQ interrupt. This routing is programmed via the interrupt controller level register (ICLR).

The second level of the interrupt structure is represented by registers contained in the source device (the device generating the first-level interrupt bit). Second-level interrupt status gives additional information about the interrupt and is used inside the interrupt service routine. In general, multiple second-level interrupts are OR'ed to produce a first- level interrupt bit. The enabling of interrupts is performed inside the source device.

In most cases, the root source of an interrupt can be determined through reading two register locations: the ICIP or ICFP (depending on which interrupt handler the software is in) to determine the interrupting device, followed by the status register within that device to find the exact function needing service. When the SA-1100 is in idle mode (see the Section 9.5, "Power Manager" on page 9-26), any enabled interrupt causes it to resume operation. The interrupt mask register is ignored during idle mode. Figure 9-2 shows a block diagram of the interrupt controller.

Figure 9-2. Interrupt Controller Block Diagram



9.2.1 Interrupt Controller Register Definitions

The interrupt controller contains four registers: the interrupt controller IRQ pending register (ICIP), the interrupt controller FIQ pending register (ICFP), the interrupt controller mask register (ICMR), and the interrupt controller level register (ICLR). Following reset, the FIQ and IRQ interrupts are disabled within the CPU, and the states of all of the interrupt controller's registers are unknown and must be initialized by software before interrupts are enabled within the CPU.

9.2.1.1 Interrupt Controller Pending Register (ICPR)

The ICPR is a 32-bit read-only register that shows all active interrupts in the system. These bits are not affected by the state of the mask register (ICMR). The following table shows the pending interrupt source assigned to each bit position in the ICPR. Also included in the table are the source units for the interrupts and the number of second-level interrupts associated with each. For more detail on the second-level interrupts, see the section describing that unit.

Bit Position	Unit	Source Module	# of Level 2 Sources	Bit Field Description
IP<31>	System	Real-time clock	1	RTC equals alarm register.
IP<30>			1	One Hz clock TIC occurred.
IP<29>		Operating system timer	1	OS timer equals match register 3.
IP<28>			1	OS timer equals match register 2.
IP<27>			1	OS timer equals match register 1.
IP<26>			1	OS timer equals match register 0.
IP<25>	Peripheral	DMA controller	3	Channel 5 service request.
IP<24>			3	Channel 4 service request.
IP<23>			3	Channel 3 service request.
IP<22>			3	Channel 2 service request.
IP<21>			3	Channel 1 service request.
IP<20>			3	Channel 0 service request.
IP<19>		Serial port 4b	3	SSP service request.
IP<18>		Serial port 4a	8	MCP service request.
IP<17>		Serial port 3	6	UART service request.
IP<16>		Serial port 2	6+6	UART/HSSP service request.
IP<15>		Serial port 1b	6	UART service request.
IP<14>		Serial port 1a	5	SDLC service request.
IP<13>		Serial port 0	6	UDC service request.
IP<12>		LCD controller	12	LCD controller service request.
IP<11>	System	General-purpose I/O	17	“OR” of GPIO edge detects 27-11.
IP<10>			1	GPIO<10> edge detect.
IP<9>			1	GPIO<9> edge detect.
IP<8>			1	GPIO<8> edge detect.
IP<7>			1	GPIO<7> edge detect.
IP<6>			1	GPIO<6> edge detect.
IP<5>			1	GPIO<5> edge detect.
IP<4>			1	GPIO<4> edge detect.
IP<3>			1	GPIO<3> edge detect.
IP<2>			1	GPIO<2> edge detect.
IP<1>			1	GPIO<1> edge detect.
IP<0>			1	GPIO<0> edge detect.
		Total level 2 interrupt sources	110	

Several units have more than one source per interrupt signal. When an interrupt is signalled from one of these units, the interrupt handler routine identifies which interrupt was signalled using the interrupt controller’s flag register (this identifies the unit that made the request, but not the exact source). The handler then reads the interrupting unit’s status register to identify which source within the unit signalled the interrupt. For all interrupts that have one corresponding source, the interrupt handler routine needs to use only the interrupt controller’s registers to identify the exact cause of the interrupt.

9.2.1.2 Interrupt Controller IRQ Pending Register (ICIP) and FIQ Pending Register (ICFP)

The ICIP and the ICFP contain one flag per interrupt (32 total) that indicates an interrupt request has been made by a unit. Inside the interrupt service routine, the ICIP and ICFP are read to determine the interrupt source. In general, software then reads status registers within the interrupting device to determine how to service the interrupt.

Bits within the ICPR are read only, and represent the logical OR of status bits for a given interrupt within the source unit. Once an interrupt has been serviced, the handler clears the pending interrupt at the *source* by writing a one to the necessary status bit. Clearing the interrupt status bit at the source automatically clears the corresponding ICIP and ICFP flag provided there are no other interrupt status bits set within the source unit.

All interrupt source status bits are cleared by writing a one to them. Writing a zero to an interrupt status bit has no effect. The following table shows the bit locations corresponding to the 32 separate interrupt pending status flags in the ICIP. The next table shows the bit locations corresponding to the 32 separate interrupt pending status flags in the ICFP. This is a read-only register.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read	IP31	IP30	IP29	IP28	IP27	IP26	IP25	IP24	IP23	IP22	IP21	IP20	IP19	IP18	IP17	IP16
Reset	These flags reflect the OR of the reset state of the individual interrupt status bits at the source unit.															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	IP15	IP14	IP13	IP12	IP11	IP10	IP9	IP8	IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0
Reset	These flags reflect the OR of the reset state of the individual interrupt status bits at the source unit.															

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read	FP31	FP30	FP29	FP28	FP27	FP26	FP25	FP24	FP23	FP22	FP21	FP20	FP19	FP18	FP17	FP16
Reset	These flags reflect the OR of the reset state of the individual interrupt status bits at the source unit.															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	FP15	FP14	FP13	FP12	FP11	FP10	FP9	FP8	FP7	FP6	FP5	FP4	FP3	FP2	FP1	FP0
Reset	These flags reflect the OR of the reset state of the individual interrupt status bits at the source unit.															

9.2.1.3 Interrupt Controller Mask Register (ICMR)

The interrupt controller mask register (ICMR) contains one mask bit per pending interrupt bit (32 total). The mask bits control whether a pending interrupt bit will generate a processor interrupt (IRQ or FIQ). When a pending interrupt becomes active, it is sent to the CPU only if its corresponding ICMR mask bit is set to a one. Note that the mask bits are ignored when the SA-1100 is in idle mode. While in idle, if any interrupt source makes a request, the corresponding pending bit is set and the interrupt automatically becomes active, regardless of the state of its mask bit.

Mask bits serve two purposes. First, they allow periodic software polling of interruptible sources while preventing them from actually causing an interrupt. Second, they allow the interrupt handler routine to prevent interrupts of lower priority from occurring while still maintaining a list of pending interrupts that may have occurred previously (or during the servicing of another interrupt). The ICMR is not initialized at reset; a question mark indicates that the values are unknown at reset.

The following table shows the bit locations corresponding to the 32 separate interrupt mask bits.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	Name	Description
{n}	IM{n}	Interrupt mask n (where n = 0 through 31). 0 – Pending interrupt is masked from becoming active (interrupts not sent to CPU, Power Manager). 1 – Pending interrupt is allowed to become active (interrupt sent to CPU, Power Manager). **Note**: IM bits are ignored during idle mode.

9.2.1.4 Interrupt Controller Level Register (ICLR)

The interrupt controller level register (ICLR) controls whether a pending interrupt generates an FIQ or an IRQ CPU interrupt. If a pending interrupt is unmasked, the corresponding ICLR bit field is decoded to select which CPU interrupt should be asserted. If the interrupt is masked, then the corresponding bit in the ICLR has no effect. The following table shows the location of all interrupt level bits in the ICLR; question marks indicate that the values are unknown at reset.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	IL31	IL30	IL29	IL28	IL27	IL26	IL25	IL24	IL23	IL22	IL21	IL20	IL19	IL18	IL17	IL16
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	IL15	IL14	IL13	IL12	IL11	IL10	IL9	IL8	IL7	IL6	IL5	IL4	IL3	IL2	IL1	IL0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	Name	Description
{n}	IL{n}	Interrupt level n (where n = 0 through 31). 0 – Interrupt routed to CPU IRQ interrupt input. 1 – Interrupt routed to CPU FIQ interrupt input.

9.2.1.5 Interrupt Controller Control Register (ICCR)

The interrupt controller control register (ICCR) contains a single control bit, the disable idle mask bit (DIM). When set, this bit inhibits the idle mode operation where the output of the ICMR is OR'ed to all ones. If this bit is set, then the interrupts that are capable of bringing the SA-1100 out of idle mode are defined by the contents of the ICMR. The following table shows the location of all interrupt level bits in the ICCR.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	Reserved															DIM
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
{0}	DIM	Disable idle mask. 0 – All enabled interrupts will bring the SA-1100 out of idle mode. 1 – Only enabled and unmasked (as defined in the ICMR) will bring the SA-1100 out of idle mode. This bit is cleared during all resets.
1..31	—	Reserved.

9.2.2 Interrupt Controller Register Locations

The following table shows the registers associated with the interrupt controller block and the physical addresses used to access them.

Address	Name	Description
0h 9005 0000	ICIP	Interrupt controller IRQ pending register
0h 9005 0004	ICMR	Interrupt controller mask register
0h 9005 0008	ICLR	Interrupt controller level register
0h 9005 0010	ICFP	Interrupt controller FIQ pending register
0h 9005 0020	ICPR	Interrupt controller pending register
0h 9005 000C	ICCR	Interrupt controller control register

9.3 Real-Time Clock

The SA-1100 contains a real-time clock (RTC) that provides a general-purpose real-time reference for use by the system. The RTC is uninitialized after a hardware reset (nRESET) and must be written by the user to the desired value. Thereafter, the counter will remain valid until another hardware reset (assumed to be infrequent). The value of the counter is unaffected by transitions into and out of sleep, idle, software reset, or a watchdog reset. The counter is incremented on rising edges of the 1-Hz clock.

In addition to the counter [RTC counter register (RCNR)], the RTC incorporates a 32-bit alarm register (RTAR). The RTAR may be programmed with a value to be compared against the counter. On each rising edge of the 1-Hz clock, the counter is incremented and then compared to the RTAR. If the values match, then a status bit is set. This status bit is also routed to the interrupt controller and may be programmed to generate a CPU interrupt.

Another interruptible status bit is available that is set whenever the 1 Hz clock ticks. Each status bit may be cleared by writing a one to the status register in the desired bit position. The 1-Hz clock is generated by dividing down the 32.768-kHz crystal oscillator output. This divider logic is programmable to allow the user to “trim” the counter to adjust for inherent inaccuracies in the crystal’s frequency. This trimming mechanism permits the user to adjust the RTC to an accuracy of +/- 5 seconds per month. The trimming procedure is described later in this section.

9.3.1 RTC Counter Register (RCNR)

The RTC counter register (RCNR) is a read/write register and is not cleared by any reset source. The counter may be written by the processor at any time although it is recommended that the operating system prevent inadvertent writes to the RCNR through the use of the MMU protection mechanisms.

Because of the asynchronous nature of the 1-Hz clock relative to the processor clock, writes to this counter are controlled by a hardware mechanism that delays the actual write to the counter by up to one 32-kHz-clock (~ 30 µs) after the processor store is performed.

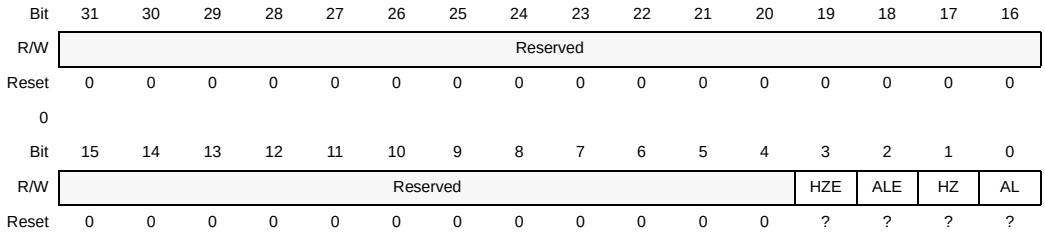
After the processor writes to the RCNR, all other writes to this register location are ignored until the new value is actually loaded into the counter. The RCNR may be read at any time. Reads reflect the value in the counter immediately after it increments or loads.

9.3.2 RTC Alarm Register (RTAR)

The real-time clock alarm register is a 32-bit register that is readable and writable by the processor. Following each rising edge of the 1-Hz clock, this register is compared to the RCNR. If the two are equal and the enable bit is set, then the alarm bit in the RTC status register is set. The value in this register is undefined after the assertion of nRESET.

9.3.3 RTC Status Register (RTSR)

The following table shows the location of all bits in the RTSR. All reserved bits are read as zeros and are unaffected by writes; a question mark indicates that the value is unknown at reset. The AL and HZ bits in this register are routed to the interrupt controller where they may be enabled to cause an interrupt. The AL and HZ bits are cleared by writing ones to them.



Bit	Name	Description
0	AL	RTC alarm detected. 0 – No alarm has been detected. 1 – An alarm has been detected (RTNR matches RCAR).
1	HZ	1-Hz rising-edge detected. 0 – No rising edge has been detected. 1– A rising edge has been detected.
2	ALE	RTC alarm interrupt enable. 0 – The RTC alarm interrupt is not enabled. 1 – The RTC alarm interrupt is enabled.
3	HZE	1-Hz interrupt enable. 0 – The 1-Hz interrupt is not enabled. 1 – The 1-Hz interrupt is enabled.
31..4	—	Reserved.

9.3.4 RTC Trim Register (RTTR)

The RTTR is programmed by the user to select the frequency of the 1-Hz clock. If this register is not programmed and left at its reset value (all zeros), then the 1-Hz clock will actually be running at 32.768 kHz. See the following section for details on how to calculate the value in this register. The following table shows the location of all bits in the RTTR. All reserved bits are read as zeros and are unaffected by writes.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	Reserved						D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0..15	C0-C15	Clock divider count. This value is the integer portion of the clock trim logic.
16..25	D0-9	Trim delete count. This value represents the number of 32-kHz clocks to delete when clock trimming begins.
26..31	—	Reserved.

9.3.5 Trim Procedure

The 1-Hz clock feeding the RTC is obtained by dividing the output of the 32.768-kHz oscillator down. Since 32768 is a power of two, a 15-bit divider will generate a 1-Hz clock (given a perfect crystal and perfect board environment). The inherent inaccuracies of crystals, aggravated by varying capacitance of the board connections, and so on, cause the timebase to be somewhat inaccurate, requiring a periodic adjustment in the 1 Hz clock period. The SA-1100, through the RTTR, allows the user to adjust or "trim" the 1 Hz timebase to an accuracy of +/- 5 seconds per month. At reset, the RTTR contains zeros that disable the trim circuitry. When the trim circuitry is disabled, the 1-Hz clock feeding the RTC is the same frequency as the output of the 32.768-kHz oscillator. The RTTR is reset to all zeros each time the nRESET signal is asserted.

9.3.5.1 Oscillator Frequency Calibration

To generate the value to be entered into the RTTR, the user must first measure the output frequency of the 32.768-kHz oscillator using an accurate timebase, such as a frequency counter. This clock is made externally visible by selecting the alternate function for GPIO<27>. To gain access to the clock, this pin must be programmed as an output and then switched over to the alternate function. See the Section 9.1, “General-Purpose I/O” on page 9-1 in this chapter for details on how to gain access to the clock. The trim is accomplished by dividing the output of the oscillator by an integer value and then doing fine-grain fractional adjustment by periodically deleting clocks from the stream feeding this integer divider.

9.3.5.2 RTTR Value Calculations

After the true frequency of the oscillator is known, it must be split into integer and fractional portions. The integer portion of the value (minus one) is loaded into the C0-C15 field of the RTTR. This value is compared against a 16-bit counter clocked by the output of the 32.768-kHz oscillator. The counter resets and generates a pulse when the two values are equal. This pulse constitutes the raw 1-Hz signal.

The fractional part of the adjustment is done by periodically deleting clocks from the clock stream feeding the integer counter. The period, called the "trim interval," is hardwired to be 2^{10} -1 seconds (approximately 17 minutes). The number of clocks deleted, called the "trim delete value," is a 10-bit programmable counter allowing from 0 to 2^{10} -1 32-kHz clocks to be deleted from the input clock stream once per trim interval. D0-D9 represents the number of clocks deleted per trim operation. In summary, every 2^{10} -1 seconds, the integer counter stops clocking for a period equal to the fractional error that has accumulated. If this counter is programmed to a zero (as it is at a hard reset), then no trim operations will occur and the RTC will be clocked with the raw 32.768-kHz clock. The relationship between the nominal 1-Hz clock frequency and the nominal 32.768-kHz clock (f1 and f32K respectively) is shown in the following equation.

$$f1 = \frac{(2\text{^}10\text{-}1)*(C<15..0> +1) - D<9..0>}{(2\text{^}10\text{-}1)*(C<15..0> +1)} * \frac{f32k}{(C<15..0> +1)}$$

Trim Example #1 – Measured Value Has No Fractional Component

In this example, the oscillator output is measured to be 36045.000 cycles/s (Hz). This output is exactly 3277 cycles over the nominal frequency of the crystal and has no fractional component. As such, only the integer trim function is needed and no fractional trim is required. Accordingly, the C0-C15 field of the RTTR is loaded with the binary equivalent of 36045-1, or 0x8CCC. The D0-D9 field is left at zero (power-up state) to disable fractional trimming. This trim exercise leaves an error of zero in trimming.

Trim Example #2 – Measured Value Has a Fractional Component

This example is a more common case in that the measured frequency of the oscillator has a fractional component. If the oscillator output is measured to be 32768.92 cycles/s (Hz), an integer trim is necessary so that the *average* number of cycles counted before generating one 1-Hz clock is 32768.92. Similar to the previous example, the integer field D0-D15 is loaded with the hexadecimal equivalent of 32768-1 or 0x7FFF.

Because the actual clock frequency is 0.92 cycles per second faster than the integer value, the 1-Hz clock generated by just the integer trimming is slightly *faster* than needed and must be slowed down. Accordingly, the fractional trim must be programmed to delete 0.92 cycles per second on average to bring the 1-Hz output frequency down to the proper value. Since the trimming procedure is performed only every 2^{10}-1=1023 seconds, the trim must be set to delete (.92*1023) = 941.16 clocks every 1023 seconds. The fractional component of this value cannot be trimmed out and constitutes the error in trimming, described below. The counter should be loaded with the hexadecimal equivalent of 941, or 0x3AD.

This trim setting leaves an error of .16 cycles per 1023 seconds. The error calculation yields (in parts-per-million or ppm):

$$\text{Error} = \frac{0.16 \text{ cycles}}{1023 \text{ sec}} X \frac{1 \text{ sec}}{32768 \text{ cycles}} = 0.002 \text{ ppm}$$

Maximum Error Calculation Versus Real-Time Clock Accuracy

As seen from trim example #2, the maximum possible error approaches 1 clock per 2^{10}-1 seconds. Calculating the ppm error for this scenario yields:

$$\text{Error (maximum)} = \frac{1 \text{ cycle}}{1023 \text{ sec}} X \frac{1 \text{ sec}}{32768 \text{ cycles}} = 0.03 \text{ ppm}$$

To maintain an accuracy of +/- 5 seconds per month, the required accuracy is calculated to be:

$$\text{Error} = \frac{5 \text{ sec}}{\text{month}} X \frac{1 \text{ month}}{2592000 \text{ sec}} = 1.9 \text{ ppm}$$

This calculation indicates that the accuracy of the SA-1100 trim mechanism is more than adequate to compensate for the static environmental and manufacturing variables, and still provides acceptable accuracy.

9.3.6 Real-Time Clock Register Locations

The following table describes the real-time clock registers.

Address	Name	Description
0h 9001 0004	RCNR	RTC count register
0h 9001 0000	RTAR	RTC alarm register
0h 9001 0010	RTSR	RTC status register
0h 9001 0008	RTTR	RTC timer trim register

9.4 Operating System Timer

The SA-1100 contains a 32-bit operating system timer that is clocked by the 3.6864-MHz oscillator. The operating system count register (OSCR) is a free-running up-counter that is not cleared during any reset (contains unknown value after reset). The OS timer also contains four 32-bit match registers (OSMR<3:0>). Each register can be written and read by the user. When the value in the OSCR matches (is equal to) the value within any of the match registers, and the interrupt enable bit is set, the corresponding bit in the OSSR is set. These bits are also routed to the interrupt controller where they can be programmed to cause an interrupt. OSMR<3> also serves as a watchdog match register that resets the SA-1100 when a match occurs. The only register that is reset to a known state is the watchdog match enable register (WMER). The user must initialize all other registers and clear any set status bits before the FIQ and IRQ interrupts are enabled within the CPU.

9.4.1 OS Timer Count Register (OSCR)

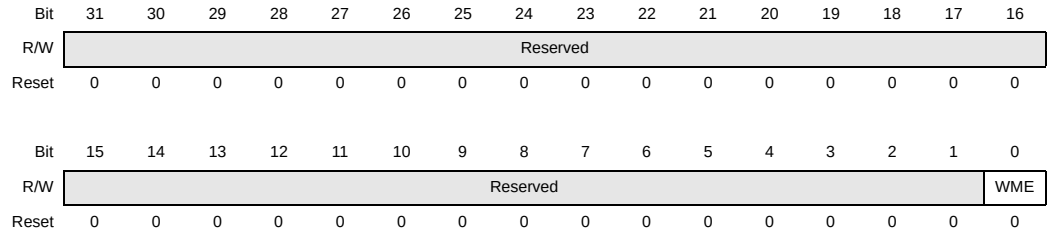
The OS timer count register is a 32-bit counter that increments on rising edges of the 3.6864-MHz clock. This counter can be read or written at any time. It is recommended that the system write-protect this register through the MMU protection mechanisms.

9.4.2 OS Timer Match Registers 0–3 (OSMR<0>, OSMR<1>, OSMR<2>, OSMR<3>)

These registers are 32 bits wide and are readable and writable by the processor. They are compared against the OSCR following every rising edge of the 3.6864-MHz clock. If any of these registers match the counter at this time, then the corresponding status bit in the OSSR is set. The status bits are routed to the interrupt controller where they can be unmasked to cause a CPU interrupt. OSMR<3> may also serve as a watchdog timer. See the Section 9.4.6, “Watchdog Timer” on page 9-24 for operation information.

9.4.3 OS Timer Watchdog Match Enable Register (OWER)

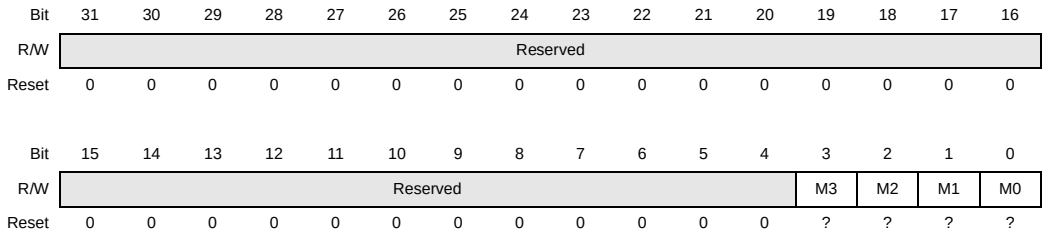
The watchdog enable register contains a single control bit (bit 0) that enables the watchdog function. This bit is set by writing a one to it. It can only be cleared by one of the reset functions (hardware reset, software reset) and by entering sleep mode. A watchdog reset also clears the watchdog enable bit. The format of this register follows:



Bit	Name	Description
0	WME	Watchdog match enable. 0 – OS timer match register<3> matches cause an interrupt request. 1 – OS timer match register<3> matches cause a reset of the SA-1100. **Note**: This is a write-once bit that once written, can only be changed after a hardware (pin), software (SWR), or sleep mode reset.
31..1	—	Reserved.

9.4.4 OS Timer Status Register (OSSR)

This status register contains status bits indicating whether a match has occurred on any of the four match registers. These bits are set when the event occurs (following the rising edge of the 3.6864-MHz clock) and cleared by writing a one to the proper bit position. Writing zeros to this register has no effect. All reserved bits read as zeros and are unaffected by writes; a question mark indicates that the value is unknown at reset.



Bit	Name	Description
0	M0	Match status channel 0. 0 – OS timer match register<0> has not matched the OS timer counter since the last clear. 1 – OS timer match register<0> has matched the OS timer counter.
1	M1	Match status channel 1. 0 – OS timer match register<1> has not matched the OS timer counter since the last clear. 1 – OS timer match register<1> has matched the OS timer counter.
2	M2	Match status channel 2. 0 – OS timer match register<2> has not matched the OS timer counter since the last clear. 1 – OS timer match register<2> has matched the OS timer counter.
3	M3	Match status channel 3. 0 – OS timer match register<3> has not matched the OS timer counter since the last clear. 1 – OS timer match register<3> has matched the OS timer counter.
31..4	—	Reserved.

9.4.5 OS Timer Interrupt Enable Register (OIER)

This register contains four enable bits indicating whether a match between one of the match registers and the OS timer counter will set a status bit in the OSSR. Each match register has a corresponding enable bit. Clearing an enable bit does not clear the corresponding interrupt status bit if that bit is already set.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	Reserved												E3	E2	E1	E0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	E0	Interrupt enable channel 0. This bit is set by software and allows a match between match register 0 and the OS timer to assert interrupt bit M0 in the OSSR.
1	E1	Interrupt enable channel 1. 0 – OS timer match register<1> has not matched the OS timer counter since the last clear. 1 – OS timer match register<1> has matched the OS timer counter.
2	E2	Interrupt enable channel 2. 0 – OS timer match register<2> has not matched the OS timer counter since the last clear. 1 – OS timer match register<2> has matched the OS timer counter.
3	E3	Interrupt enable channel 3. 0 – OS timer match register<3> has not matched the OS timer counter since the last clear. 1 – OS timer match register<3> has matched the OS timer counter.
31..4	—	Reserved.

9.4.6 Watchdog Timer

OSMR<3> may also serve as a watchdog compare register. This function is enabled by setting bit 0 in the OWER. When a compare against this register occurs when the watchdog is enabled, reset is applied to the SA-1100 and most internal states are cleared (with exceptions listed below). Internal reset is asserted for 256 processor clocks and then removed, allowing the SA-1100 to boot. Units that do not receive this internal reset are: the power manager, the refresh timer, and the PLL configuration. Watchdog reset affects the SA-1100 similar to a software reset. See the Section 9.6, “Reset Controller” on page 9-41 for details on what is affected by each kind of reset. When the SA-1100 comes out of a watchdog reset, a bit is set in the reset controller status register (RCSR) to indicate that the event happened.

The following procedure is suggested when using OSMR<3> as a watchdog: each time the operating system services the register, the current value of the counter is read, and a number is then added to the value read, corresponding to the amount of time before the next timeout (care must be

taken to account for counter wraparound). This number is then written back to OSMR<3>. The OS code must repeat this procedure periodically before each match occurs. If the match occurs, the OS timer will assert a reset.

9.4.7 OS Timer Register Locations

Table 9-1 shows the registers associated with the OS timer and the physical addresses used to access them.

Table 9-1. OS Timer Register Locations

Address	Name	Description
0h 9000 0000	OSMR<0>	OS timer match registers<3:0>
0h 9000 0004	OSMR<1>	
0h 9000 0008	OSMR<2>	
0h 9000 000C	OSMR<3>	
0h 9000 0010	OSCR	OS timer counter register
0h 9000 0014	OSSR	OS timer status register
0h 9000 0018	OWER	OS timer watchdog enable register
0h 9000 001C	OIER	OS timer interrupt enable register

9.5 Power Manager

The SA-1100 contains power management logic that controls the transition between three different modes of operation: run, idle, and sleep. These modes are used to reduce processor power consumption at times when some functions are not needed, or when the system's power supply is low or out of regulation. Each of the respective modes is associated with a reduced level of power consumption. Idle mode is entered via software. Sleep mode is entered either via software or by asserting one of two input pins that indicate a power supply fault. Idle mode is exited through an interrupt. Sleep mode is exited through a preprogrammed wake-up condition. Both modes may be exited in extreme cases via hardware reset. If none of the power management modes is active and the SA-1100 is out of reset, then it is said to be in run mode.

9.5.1 Run Mode

Run mode is the normal operating mode of the SA-1100: all power supplies are enabled, all clocks are running, and every on-chip resource is functional. This is the normal state of operation for the processor while it is executing code. Under usual conditions, the processor enters run mode after successful power-up and reset of the part.

9.5.2 Idle Mode

Idle mode allows a software application to stop the CPU when not in use, while continuing to monitor interrupt service requests both on or off-chip. When an interrupt occurs, the CPU is reactivated. During idle mode, the SCM, PM, and MPCM are each fully operational.

In idle mode, the CPU clock is stopped. Since the SA-1100 is static, all CPU state information is saved. This allows the part to be switched back to run mode, starting operation exactly where it left off. During idle mode, all other on-chip resources are active, including: all system unit modules (real-time clock, operating system timer, interrupt controller, general-purpose I/O, and power manager); all peripheral unit modules (DMA controller, LCD controller, serial controller 0-4); and all memory controller resources. The PLL also remains in lock so that the part can be brought out of idle mode quickly when an interrupt occurs.

9.5.2.1 Entering Idle Mode

Idle mode is entered while in run mode by executing a three instruction sequence consisting of the privileged on-chip coprocessor 15 instruction 'disable clock switching', a load from a noncacheable memory location (C=B=0), and the privileged on-chip coprocessor 15 instruction 'wait for interrupt'. This sequence must reside in the first three words of an instruction cache line, which requires that the linker align the idle mode instruction sequence on an eight word boundary. Idle mode is entered by following the exact code sequence:

```
AREA |Idle$$Code|, CODE, READONLY, ALIGN=5
                                ;Aligned to 8 word boundary
                                ;p15 = coprocessor 15
                                ;r0  = register 0 (contents not used)
                                ;c15 = test, clk, and idle cntl register
                                ;c2 = CRm = 0b0010
mcr  p15, 0, r0, c15, c2, 2     ;2   = OPC_2 = 0b010
ldr  r0, <r1>                   ;<r1> points to non-cachable mem loc
mcr  p15, 0, r0, c15, c8, 2     ;c8  = CRm = 0b1000
```

9.5.2.2 Exiting Idle Mode

Any enabled interrupt from the system unit or peripheral unit will cause a transition from idle mode back to run mode. Note that the interrupt controller (ICMR) mask register is ignored during idle mode, meaning that an interrupt does not need to be unmasked to bring the SA-1100 out of idle. When an interrupt occurs, the CPU clocks are reactivated, the wait for interrupt instruction is completed, and run program flow resumes.

A transition from idle to run mode can also occur by asserting the nRESET pin or if OSMR<3> is configured as a watchdog and a match occurs that causes the assertion of reset. Since the watchdog timer (if enabled) is functional during idle, care must be taken to set the watchdog match register far enough in advance to ensure that another interrupt is guaranteed to bring the SA-1100 out of idle before the watchdog reset occurs. It is recommended that either an RTC alarm or another OS timer channel be used for this purpose.

When in idle mode, if the BATT_FAULT and/or VDD_FAULT pins are asserted, the SA-1100 enters sleep mode.

9.5.3 Sleep Mode

Sleep mode offers the greatest power savings to the user and consequently the lowest level of available functionality. In the transition from run or idle to sleep mode, the SA-1100 performs an orderly shutdown of on-chip activity, applies an internal reset to the processor, and then negates the PWR_EN pin indicating to the external system that the VDDI (1.5-V supply) should be driven to zero volts. Internally, this switches off the power to the majority of the processor at this time. (The VDDX I/O voltage supply must remain powered during sleep.) Running off the 32.768-kHz crystal oscillator, the sleep state machine watches for a preprogrammed wake-up event to occur, after which it asserts PWR_EN (to reestablish the VDDI power supply), and steps through an orderly wake-up sequence. When the power supply and clocks are stable, the power manager brings the SA-1100 out of reset. Status bits in the power manager GPIO sleep state register (PGSR) may be read to indicate to software that the reset was due to sleep mode.

9.5.3.1 CPU Preparation for Sleep Mode

In preparation for sleep mode, software should initialize the power manager GPIO sleep state register (PGSR) and the power manager wake-up enable register (PWER). Also, the GPIO falling-edge detect and GPIO rising-edge detect enable registers (GFER and GRER) should be written with the appropriate values. The OPDE bit in the power manager configuration register (PCFR) should also be programmed with the desired value.

9.5.3.2 Events Causing Entry into Sleep Mode

Sleep mode can be entered in one of two ways: via software or a power supply fault. Entry into sleep mode via software is accomplished by setting the force sleep bit in the power manager control register (PMCR). This bit is set by software and cleared by hardware during sleep. When the SA-1100 wakes up from sleep, this bit is already cleared.

Entry into sleep via a power supply fault is caused by the assertion of either the VDD_FAULT or BATT_FAULT pins. The VDD_FAULT pin should be used to indicate that the main power supply is out of regulation. The BATT_FAULT pin should be used to indicate that the battery has been removed or is low. These pins have identical operation for the purpose of entering sleep mode. They have different implications during the wake-up sequence as described in the following section.

9.5.3.3 The Sleep Shutdown Sequence

The sleep state machine begins the shutdown sequence. This sequence consists of three steps.

- In the first step, the following actions occur:
  a. Power manager switches the GPIO output pins to their sleep state. This sleep state is programmed in advance by loading the power manager GPIO sleep state register (PGSR) into the GPIO output data register. (See the Section 9.1, “General-Purpose I/O” on page 9-1.)
  b. The DRAMs are placed into self-refresh mode. The memory controller finishes whatever memory operation might be in progress and then drives the RAS<3:0> and CAS<3:0> pins low.
  c. If the sleep sequence was entered due to the assertion of VDD_FAULT or BATT_FAULT, the possible wake-up sources are reset from what was programmed by software to their "fault state". The fault state is to allow a transition only on GP<0> and GP<1> to act as a wake-up event.
- In the second step of sleep shutdown, the following actions occur:
  a. All potential wake-up sources are cleared. This involves clearing all the GPIO edge detect status bits and clearing the RTC alarm interrupt bit. These bits are cleared to prevent latent status bits from causing an immediate wake-up. This functionality is provided to cover the situation of entering sleep due to a power fault because the CPU does not have the ability to prepare for the entry into sleep.
  b. An internal reset is applied to the SA-1100. All units are reset and the RESET_OUT pin is asserted.
- In the third step of sleep shutdown, the following actions occur:
  a. The 3.686-MHz oscillator is stopped. This action is dependent on the state of the oscillator power-down enable bit (OPDE) in the power manager configuration register (PCFR). If this bit is set, then the oscillator is stopped during sleep, resulting in greater power savings. If the bit is cleared (the power-on reset state), then the oscillator continues to run during sleep and results in a faster wake-up sequence.
  b. The PWR_EN pin is negated. The external system must respond to this negation by disabling the VDDI power supply. In contrast to the SA-110, the SA-1100 systems are not required to drive VDDI to zero volts in sleep. However, the power supply should be disabled to prevent power consumption.

Each step in the sleep shutdown sequence takes one cycle of the 32.768-kHz clock (~30 microseconds).

9.5.3.4 During Sleep Mode

During sleep mode, the SA-1100 watches for preprogrammed wake-up events. These events are either programmed by the CPU prior to setting the force sleep bit or by the power manager when a fault condition is detected.

9.5.3.5 The Sleep Wake-Up Sequence

When a valid wake-up event is detected and there is no BATT_FAULT, the SA-1100 begins a wake-up sequence. If BATT_FAULT is asserted, then the wake-up event is ignored. VDD_FAULT is always ignored at this time because the VDDI supply is disabled at this time. The wake-up sequence occurs in three steps.

- In the first step of the wake-up sequence, the following actions occur:
  a. The PWR_EN pin is asserted, indicating that the external supply must apply power on the VDDI pins.
  b. An internal timer begins to time the power ramp. This timer waits for approximately 10 ms.
  c. The 3.686-MHz oscillator is enabled for operation if it was originally programmed to be disabled.
  d. If BATT_FAULT is asserted at any time during the sleep wake-up sequence, the power manager transitions back to sleep mode through the fault state.
- In the second step of the wake-up sequence (after the power ramp timer has expired), the following actions occur:
  a. A second internal timer begins to time the 3.686-MHz oscillator as it begins to ramp up to speed. This timer waits for 150 ms. If the OPDE bit in the PCFR is zero, then the oscillator was never disabled and this timer is not used. In this case, the power manager transitions to the third step directly without waiting for the oscillator timer to complete.
  b. If BATT_FAULT or VDD_FAULT is asserted at any time during the oscillator ramp, the power manager transitions back to sleep mode through the fault state.
- In the third step of the wake-up sequence (after the 3.6864-MHz oscillator is stabilized), the following actions occur:
  a. The SA-1100 internal reset is negated and the CPU begins a normal boot sequence.
  b. The RESET_OUT pin is negated, indicating that the SA-1100 is about to perform a fetch from the reset vector location.

During the fault state entered through the assertion of VDD_FAULT or BATT_FAULT, the following actions occur:

- All potential wake-up sources are cleared (all GPIO edge detects and the RTC alarm interrupt).
- The power manager wake-up source register (PWER) is loaded with 0x0000 0003 and bits 0 and 1 of the GFER and the GRER (see the Section 9.1, “General-Purpose I/O” on page 9-1) are set. This limits the potential wake-up sources to a rising or falling edge on GP<0> or GP<1>. This wake-up fault state is provided to prevent spurious events from causing an unwanted wake-up during a low battery or shorted power supply situation. This fault state setting of PWER, GRER, and GFER registers is also the default state of the registers after a hardware reset.

9.5.3.6 Booting After Sleep Mode

When the SA-1100 boots after sleep mode (or at any other time), it must examine the reset controller status register (RCSR) to determine why it just booted. This register has bits to indicate sleep reset, software reset, watchdog reset, or hardware reset (assertion of nRESET). See the Section 9.6, “Reset Controller” on page 9-41 for more details on reset.

Next, software should examine the power manager sleep status register (PSSR) to determine why it was in sleep. This register has bits to indicate whether a VDD_FAULT, BATT_FAULT, or force sleep bit has been asserted since the register was last cleared. It is possible for multiple bits to be set in this register.

Also, the SA-1100 provides the power manager scratchpad register (PSPR) for saving any general processor state during sleep. This register may be written by the processor and the contents will survive sleep mode. The bits in this register are not explicitly used by the SA-1100, but may be used by software to index into ROM space to retrieve memory controller configuration, for example.

Note: The nRESET pin must not be asserted during sleep mode if the DRAM contents are to be preserved. The assertion and subsequent negation of nRESET during sleep mode causes the SA-1100 to clear the FS bit in the force sleep register, assert PWR_EN, time the PLL lock sequence, and subsequently negate the internal reset signal. This causes the SA-1100 to perform a normal boot sequence because all information about the previous sleep state is lost.

9.5.3.7 Reviving the DRAMs from Self-Refresh Mode

Because the DRAMs are placed in self refresh prior to the sleep mode shutdown, their contents are preserved during sleep. After exiting sleep, software must reconfigure the DRAM control registers, which lost power during sleep mode, and then take the DRAMs out of self-refresh mode. Clearing the DRAM hold (DH) bit in the power management status register (PMSR) will cause the RAS<3:0> and CAS<3:0> pins to return to the negated state (high) in preparation for a DRAM access.

9.5.4 Notes on Power Supply Sequencing

On the SA-1100, as on the SA-110, it is important that VDDX (3.3 V nominal) power-up occur before VDDI (1.5 V nominal). One approach to ensuring this sequencing is to power the 1.5-V supply using the 3.3-V supply. On the SA-1100, a second simple option is available. If the PWR_EN output is used to enable the 1.5-V supply, the SA-1100 will enforce the required sequencing by holding PWR_EN deasserted until the 3.3-V supply is sufficiently high.

9.5.5 Assumed Behavior of an SA-1100 System in Sleep Mode

The assumed model of an SA-1100 system in sleep mode is one in which the system is relatively quiet. In particular, there should be no gratuitous switching on of the SA-1100 input pins. Although there will be some switching in GPIOs to bring the processor out of sleep and potentially on the VDD_FAULT and BATT_FAULT pins, the switching is a low-frequency activity and usually brings the SA-1100 out of sleep mode.

The major concern is for power dissipation in sleep and requirements for the power supplies on the processor during sleep. The SA-1100 generates these supplies using several on-chip regulators with limited current capacity. Excessive activity on-chip pins might load these regulators beyond their capacity and result in droop of the on-chip supplies. One example is that of a component tied to one of the GPIO pins that constantly transmits to the processor. If the system design indicated that activity from this detector should not bring the SA-1100 out of sleep, the transitions from this GPIO might result in switching in the processor that would exceed the sleep current limit. This concern exists regardless of whether the GPIO is enabled as a wake-up source.

Figure 9-3 shows the three power-related modes of the SA-1100 and the actions that cause transitions between the modes. Table 9-2 summarizes what power and clock supplies are used by each module within the SA-1100, as well as the status of the power and clock supplies to each unit during each of the three power-related modes.

Figure 9-3. Transitions Between Modes of Operation

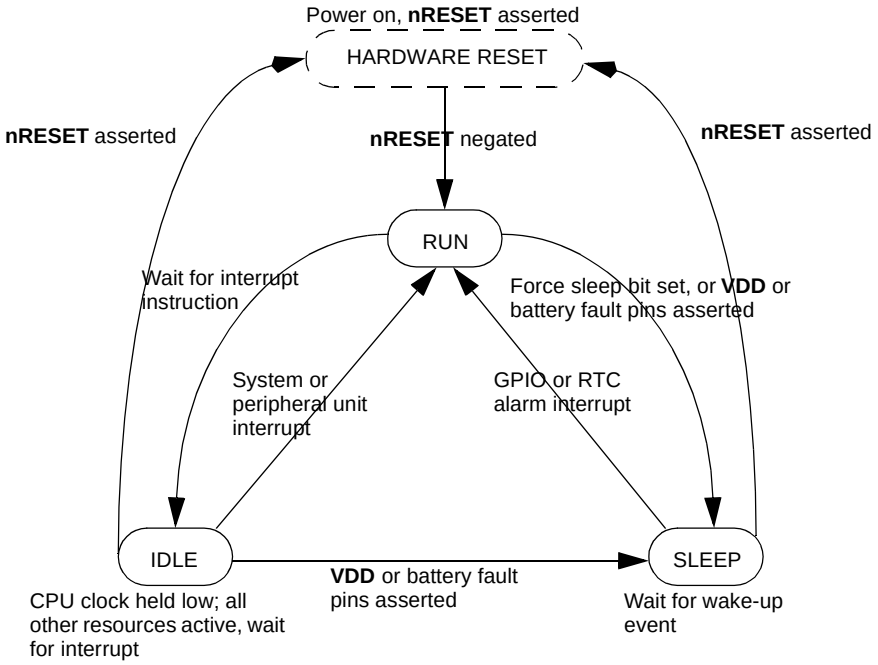


Table 9-2. SA-1100 Power and Clock Supply Sources and States During Power-Down Modes

			Power Management Mode					
Module	**Supply Source**		**Run**		**Idle**		**Sleep**	
	Pwr	**Clk**	**Pwr**	**Clk**	**Pwr**	**Clk**	**Pwr**	**Clk**
CPU	VDD	3.6864 MHz	On	Running	On	Stopped	Disabled	Stopped
MMUs (I&D)								
Write buffer								
Read buffer								
JTAG						Running		
OS timer								
LCD controller								
Serial channel 0-4								
Memory and PCMCIA control								
Real-time clock	VDDX	32.768 kHz					On	Running
Interrupt controller								
Power manager								
General-purpose I/O								
Pin pads								

9.5.6 Pin Operation in Sleep Mode

The SA-1100 pins are categorized by the following types based on their behavior during sleep mode:

Type 1 – These pins are outputs and are driven low during sleep. These pins hold their state after sleep mode is exited until the DRAM_control_hold bit in the PSSR is cleared.

Type 2 – These pins are outputs and are normally driven to a one in sleep. To support systems that power down external devices, these pins can also be tristated in sleep through the use of the FLOAT_STATIC and FLOAT_PCMCIA bits in the PCFR. See the Section 9.5, “Power Manager” on page 9-26.

Type 3 – These pins are I/Os. When programmed as outputs, they can be actively held high or low during sleep. When programmed as inputs, they are actively sampled by the SA-1100.

Type 4 – These pins are I/Os but become inputs during sleep. They can be programmed to hold the pin state at a zero or can be tristated. The receivers on these pins are disabled during sleep. These pins hold their state after sleep mode is exited until either the peripheral_control_hold bits in the PSSR are cleared.

Type 5 – These pins are outputs and are actively driven during sleep.

Type 6 – These pins are outputs and are tristated during sleep.

Type 7 – These pins are inputs and are actively sampled during sleep.

Type 8 – These pins are inputs and are not observed during sleep; the receiver is disabled.

Type 9 – These pins are analog inputs and outputs, and are always active.

Table 9-3. Pin State During Step

Pin Name	Type	Pin Name	Type	Pin Name	Type	Pin Name	Type
A<25:0>	1	nPREG	1	RXD_2	4	nRESET_OUT	1
D<31:0>	1	L_DD<7:0>	4	TXD_3	4	nTRST	8
nCS<3:0>	2	L_FCLK	4	RXD_3	4	TDI	8
nOE	2	L_BIAS	4	GP<27:0>	3	TDO	6
nWE	2	TXD_C	4	ROM_SEL	8	TMS	8
nRAS<3:0>	1	RXD_C	4	PXTAL	9	TCK	8
nCAS<3:0>	1	SCLK_C	4	PEXTAL	9	TCK_BYP	7
nPIOW	2	SFRM_C	4	TXTAL	9	TESTCLK	7
nPIOR	2	UDC+	4	TEXTAL	9	VDD	—
nPCE<2:1>	2	UDC-	4	PWR_EN	5	VDDX	—
nIOIS16	2	TXD_1	4	BATT_FAULT	7	VSS	—
nPWAIT	2	RXD_1	4	VDD_FAULT	7	VSSX	—
PSKTSEL	1	TXD_2	4	nRESET	7	—	—

9.5.7 Power Manager Registers

The power manager is controlled through eight 32-bit registers. The power manager control register (PMCR) is used to allow software invocation of sleep mode. The sleep status register (PSSR) contains status bits that indicate why sleep mode was invoked. The power manager scratchpad register (PSPR) is a general-purpose register used to store processor data during sleep. The power manager wake-up enable register (PWER) is used to program the desired wake-up sources in the system. The power manager general configuration register (PCFR) contains bits used to control various configurable functions within the SA-1100. The power manager PLL configuration register (PPCR) allows the user to change the PLL operating frequency. The power manager GPIO sleep state register (PGSR) is used to program the value loaded onto GPIO outputs when the SA-1100 transitions into sleep mode. The power manager oscillator status register (POSR) contains a single bit that indicates whether the 32.768-kHz oscillator has stabilized after a hardware reset.

9.5.7.1 Power Manager Control Register (PMCR)

Sleep mode is invoked by setting the force bit within the power manager control register (PMCR). The force bit is automatically cleared upon exiting sleep mode or when a hardware reset occurs. Writing zero to the force bit has no effect. For reserved bits, writes are ignored and reads return zero. This register should be protected by programming MMU permissions. The following table shows the PMCR.

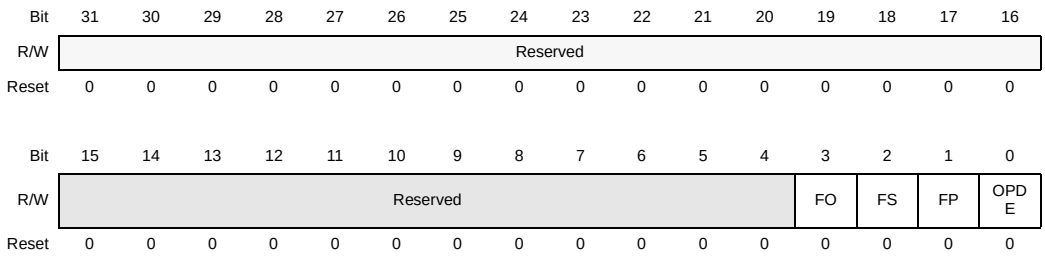
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	Reserved															SF
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	SF	Sleep force. 0 - Do not force invocation of sleep mode. 1 - Force invocation of sleep mode. **Note:** This bit is cleared on wake-up or a hardware reset.
31..1	—	Reserved.

9.5.7.2 Power Manager General Configuration Register (PCFR)

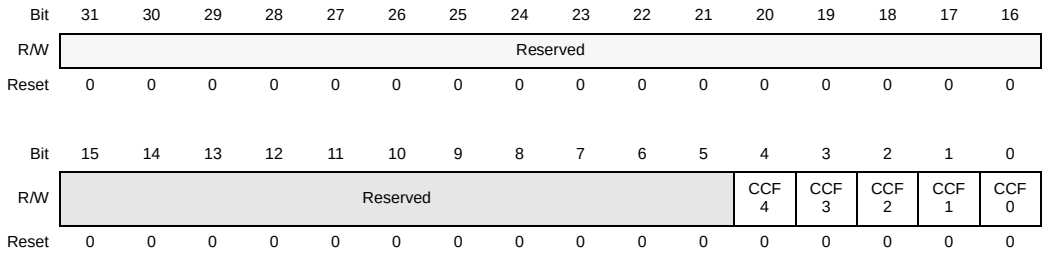
The PCFR contains bits used to configure various functions within the SA-1100. The OPDE bit, if set, allows the 3.6864-MHz oscillator to be disabled during sleep mode. This bit is cleared on the assertion of nRESET. The FP and FS bits control the state of the PCMCIA control pins and the static memory control pins during sleep. The following table shows the bit-field definitions for this register. The FO bit forces the SA-1100 to assume that the 32-kHz oscillator is stable instead of waiting for the requisite 2–10 seconds using an internal counter. This function is primarily useful for "warm" hardware resets where the oscillator is already stable when the processor comes out of reset.



Bit	Name	Description
0	OPDE	3.6864-MHz oscillator power-down enable. 0 – Do not stop the oscillator during sleep mode (reset condition). 1 – Stop the 3.6-MHz oscillator during sleep mode.
1	FP	Float PCMCIA controls during sleep mode. This bit determines whether the PCMCIA control signals are driven to a high (negated) state during sleep or not driven (floated). A zero indicates that the pins are driven low. A one indicates that they will be floated. This bit is zero at hardware reset. The PCMCIA signals affected by this bit are: **nPOE**, **nPWE**, **nPIOW**, **nPIOR**, **nPCE<2:1>**, **nIOIS16**, and **nPWAIT**. **PSKSEL** and **nPREG** are derived from address signals and assume the state of the address bus during sleep.
2	FS	Float static chip selects during sleep mode. This bit determines whether the static chip select control signals are driven to a high during sleep or floated. A zero indicates that the pins are driven low. A one indicates that they will be floated. The static chip select signals affected by this bit are: **nCS<3:0>**, **nOE**, and **nWE**. This bit is zero at hardware reset.
3	FO	Force 32-kHz oscillator enable on. This bit is used to allow software to force the SA-1100 to use the 32-kHz oscillator for internal clocking functions instead of waiting for it to stabilize in the normal way. This function is useful primarily to attain rapid functionality after a "warm" hardware reset when it is known that the oscillator is stable. Use of this bit is intended for test purposes and *some* customer use in special situations. It should be used with care, however, since setting this bit when the 32-kHz oscillator is not stable will yield unpredictable results.
31..4	—	Reserved.

9.5.7.3 Power Manager PLL Configuration Register (PPCR)

The PPCR contains bits used to configure the core operating frequency generated by the PLL. The following table shows the bit-field definitions for this register. See Chapter 8, “Clocks” for the frequencies generated through settings in this register. Note that the contents of this register are preserved during sleep mode and do not need to be re-initialized after a wake-up event. The PPCR is only cleared upon the assertion of nRESET (hard reset).



Bit	Name	Description
4-0	CCF<4:0>	Clock speed configuration. See Chapter 8, “Clocks” for the values in this field.
31..5	—	Reserved.

9.5.7.4 Power Manager Wake-Up Enable Register (PWER)

The following table shows the location of all wake-up interrupt enable bits in the PWER. For a GPIO to serve as a wake-up source, it must be programmed as an input in the GPDR. When a fault condition is detected in the VDD_FAULT or BATT_FAULT pins, this register is set to hexadecimal 0000 0003, enabling only GP<1,0> as wake-up sources. This register is also set to this value on hard reset (nRESET asserted). For reserved bits, writes are ignored and reads return zero.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	WE31	Reserved			WE27	WE26	WE25	WE24	WE23	WE22	WE21	WE20	WE19	WE18	WE17	WE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

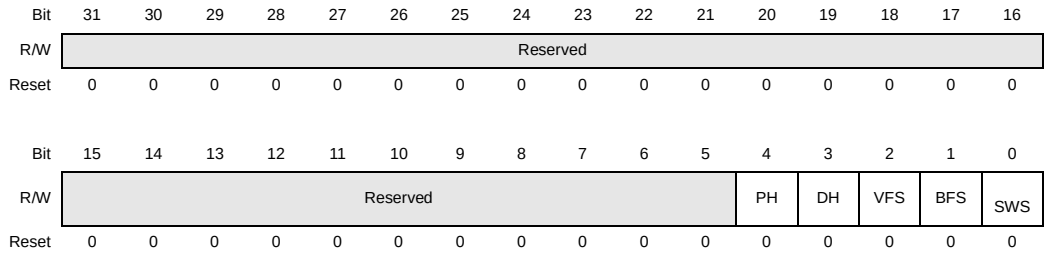
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	WE15	WE14	WE13	WE12	WE11	WE10	WE9	WE8	WE7	WE6	WE5	WE4	WE3	WE2	WE1	WE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bit	Name	Description
{n}	WE{n}	Sleep wake-up enable n (where n = 0 through 27). 0 – Wake-up due to GPIO<n> edge detect disabled. 1 – Wake-up due to GPIO<n> edge detect enabled.
30..28	—	Reserved.
31	WE31	Sleep wake-up enable 31. 0 – Wake-up due to RTC alarm disabled. 1 – Wake-up due to RTC alarm enabled.

9.5.7.5 Power Manager Sleep Status Register (PSSR)

PSSR contains five status flags. The software sleep status flag is set when sleep mode is entered as a result of the force sleep (FS) control bit being set by the CPU. The battery fault status bit is set any time the BATT_FAULT pin is asserted (even when the SA-1100 is already in sleep mode). The VDD fault status bit is set only when the assertion of the VDD_FAULT pin causes sleep mode invocation (that is, if the force sleep bit is asserted and sleep mode is entered followed by the assertion of the VDD_FAULT pin, the VDD fault status bit is not set). Hardware (power-on) reset clears PSSR, but the sleep mode reset, software reset, and watchdog reset do not affect this register. The peripheral hold and DRAM hold bits indicate that those two interfaces retain the same value as during sleep until these bits are cleared.

The five status flags are cleared when a one is written to them. Writing zero to any status bit has no effect. Reserved bits read as zeros and are unaffected by writes. The following table shows the PSSR.



Bit	Name	Description
0	SS	Software sleep status. 0 – Chip has not been placed in sleep mode by setting the force sleep (FS) control bit since it was last cleared by reset or by the CPU. 1– Chip was placed in sleep mode by setting the force sleep (FS) control bit.
1	BFS	Battery fault status. 0 – **BATT_FAULT** pin has not been asserted since it was last cleared by a hardware reset or by the CPU. 1 – **BATT_FAULT** pin has been asserted.
2	VFS	VDD fault status. 0 – **VDD_FAULT** pin has not been asserted since it was last cleared by a hardware reset or by the CPU. 1 – **VDD_FAULT** pin was asserted in run or idle mode and caused the chip to enter sleep mode. **Note**: This bit will not be set by the assertion of **VDD_FAULT** while the SA-1100 is in sleep mode.

Bit	Name	Description
3	DH	DRAM control hold. This bit is set upon exit from sleep mode and indicates that the RAS<3:0> and CAS<3:0> continue to be held low and that the DRAMs are still in self-refresh mode. This bit should be cleared by the processor (by writing a one to it) after the DRAM interface has been configured but before any DRAM access is attempted. The RAS and CAS lines are released when this bit is cleared. This bit is cleared on hardware reset.
4	PH	Peripheral control hold. This bit is set upon exit from sleep mode and indicates that the peripheral pins are being held in their sleep state. This bit should be cleared by the processor (by writing a one to it) after the peripheral interfaces have been configured but before they are actually used by the processor.
31..5	—	Reserved.

9.5.7.6 Power Manager Scratch Pad Register (PSPR)

The power manager also contains a 32-bit register to save processor configuration information in any format the user desires. The power manager scratch pad register (PSPR) is a holding register that is powered by the VDDx power supply pins and is never reset (only configured via writes). Any value can be written to it while in run mode. The value remains intact while in sleep mode, and can be read once sleep mode is exited. The user may use the register value to represent processor configuration prior to sleep mode invocation. (The 32 bits can represent encoded configuration information or can act as a pointer to ROM where a configuration table is kept.) The PSPR is a simple read/write register. See the Section 9.5.8, “Power Manager Register Locations” on page 9-40 for its physical address.

9.5.7.7 Power Manager GPIO Sleep State Register (PSSR)

The sleep state register (PSSR) allows the user to select the output state of each GPIO pin when the SA-1100 goes into sleep mode. When a transition to sleep is required (either through software or through the assertion of the BATT_FAULT or VDD_FAULT pins), the contents of the PSSR is loaded into the GPIO output data register. [This register is normally controlled by software through the GPSR (set) and GPCR (clear) registers]. Only pins already configured as outputs will reflect the new state; however, all 28 bits of the output register are loaded. After the SA-1100 reenters the run mode from sleep, these GPIO pins retain their programmed sleep state until changed by writing ones to the GPSR or GPCR registers; question marks indicate that the values are unknown at reset. If a pin direction is switched from an input to an output, the last contents of the register will be driven onto the pin.

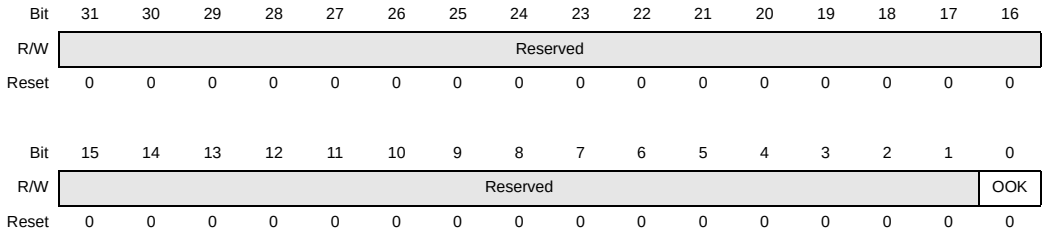
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	Reserved				SS27	SS26	SS25	SS24	SS23	SS22	SS21	SS20	SS19	SS18	SS17	SS16
Reset	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	SS15	SS14	SS13	SS12	SS11	SS10	SS9	SS8	SS7	SS6	SS5	SS4	SS3	SS2	SS1	SS0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	Name	Description
{n}	SS{n}	Sleep state of GPIO n (where n = 0 through 27) 0 – This pin is driven to a zero during the transition to sleep (if programmed as an output). 1 – This pin is driven to a one during the transition to sleep (if programmed as an output).
31..28	—	Reserved

9.5.7.8 Power Manager Oscillator Status Register (POSR)

The power manager oscillator status register (POSR) is a single-bit, read-only register that contains a status bit indicating whether the 32.768-kHz oscillator is up to speed after a hardware reset. This bit is set after the expiration of a timer that is clocked by a ring oscillator. This bit will be set within 2–10 seconds after the negation of nRESET.



Bit	Name	Description
0	OOK	Oscillator OK. This bit is cleared on a hardware reset and set after the 32.768-kHz oscillator has stabilized. This bit is read only.
31..28	—	Reserved.

9.5.8 Power Manager Register Locations

Table 9-4 shows the registers associated with the power manager and the physical addresses used to access them

Table 9-4. Power Manager Register Locations

Address	Name	Description
0h 9002 0000	PMCR	Power manager control register
0h 9002 0004	PSSR	Power manager sleep status register
0h 9002 0008	PSPR	Power manager scratch pad register
0h 9002 000C	PWER	Power manager wake-up enable register
0h 9002 0010	PCFR	Power manager general configuration register
0h 9002 0014	PPCR	Power manager PLL configuration register
0h 9002 0018	PGSR	Power manager GPIO sleep state register
0h 9002 001C	POSR	Power manager oscillator status register

9.6 Reset Controller

The reset controller manages the various reset sources within the SA-1100. From a programmer's view, it is visible as two registers: one used to invoke software reset and one to read status after booting to indicate why the processor was reset.

The four types of reset in the SA-1100 include:

- **Hardware reset**

  Hardware reset is invoked when the nRESET pin is asserted and resets all units in the SA-1100 to a known state. Hardware reset is intended to be used for power-up only. Because the memory controller receives a full reset, all DRAM contents will be lost during hardware reset. The RESET_OUT pin is asserted during hardware reset.

- **Software reset**

  Software reset is invoked when the software reset (SWR) bit in the RSRR is set by software. Software reset applies reset to the majority of the SA-1100 as well as causing the assertion of the RESET_OUT pin. During software reset, the DRAM refresh and configuration are not cleared. This allows DRAM contents to survive a software reset. After the SWR bit is set, the SA-1100 stays reset for 256 processor clocks and then is allowed to boot again.

- **Watchdog reset**

  Watchdog reset is invoked when the watchdog enable bit (WE) in the OWER is set and the OSMR3 matches the OS timer counter. When watchdog reset is invoked, the rest of the reset sequence is identical to software reset. The watchdog enable bit cannot be cleared under program control. Only one of the four reset types can clear it.

- **Sleep reset**

  Sleep reset is invoked automatically when the SA-1100 enters sleep mode. During sleep mode, the majority of the processor loses power and will receive reset prior to the negation of the PWR_EN pin. Sleep reset does not affect the power manager, RTC, or GPIO wake-up register. During sleep reset, although the memory controller is in reset, the RAS<3:0> and CAS<3:0> pins are held in the self-refresh state required by the DRAMs.

After booting from a reset, software can examine the reset controller reset status register (RCSR) to determine which types of reset caused the reset condition.

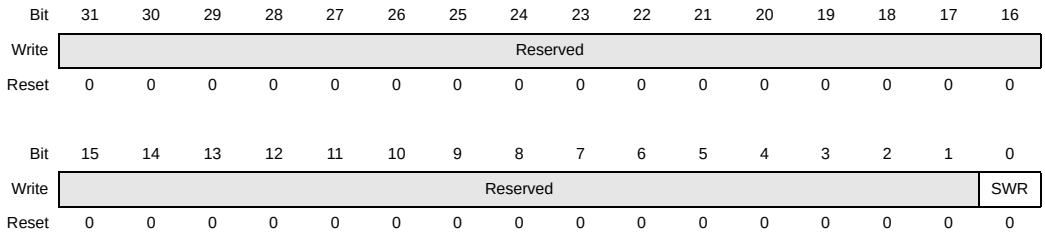
9.6.1 Reset Controller Registers

The reset controller contains two registers, the reset controller software reset register (RSRR) and the reset controller reset status register (RCSR).

9.6.1.1 Reset Controller Software Reset Register (RSRR)

The reset controller software reset register has a software reset bit, which when set, causes a reset of the SA-1100. The software reset bit (SWR) is located within the least significant bit of the write-only reset controller software reset register (RSRR). Writing a one to this bit causes all on-chip resources to reset but does not cause the PLL to go out of lock. The software reset bit is self-resetting. It is automatically cleared to zero several system clock cycles after a one is written to it. Writing zero to the software reset bit has no effect. Care should be taken to restrict access to this register by programming MMU permissions. For reserved bits, writes have no effect. Reading this register returns zeros.

The following table shows the RSRR.



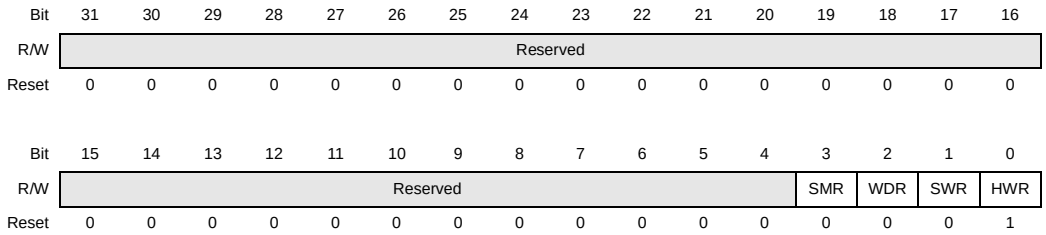
Bit	Name	Description
0	SWR	Software reset. 0 – Do not invoke a software reset of the chip. 1 – Invoke a software reset of the chip. **Note**: This bit is self-resetting, and is automatically cleared several system clock cycles after it has been set.
31..1	—	Reserved.

9.6.1.2 Reset Controller Status Register (RCSR)

The reset controller reset status register (RCSR) is used by the CPU to determine the last cause or causes of the reset. The SA-1100 has four sources of reset:

- Hardware reset
- Software reset
- Watchdog reset
- Sleep mode reset

Each RCSR status bit is set by a different source of reset, and can be cleared by writing a one back to that bit. Note that the hardware reset state of software, watchdog, and sleep mode reset bits is zero. The table below shows the status bits within RCSR. For reserved bits, writes are ignored and reads return zero.



Bit	Name	Description
0	HWR	Hardware reset. 0 – Hardware reset has not occurred since the last time the CPU cleared this bit. 1 – Hardware reset has occurred since the last time the CPU cleared this bit.
1	SWR	Software reset. 0 – Software reset has not occurred since the last time the CPU cleared this bit. 1 – Software reset has occurred since the last time the CPU cleared this bit.
2	WDR	Watchdog reset. 0 – Watchdog reset has not occurred since the last time the CPU cleared this bit. 1 – Watchdog reset has occurred since the last time the CPU cleared this bit.
3	SMR	Sleep mode reset. 0 – Sleep mode reset has not occurred since the last time the CPU cleared this bit. 1 – Sleep mode reset has occurred since the last time the CPU cleared this bit. **Note**: Each status flag can be cleared only by reading a one and then writing a zero to it.
31..4	—	Reserved.

9.6.2 Reset Controller Register Locations

Table 9-5 shows the registers associated with the reset controller and the physical addresses used to access them.

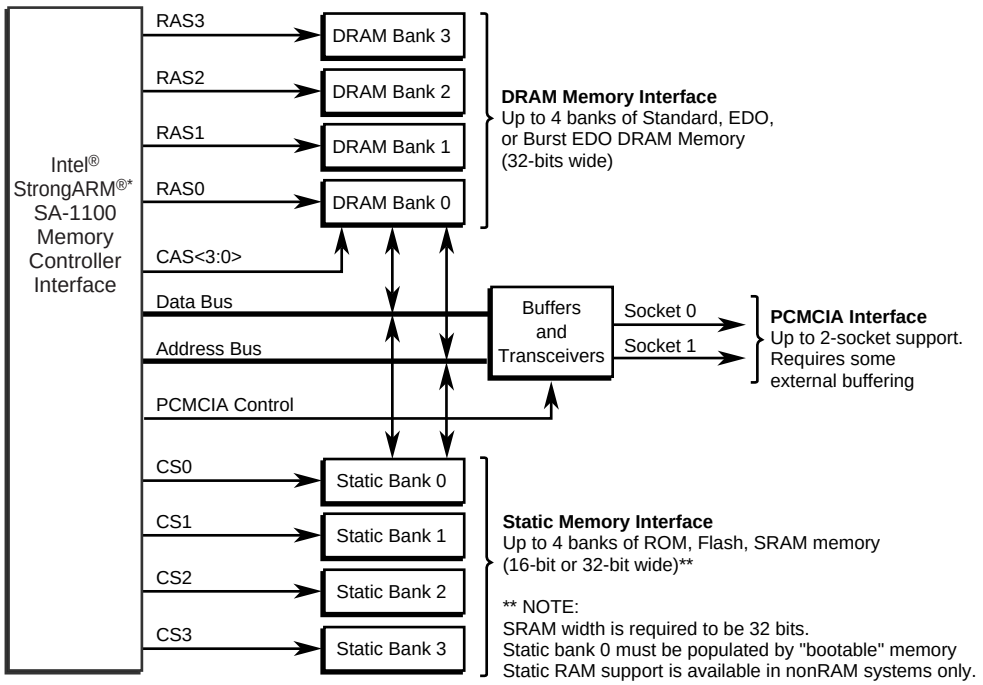
Table 9-5. Reset Controller Register Locations

Address	Name	Description
0h 9003 0000	RSRR	Reset controller software reset register
0h 9003 0004	RCSR	Reset controller status register

Memory and PCMCIA Control Module 10

The external memory bus interface for the Intel® StrongARM® SA-1100 Microprocessor (SA-1100) supports standard fast-page and EDO asynchronous DRAMs, burst and nonburst ROMs, Flash EPROMs, SRAM, and PCMCIA expansion devices. It is programmable through the memory interface configuration registers. Figure 10-1 shows a block diagram of the maximum configuration of the memory controller.

Figure 10-1. General Memory Interface Configuration



* StrongARM is a registered trademark of ARM Limited..

A6841-01

10.1 Overview of Operation

The SA-1100 memory interface supports three interfaces:

- **DRAM Memory Interface**

The dynamic memory interface supports four 32-bit wide banks of fast-page or EDO asynchronous DRAMs. Each bank is allocated 128 Mbyte of the internal memory map. However, the actual size of each bank is dependent on the particular DRAM configuration used. If multiple banks are populated, each must be identical in size and configuration. There are 4 bank selects, nRAS<3:0>,

4 byte selects, nCAS<3:0>, 12 bits of multiplexed row and column addresses, nWE, and nOE. The SA-1100 performs CAS before RAS refresh (CBR) during normal operation and supports self-refreshing DRAMs during power-down sleep mode.

- **Static Memory Interface**

The static memory interface has four chip selects, nCS<3:0>, and 26 bits of byte address, A<25:0>, for access of up to 64 Mbyte of memory in each of four banks. Each chip select is individually programmable for selecting nonburst ROM, burst ROM, Flash EPROM, or asynchronous SRAM. Each may be individually configured to be 16 or 32 bits wide except SRAM, which, if used, must be 32 bits. nOE is asserted on reads and nWE is asserted on writes. For SRAMs, nCAS<3:0> are byte selects for both reads and writes. Because the nCAS<3:0> pins are used to control both SRAM and DRAM, systems with both memory types are not supported.

When the SA-1100 comes out of reset, it begins fetching and executing instructions at address 0x00, which corresponds to memory selected by nCS0. This is where boot ROM is expected to be.

- **PCMCIA Interface**

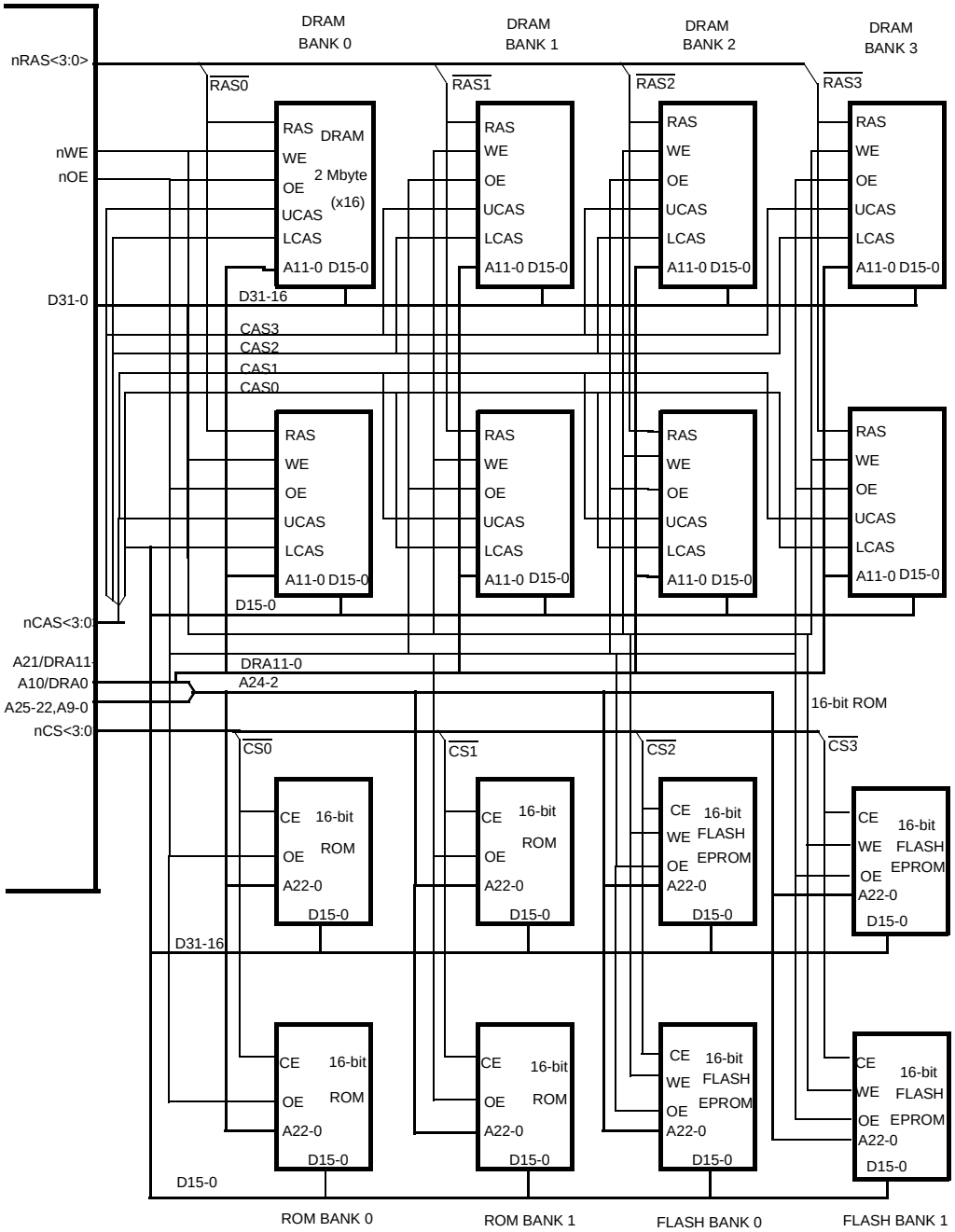
The PCMCIA interface provides control signals to support a single PCMCIA card slot with additional hooks to support two slots. It shares address and data pins with the memory devices. It uses address lines, A<25:0>, and data lines, D<15:0>. nPREG is actually A<26> and selects register space (I/O or attribute) versus memory space. nPOE and nPWE are provided for memory and attribute reads and writes. nPIOR, nPIOW, and nIOIS16 control I/O reads and writes. nPWAIT allows for extended access times. nPCE1 and nPCE2 are byte select low and high, respectively. PSKTSEL selects between two card slots.

This interface also supports 32-bit accesses that are outside the PCMCIA specification. There are several restrictions with respect to the use of this feature that are described later in this chapter.

10.1.1 Example Memory System

Figure 10-2 shows a system using 1M x 16 DRAMs for a total of 16 Mbyte of DRAM. Two banks of ROM and two banks of Flash EPROM are shown, each on a 32-bitwide databus. The PCMCIA interface is not shown.

Figure 10-2. Example Memory Configuration



10.1.2 Types of Memory Accesses

The SA-1100 performs memory accesses for the following operations:

- Unbuffered write
- Uncached read
- Buffered write
- Linefetch
- Read buffer fetch
- Internal DMA write
- Level 1 translation fetch
- Level 2 translation fetch
- Cache line copyback
- Read-lock-write sequence
- Internal DMA read

SA-1100 will only generate a subset of all possible transactions on the bus. Many of these transactions may be completed internal to the processor by accessing caches, the read buffer, on-chip registers, or the memory space that returns zeroes for flushing the cache.

If a memory access is followed by an idle period on the bus, the control signals will return to their inactive state and the address and data signals will remain at their previous values to avoid unnecessary bus transitions and eliminating the need for many pull-up resistors.

10.1.3 Reads

Read bursts are generated by DMA requests, read buffer requests, and cache line fills. All cache line fills are 8 words long. DMA and read buffer requests may be 1, 4, or 8 words long. All other reads are single accesses.

Data and instruction cache line fills start on an 8-word boundary and will be 8 words long.

10.1.4 Writes

For single access writes, one byte, half-word, or word is written. The write burst sizes are 1, 2, 3, or 4 full words. A write burst size of 8 words may be generated by castouts and all 32 bytes are written.

For stores to DRAM or SRAM memory spaces, the nCAS<3:0> lines enable a corresponding byte of the data bus during a write transaction. Flash memory space stores must be the width of the Flash data bus, either 16 or 32 bits.

10.1.5 Transaction Summary

Table 10-1 lists all the transactions that the SA-1100 can generate. No burst will cross an aligned 32-byte boundary. Note that on a 16-bit bus, the read single operation becomes a two half-word burst with address bit 1 always starting at 0. Writes to Flash memory space will take place in one single operation regardless of bus size.

Table 10-1. SA-1100 Transactions

Bus Operation	Burst Size	Starting Address Bits <4:2>	Description
Read single	1	Any	Generated by core, DMA, or read buffer request.
Read burst	4	0 4	Generated by read buffer or DMA request.
Read burst	8	0	Generated by cacheline fills or read buffer request.
Write single	1	Any	1..4 bytes are written as specified by the byte mask. Generated by write buffer or DMA request.
Write burst	2	0, 1, 2 4, 5, 6	All 4 bytes of each word are written. Generated by write buffer or DMA request.
Write burst	3	0, 1 4, 5	All 4 bytes of each word are written. Generated by write buffer or DMA request.
Write burst	4	0 4	All 4 bytes of each word are written. Generated by write buffer or DMA request.
Write burst	8	0	Cacheline copyback. All 32 bytes are written. Generated by write buffer.

10.1.6 Read-Lock-Write

The read-lock-write sequence is generated by an SWP instruction to a noncacheable/nonbufferable location. Locked access to memory is ensured through internal arbitration of accesses to the memory controller.

10.1.7 Aborts and Nonexistent Memory

Reads from reserved address locations (as specified in the memory map) will result in a data abort exception. Writes to reserved address space will have no effect.

Reads and writes from or to nonexistent memory are not detected in hardware. In case no memory is selected on a read, the value last driven on the data bus is returned.

A single access to a disabled DRAM bank (MDCNFG:DEx=0) will cause a CBR refresh cycle to all banks. Zeros are returned to the register file on reads and writes are dropped. A burst read access to a disabled DRAM bank will result in a data abort exception.

10.2 Memory Configuration Registers

The SA-1100 memory interface is programmed through a set of configuration registers that are described in the following sections.

Table 10-2 shows the registers associated with the memory interface and the physical addresses used to access them. All addressing is little endian. These registers are readable and writable only as full words. They are grouped together within one page and thus all have the same memory protections.

Table 10-2. Memory Interface Control Registers

Physical Address	Symbol	Register Name
0xA000 0000	MDCNFG	DRAM configuration register
0xA000 0004	MDCAS0	DRAM CAS waveform shift register 0
0xA000 0008	MDCAS1	DRAM CAS waveform shift register 1
0xA000 000C	MDCAS2	DRAM CAS waveform shift register 2
0xA000 0010	MSC0	Static memory control register 0
0xA000 0014	MSC1	Static memory control register 1
0xA000 0018	MECR	Expansion bus configuration register

10.2.1 DRAM Configuration Register (MDCNFG)

MDCNFG is a read/write register and contains control bits for configuring the DRAM. All DRAM banks must be implemented with the same type of DRAM devices. Question marks indicate that the values are unknown at reset.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read	DRI14	DRI13	DRI12	DRI11	DRI10	DRI9	DRI8	DRI7	DRI6	DRI5	DRI4	DRI3	DRI2	DRI1	DRI0	TDL1
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
-	?															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	TDL0	TRASR3	TRASR2	TRASR1	TRASR0	TRP3	TRP2	TRP1	TRP0	CDB2	DRAC1	DRAC0	DE3	DE2	DE1	DE0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0

Bit	Name	Description
3..0	DE<3:0>	DRAM enable bank 3-0. For each DRAM bank, there is an enable bit. Reads or writes to a disabled DRAM bank trigger a single CBR refresh cycle to all banks. When all banks are disabled, the refresh counter is disabled. 0 – DRAM bank disabled. 1 – DRAM bank enabled. These bits are cleared by hardware reset.
5..4	DRAC<1:0>	DRAM row address bit count. 00 – 9 row address bits. (Select this for support of 9x9 and 9x8 DRAMs.) 01 – 10 row address bits. (Select this for support of 10x10, 10x9, and 10x8 DRAMs.) 10 – 11 row address bits. (Select this for support of 11x11, 11x10, 11x9, and 11x8 DRAMs.) 11 – 12 row address bits. (Select this for support of 12x10, 12x9, and 12x8 DRAMs.)
6	CDB2	Clock divide by 2. 0 – CAS waveform shift register (MDCAS0, 1, 2) shifted every CPU clock. 1 – CAS waveform shift register shifted every memory clock. (CPU clock divided by 2.)
10..7	TRP<3:0>	RAS precharge. Number of memory clocks nRAS deasserted before next assertion. Between any two DRAM accesses, nRAS is high for TRP+1 or 2 memory cycles (whichever is greater). Between a DRAM access and a refresh, both nRAS and nCAS are deasserted for TRP+1 or 2 memory cycles (whichever is greater).
14..11	TRASR<3:0>	RAS assertion during CBR. Number of memory clocks (minus one) nRAS asserted during CAS before RAS refresh.
16..15	TDL<1:0>	Data input latch after CAS deassertion. 00 – Read data is latched coincident with the deassertion of nCAS. 01 – Read data is latched one CPU clock cycle after the deassertion of nCAS (useful for EDOs). 10 – 2 clocks later. 11 – 3 clocks later.

Bit	Name	Description
31..17	DRI<14:0>	DRAM refresh interval. The number of memory clock cycles (divided by 4) between CAS before RAS (CBR) refresh cycles. One row is refreshed in each DRAM bank during each CBR refresh cycle. The value that must be loaded into this register is calculated as follows: DRI = Number of cycles/4 = ((Refresh time / rows) - (longest burst access time)) x Mem clock frequency /4. The longest burst access time to subtract must also take into consideration access to ROM or Flash EPROM. (These may be interrupted to service a DRAM refresh cycle after each 32-bit word. If there is a read on a 16-bit bus, a refresh cycle may be inserted after 2 read cycles. If there is a read to a 32-bit bus, the refresh waits one read cycle to be serviced. The DRAM interface inserts CBR refresh cycles between bursts of up to 8 words. Because the address pins are ignored by the DRAMs during CBR refresh cycles, PCMCIA transactions may be ongoing during a refresh cycle and will not be interrupted.)

10.2.2 DRAM CAS Waveform Shift Registers (MDCAS0, MDCAS1, MDCAS2)

MDCAS0, MDCAS1, and MDCAS2 are 32-bit read/write registers that contain the nCAS waveform for a full 8-beat burst read or write to asynchronous DRAM. Each bit represents one CPU cycle if MDCNFG:CDB2 is 0 and 2 CPU cycles (1 memory clock cycle) if MDCNFG:CDB2 is 1. The least significant bit of MDCAS0 goes out first and is the cycle coincident with the assertion of nRAS. Bit 1 is one cycle after the assertion of nRAS, and so on. MDCAS1 is appended after MDCAS0 and MDCAS2 is appended after MDCAS1. A 1 in any field causes nCAS to be deasserted in that cycle; a 0 causes nCAS to be asserted in that cycle. The memory controller counts nCAS pulses and deasserts nRAS in the cycle following the deassertion of the final nCAS pulse of the burst. All eight nCAS pulses must be programmed or the processor will hang. When MDCNFG:CDB2 is 0, the MDCAS0 must contain 1s in the lower 4 bits and each transition of nCAS must be a minimum of 2 clocks (nCAS must be asserted for a minimum of 2 CPU clock cycles and deasserted for 2). When MDCNFG:CDB2 is 1, the MDCAS0 must contain 1s in the lower 2 bits and each transition of nCAS must be a minimum of 1 bit. These registers are unaffected by reset.

10.2.3 Static Memory Control Registers (MSC1–0)

MSC1 and MSC0 are read/write registers and contain control bits for configuring static memory selected by nCS<3:0>. Reset forces the values in these registers to the slowest possible nonburst ROM timing. Timing fields are specified as numbers of memory clock cycles. The memory clock cycle consists of two CPU cycles. Each register contains two identical fields, for a total of four identical fields, each corresponding to the chip select, nCS<x>, of the same number. On hardware reset, the MSC0:SMCNFG0 field is set to 0b 1111 1111 1111 1x00 (binary) where x represents the inverse of the ROM_SEL pin. All other fields in MSC0 and MSC1 are unaffected by reset; question marks indicate that the values are unknown at reset.

MSC0 Register Format

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read	RRR1_2	RRR1_1	RRR1_0	RDN1_4	RDN1_3	RDN1_2	RDN1_1	RDN1_0	RDF1_4	RDF1_3	RDF1_2	RDF1_1	RDF1_0	RBW1	RT1_1	RT1_0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
-	?															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RRR0_2	RRR0_1	RRR0_0	RDN0_4	RDN0_3	RDN0_2	RDN0_1	RDN0_0	RDF0_4	RDF0_3	RDF0_2	RDF0_1	RDF0_0	RBW0	RT0_1	RT0_0
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	x	0	0

MSC1 Register Format

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read	RRR3_2	RRR3_1	RRR3_0	RDN3_4	RDN3_3	RDN3_2	RDN3_1	RDN3_0	RDF3_4	RDF3_3	RDF3_2	RDF3_1	RDF3_0	RBW3	RT3_1	RT3_0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
-	?															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RRR2_2	RRR2_1	RRR2_0	RDN2_4	RDN2_3	RDN2_2	RDN2_1	RDN2_0	RDF2_4	RDF2_3	RDF2_2	RDF2_1	RDF2_0	RBW2	RT2_1	RT2_0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	Name	Description
1..0	RTx<1:0>	ROM type. 00 – Nonburst ROM or Flash EPROM. 01 – Nonburst ROM or SRAM. [1] 10 – Burst-of-four ROM. 11 – Burst-of-eight ROM.
2	RBWx	ROM bus width. 0 – 32 bits 1 – 16 bits On reset, the RBW field in SMCNFG0 is loaded with the inverse of the ROM_SEL pin.
7..3	RDFx<4:0>	ROM delay first access. Number of memory clock cycles (minus 1) from address to data valid for a nonburst ROM or the first access of a burst ROM. For Flash and SRAM, this determines the read access time. One memory clock cycle is added to this value.

Bit	Name	Description
12..8	RDNx<4:0>	ROM delay next access. Number of memory clock cycles (minus 1) from address to data valid for subsequent accesses of a burst ROM. For Flash and SRAM, this determines the write pulse width. One memory clock cycle is added to this value.
15..13	RRRx<2:0>	ROM/SRAM recovery time. Number of memory clock cycles (divided by 2) from chip select deasserted after a read to next chip select (of a different memory bank) or nRAS asserted. nCS negated to nRAS asserted is 2*RRR or 1 cycle (whichever is greater). For Flash and SRAM, this field will also be used after writes to hold off subsequent accesses. This field should be programmed with the maximum of Toff, write pulse high time (Flash/SRAM), and write recovery before read (Flash).

[1] When SMCNFGx:RT=01, accesses to the selected bank will output a byte mask on **nCAS<3:0>** for both reads and writes. This option should be selected only when there is no DRAM in the system.

10.2.4 Expansion Memory (PCMCIA) Configuration Register (MECR)

MECR is a read/write register that contains control bits for configuring the timing of the PCMCIA interface. This register is unaffected by reset; question marks indicate that the values are unknown at reset.

Writes to the reserved fields have no effect and reads return zeros. The programming of each of the six fields allows the user to individually select the duration of accesses to I/O, common memory, and attribute memory for each of two PCMCIA card slots. Each field is identical and represents the number of memory clocks per tick of an internal clock, referred to as BCLK. BCLK clocks the internal PCMCIA state machine. See Figure 10-15 for a description of the PCMCIA timing diagram.

The BCLK_SEL field is designed to allow the user to program the speeds of the PCMCIA memory, attribute, and I/O accesses. When an access to a PCMCIA address space is detected, the appropriate BS_xx field is selected based on the memory map. Every (BS_xx + 1) memory clock cycles, a BCLK tick is generated to advance the PCMCIA state machine. All signals (except nPWAIT, which is asynchronous) on the PCMCIA bus are driven or sampled relative to this internal clock, although the clock itself is not driven. Table 10-3 shows the number of processor clocks per BCLK tick for each BS_xx value. Table 10-4 shows the internal BCLK cycle times for each BS_xx setting given a processor core frequency of 160 MHz (6.25-ns cycle time).

Note: The BCLK speed for a given setting will change if the processor frequency changes.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read	RES	BSM2_4	BSM2_3	BSM2_2	BSM2_1	BSN2_0	BSA2_4	BSA2_3	BSA2_2	BSA2_1	BSA2_0	BSIO2_4	BSIO2_3	BSIO2_2	BSIO2_1	BSIO2_0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
-	?															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	RES	BSM1_4	BSM1_3	BSM1_2	BSM1_1	BSN1_0	BSA1_4	BSA1_3	BSA1_2	BSA1_1	BSA1_0	BSIO1_4	BSIO1_3	BSIO1_2	BSIO1_1	BSIO1_0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	Name	Description
4..0	BSIO1<4:0>	Clock count for accesses to PCMCIA card slot 1, I/O space.
9..5	BSA1<4:0>	Clock count for accesses to PCMCIA card slot 1, attribute space.
14..10	BSM1<4:0>	Clock count for accesses to PCMCIA card slot 1, common memory space.
15	—	Reserved.
20..16	BSIO2<4:0>	Clock count for accesses to PCMCIA card slot 2, I/O space.
25..21	BSA2<4:0>	Clock count for accesses to PCMCIA card slot 2, attribute space.
30..26	BSM2<4:0>	Clock count for accesses to PCMCIA card slot 2, common memory space.
31	—	Reserved.

Table 10-3. BS_xx Bit Encoding

Bit	Name	Description
4..0	BS_xx	0b00000 – BCLK= 2 processor clocks (clk/2) 0b00001 – BCLK= 4 processor clocks 0b00010 – BCLK= 6 processor clocks .... 0b11101 – BCLK= 60 processor clocks 0b11110 – BCLK= 62 processor clocks 0b11111 – BCLK= 64 processor clocks

Table 10-4. BCLK Speeds for 160-MHz Processor Core Frequency

BCLK_SEL	BCLK Cycle Time–ns
0b00000 – Every 2 processor clocks (clk/2).	12.5
0b00001 – Every 4 processor clocks.	25
0b00010 – Every 6 processor clocks.	37.5
0b00011 – Every 8 processor clocks.	50
...	
0b11111 – Every 64 processor clocks.	400

To calculate the recommended BS_xx value for each address space: divide the command width time (the greater of twIOWR and twIORD, or the greater of twWE and twOE) by processor cycle time; divide by 2; divide again by 3 (number of BCLKs per command assertion); round up to the next whole number; and subtract 1. For example, for a processor cycle time of 6.25 ns and an nIOWR command assertion time of 165 ns, the recommended setting for BS_IO would be (165 /(2 x 3 x 6.25)) - 1 = 3.4, or 4 after rounding up.

10.3 Dynamic Interface Operation

This section describes the dynamic memory interface.

10.3.1 DRAM Overview

The dynamic memory interface supports up to four banks of identical size and type dynamic memory on a 32-bit bus. Initialization software must set up the memory interface configuration registers with the DRAM size, type, number of row address bits, nCAS waveforms, and timing parameters. The SA-1100 generates accesses of 1–8 words.

Table 10-5 shows some of the supported DRAM configurations.

Table 10-5. DRAM Memory Size Options

Bank Size (Mbyte/Bank)	DRAM Configuration (Words x Bits)	Chip Size	Number Chips / Bank	Row bits x Col. Bits	Max Memory (4 Banks, 32-bit Bus)	Total Number of Chips
1 Mbyte	256 K x 16	4 Mbit	2	9 x 9	4 Mbyte	8
2 Mbyte	512 K x 8	4 Mbit	4	10 x 9	8 Mbyte	16
2 Mbyte	512 K x 32	16Mbit	1	10 x 9	8 Mbyte	4
4 Mbyte	1 M x 4	4 Mbit	8	10 x 10	16 Mbyte	32
4 Mbyte	1 M x 16	16 Mbit	2	10 x 10, 12 x 8	16 Mbyte	8
8 Mbyte	2 M x 8	16 Mbit	4	11 x 10, 12 x 9	32 Mbyte	16
16 Mbyte	4 M x 16	64 Mbit	2	12 x 10	64 Mbyte	8

Table 10-6 shows the DRAM row and column address multiplexing. For each row address size specified, column address sizes of 11, 10, 9, and 8 are supported wherever the row address is larger than or the same size as the column address (12 rows x 11 columns are not supported). Connecting address lines to the DRAM chips as shown allows the proper addressing without having to specify the column address size.

Table 10-6. DRAM Row/Column Address Multiplexing

Number of Row Address Bits (as specified in MDCNFG:DRAC)		DRAM Address Pins at RAS Time				DRAM Address Pins at CAS Time				
		DRA11	DRA10	DRA9	DRA8-0	DRA11	DRA10	DRA9	DRA8	DRA7-0
DRAM:	12 bits	IA21	IA20	IA19	IA18-10	x	x	IA23	IA22	IA9-2
	11 bits	x	IA20	IA19	IA18-10	x	IA23	IA22	IA21	IA9-2
	10 bits	x	x	IA19	IA18-10	x	x	IA21	IA20	IA9-2
	9 bits	x	x	x	IA18-10	x	x	x	IA19	IA9-2

DRAx = SA-1100 DRAM interface address pin, A(21:10) = DRA(11:0)

IAx = Internal address bit

Note: At RAS time, all address pins, A(25:0), are driven with the internal address that corresponds to the pin of the same number. For example, a DRAM with 13 bits of row address can be accommodated by hooking up the 13th row address line of the DRAM to SA-1100 address pin A22. (MDCNFG:DRAC is a "don't care".) The column address, in this case, will be limited to a maximum of 8 bits. In general, DRAMs that utilize fewer than 8 column address bits can be used, but there will be holes in the memory map due to no physical memory being addressed by the still significant internal address bit IA9.

10.3.2 DRAM Timing

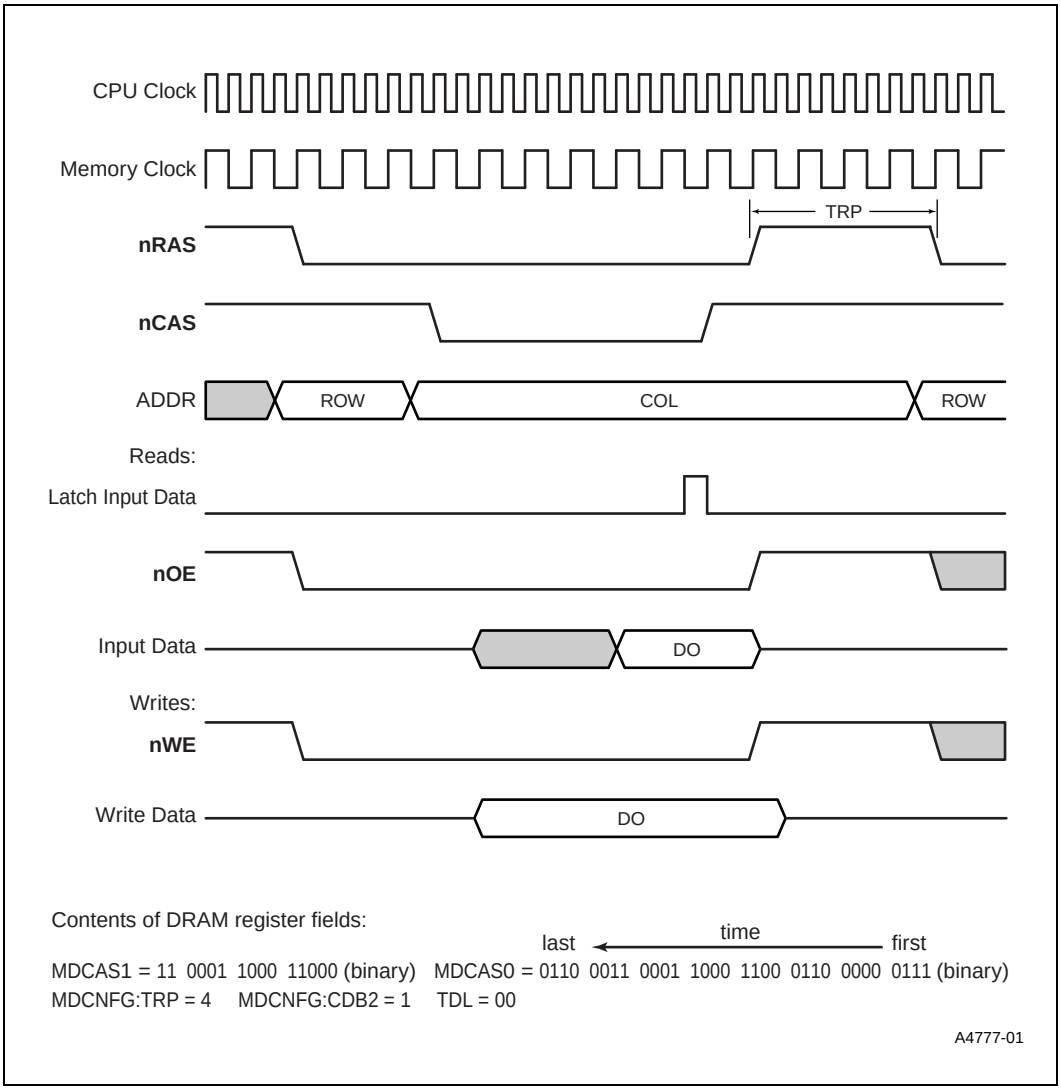
The DRAM nCAS timing is generated using shift registers. The rate at which these shift registers are clocked is determined by MDCNFG:CDB2. The time at which to sample the read data is programmable to coincide with the deassertion of nCAS or up to 3 CPU cycles later. This method provides a way to take advantage of the EDO DRAMs while still supporting the fast-page-mode DRAMs. A full 8-beat burst nCAS waveform is specified, and the memory interface controller shifts the waveform shift register once every CPU clock cycle if MDCNFG:CDB2=0 and once every 2 CPU clock cycles if MDCNFG:CDB2=1. The shifting continues until the number of nCAS pulses have been generated that corresponds to the actual number of data words being accessed.

Registers MDCAS0, MDCAS1, and MDCAS2 contain the nCAS waveform for a full 8-beat burst access to DRAM. To begin an access, the row address is output on DRA(11:0), which is A(21:10). One CPU clock later (1/2 memory clock), nRAS is asserted and the nCAS waveform begins and is shifted with each CPU clock, if MDCNFG:CDB2=0. A 1 in this shift register drives nCAS high (deasserts) at the rising edge of the CPU clock cycle, and a 0 drives nCAS low (asserts). The column address for the first beat of data will be valid 1 CPU cycle before nCAS transitions from deasserted to asserted. During reads, a rising edge is detected on the nCAS waveform and input data is latched MDCNFG:TDL cycles after the rising edge. The shift register continues to shift until the number of nCAS pulses equals the burst size of the current transaction. For write transactions, nRAS will be deasserted on the next rising memory clock edge after the last nCAS rising edge (either 1 or 2 CPU clock cycles). For read transactions, nRAS will be deasserted on the rising memory clock cycle edge that occurs either 2 or 3 CPU clock cycles after the input data is latched. For each additional beat after the first, the column address will be updated coincident with the deassertion of nCAS, or 1 CPU cycle later. For writes, the write data outputs will follow the same timing as the column address. nWE and nOE, as appropriate, follow the same timing as nRAS. After nRAS is deasserted, the timing parameter MDCNFG:TRP is used to determine the wait before the next assertion of nRAS.

If MDCNFG:CDB2=1, the nCAS waveform will be shifted every memory clock, or every 2 CPU cycles. The timing of the other signals remains the same relative to the nCAS waveform. For MDCNFG:CDB2=0, there is a requirement that nCAS high and low times be programmed with a minimum of 2 bits and the 4 least significant bits in MDCAS0 must be 1. For the MDCNFG:CDB2=1 case, high and low nCAS pulse times may be 1 bit each and the least significant 2 bits of MDCAS0 must be 1. These requirements are necessary for the internal hardware to properly generate addresses and write data, and for proper address and data setup times.

Figure 10-3 shows the rate of the shift registers during DRAM nCAS timing for a single-beat transaction.

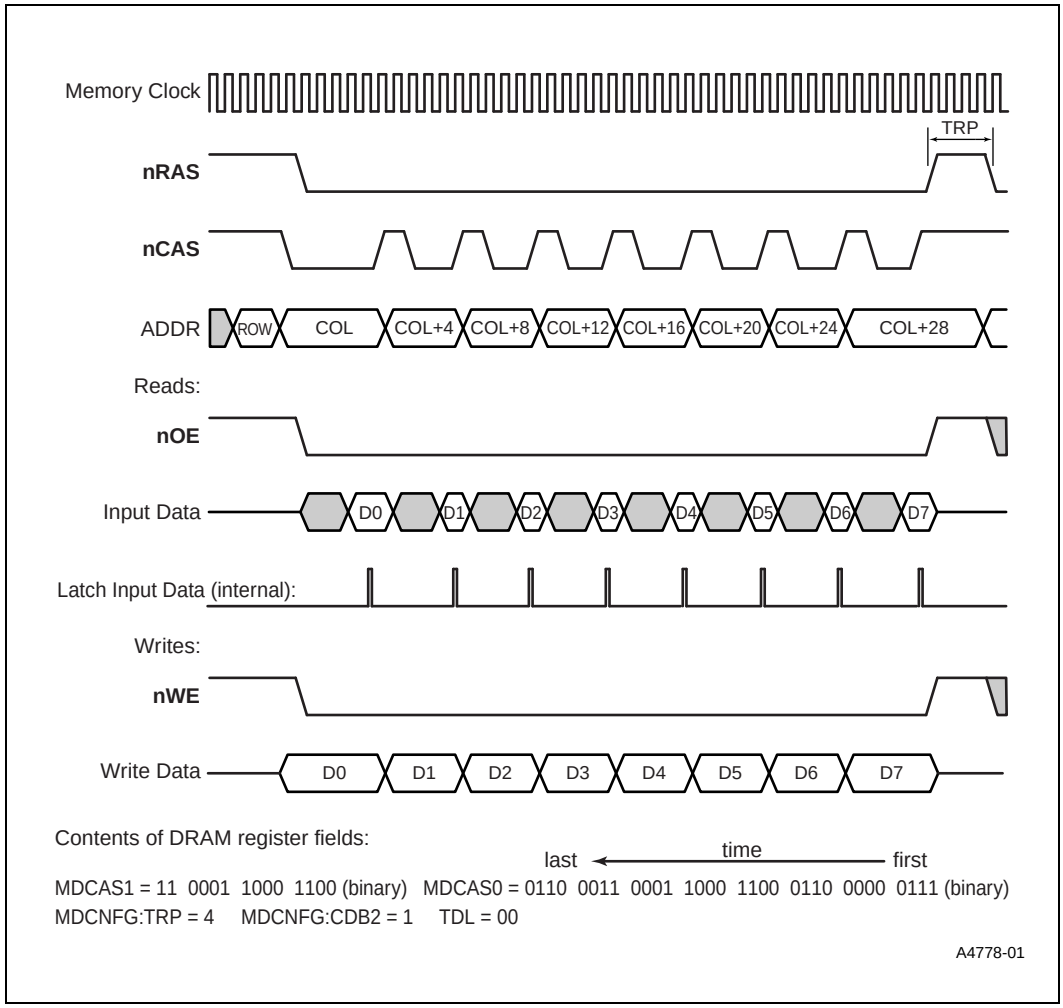
Figure 10-3. DRAM Single-Beat Transactions



Contents of DRAM register fields: last ◄ time ► first
MDCAS1=11 0001 1000 1100(binary) MDCAS0= 0110 0011 0001 1000 1100 0110 0000 0111(binary)
MDCNFG:TRP=4 MDCNFG:CDB2=1 TDL=00

Figure 10-4 shows the rate of the shift registers during DRAM nCAS timing for burst-of-eight transactions.

Figure 10-4. DRAM Burst-of-Eight Transactions



Contents of DRAM register fields:

last ← time → first

MDCAS1=11 0001 1000 1100(binary) MDCAS0= 0110 0011 0001 1000 1100 0110 0000 0111(binary)

MDCNFG:TRP=4 MDCNFG:CDB2=1 TDL=00

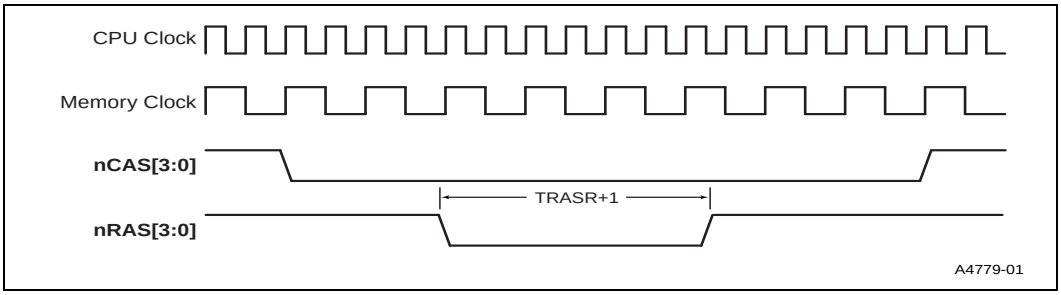
10.3.3 DRAM Refresh

The SA-1100 provides support for CAS before RAS (CBR) refresh. When the DRAM interface is enabled [by setting any of MDCNFG:DE(3-0) and setting MDCNFG:DRI greater than zero], the refresh counter starts counting up every memory cycle (2 CPU cycles) from 0. When its value reaches the value in MDCNFG:DRI times 4, the memory controller is notified that a refresh cycle is due, then the counter is cleared and resumes counting. After the current transaction completes, a refresh cycle is performed. All four nCAS lines are asserted. Two memory clock cycles later (4 CPU cycles), the nRAS signals for all enabled banks are asserted and held low for MDCNFG:TRASR+1 memory clock cycles. After that, all nRAS and nCAS signals are deasserted and MDCNFG:TRP is used to hold off subsequent DRAM accesses to allow for row precharge time. Hardware reset clears the refresh counter. Software reset does not affect it.

A read or write to any disabled DRAM bank will cause a refresh cycle to all banks to occur.

Figure 10-5 shows a timing diagram of a CBR refresh cycle.

Figure 10-5. DRAM Refresh Cycle



10.3.4 DRAM Self-Refresh in Sleep Mode

The SA-1100 will put the DRAM into the self-refresh state prior to entering sleep mode by asserting nCAS, then asserting nRAS (just as for a normal CBR refresh cycle), and maintaining nCAS and nRAS low while power and clocks are turned off.

See Section 9.5, “Power Manager” on page 9-26 for details on how to bring the DRAMs out of self-refresh mode. An access to a DRAM bank while the DRAM interface is in self-refresh mode will have undefined results, but the DRAMs will remain in self-refresh mode.

10.4 Static Memory Interface

The static memory interface is comprised of four chip selects, nCS<3:0>, and are each configurable for ROM, burst ROM, SRAM, or Flash EPROM. The data bus for each chip select region may be programmed to be 16 or 32 bits wide, although if SRAM is selected, only a 32-bit bus is supported. nOE is asserted for all reads. nWE is asserted for Flash and SRAM writes. For SRAM implementations, nCAS<3:0> signals are used for the byte enables where nCAS<3> corresponds to the MSB. The SA-1100 supplies 26 bits of byte address (A<25:0>) for access of up to 128 Mbyte per chip select. A<0> is not used in 16-bitwide bus systems and <1:0> are not used in 32-bitwide systems.

The RT fields in the MSCx registers specify the type of memory (burst-of-four ROM, burst-of-eight ROM, nonburst ROM, Flash, SRAM) and the RBW fields specify the bus width for the memory space selected by nCS<3:0>. If a 16-bit bus width is specified, transactions take place across data pins D<15:0>.

10.4.1 ROM Interface Overview

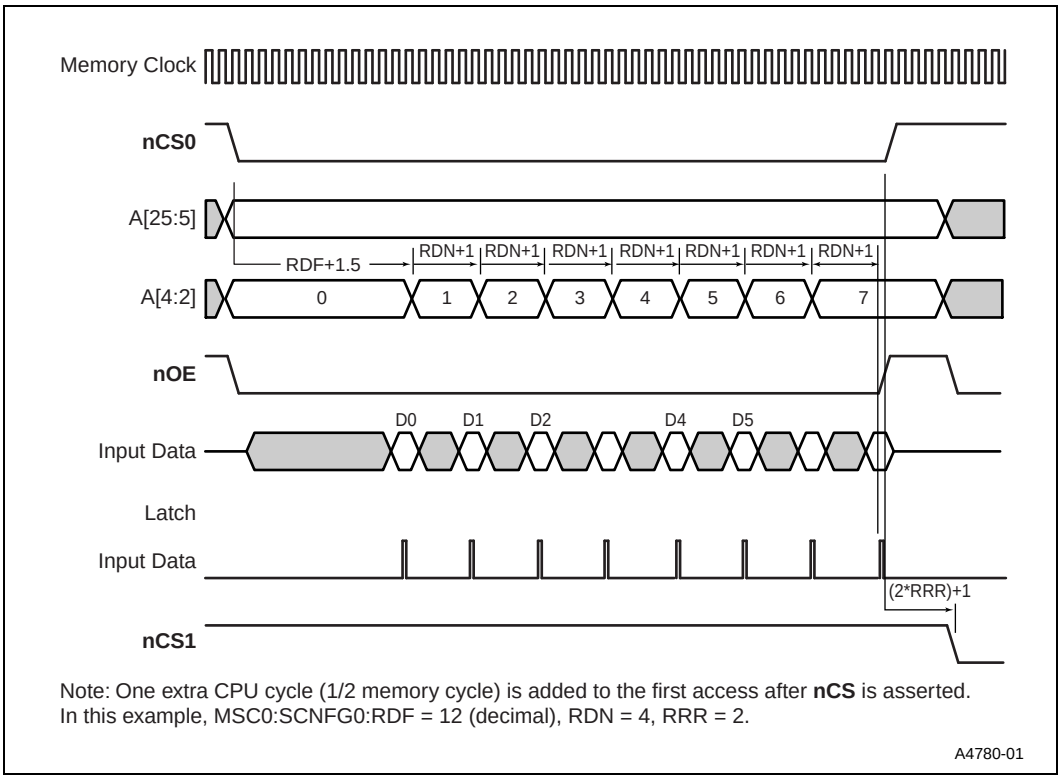
The SA-1100 provides programmable timing for both burst and nonburst ROMs. The RDF field in MSCx is the latency (in memory clock cycles) for nonburst ROMs and the first data beat of a burst ROM. RDN is the latency for the burst data beats after the first for burst ROMs. RRR delays the following access to a different memory space to allow time for the current ROM to tristate the data bus. This parameter should be programmed with the maximum tOFF value, as specified by the ROM manufacturer. One memory clock cycle is added to each of these parameters. At power-on reset, the SMCNFG0 field in the MSC0 register is initialized such that the RDF, RDN, and RRR fields are set to their maximum values to accommodate the slowest ROMs at initial boot; RT is set to be nonburst ROM; and RBW0 is loaded with the value of the inverse of the ROM_SEL pin. The remaining fields in MSC0 and MSC1 are not initialized on power-on reset. MSC0:SMCNFG0 is selected when the address space corresponding to nCS0 is accessed.

The SA-1100 supports a ROM burst size of 1, 4, or 8 words. A single DRAM CBR refresh cycle may be inserted between word accesses within a transaction. nCS and nOE are deasserted during the refresh cycle.

10.4.2 ROM Timing Diagrams and Parameters

Figure 10-6, Figure 10-7, and Figure 10-8 show the timing for burst and nonburst ROMS.

Figure 10-6. Burst-of-Eight ROM Timing Diagram



Note: One extra CPU cycle (1/2 memory cycle) is added to the first access after **nCS** is as
In this example, MSC0:SCNFG0:RDF=12(decimal), RDN=4, RRR=2.

Figure 10-7. Eight Beat Burst Read from Burst-of-Four ROM

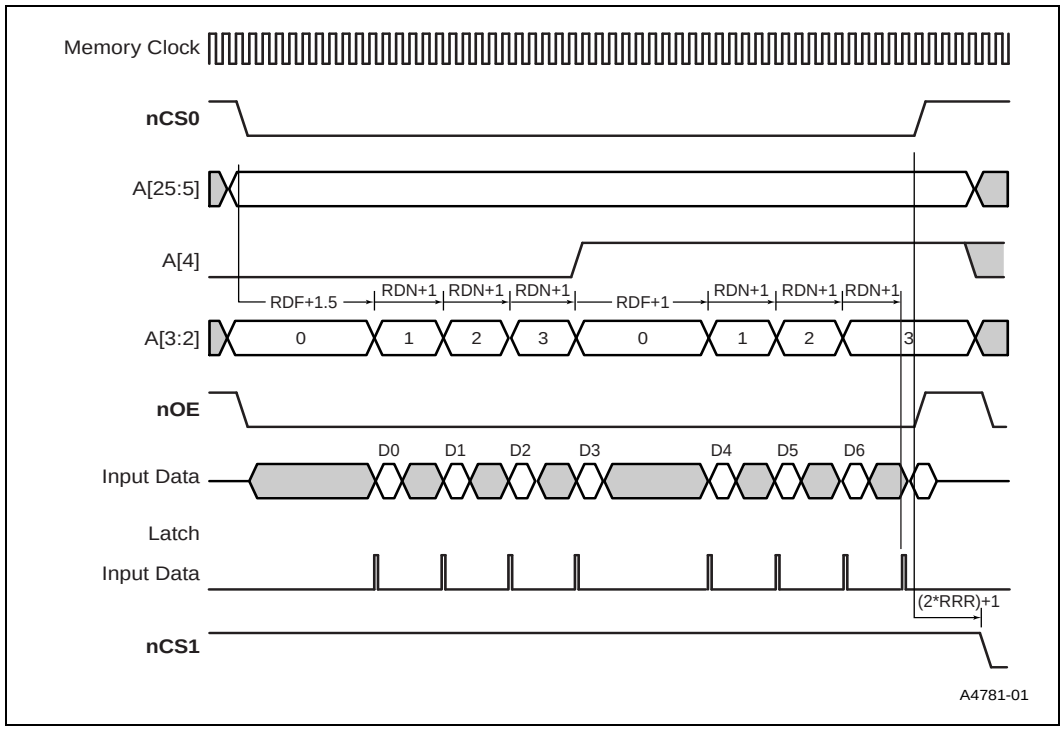
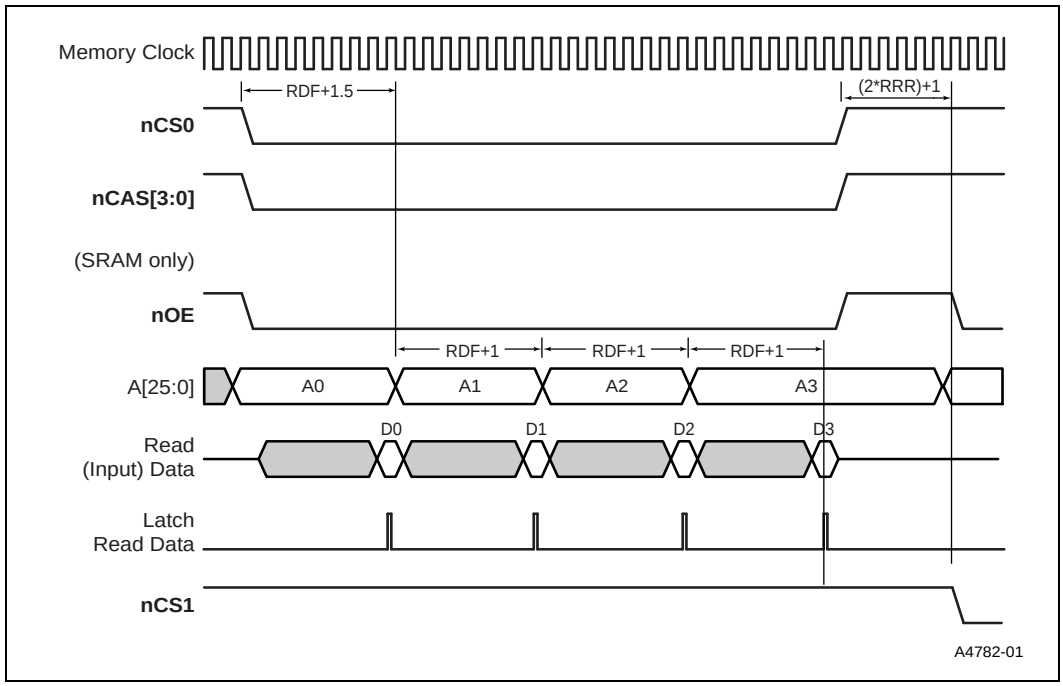


Figure 10-8. Nonburst ROM, SRAM, or Flash Read Timing Diagram – Four Data Beats



10.4.3 SRAM Interface Overview

The SA-1100 provides a 32-bit asynchronous SRAM interface that uses the nCAS pins for byte selects on both reads and writes (nCS<3:0> selects the SRAM bank, nOE is asserted on reads, and nWE is asserted on writes). Address bits A<25:2> provide addressability of up to 64 Mbyte of SRAM per bank. Because the nCAS signals are used to access SRAM, a system with both SRAM and DRAM is not supported.

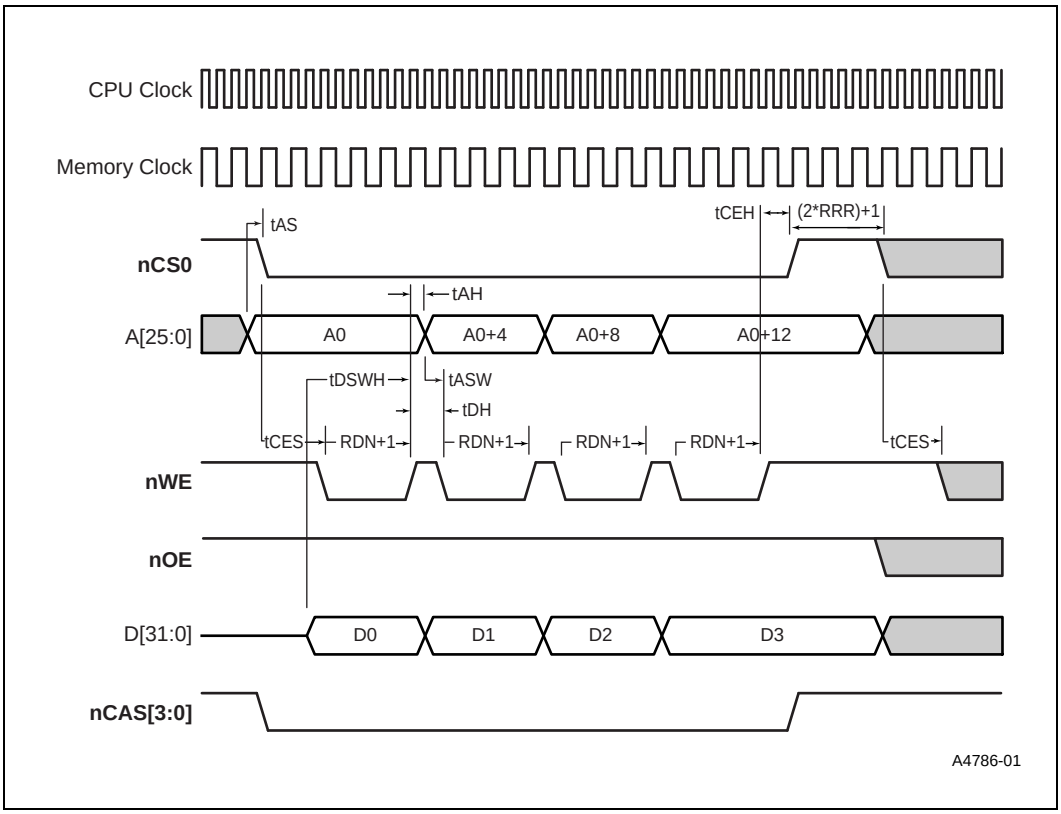
The timing for a read access is identical to that for a nonburst ROM. (See section 10.4.2 on page 19.) The RDF fields in the MSCx registers are the latency for a read access. The MSCx:RDN field controls the nWE low time during a write cycle. MSCx:RRR is the time from nCS deassertion after a read to the start of an access from a different memory bank or after a write to any other memory access. MSCx:RBW must be set to be a 32-bit bus and MSCx:RT must select SRAM.

10.4.4 SRAM Timing Diagrams and Parameters

SRAM reads have the same timing as nonburst ROMs as shown in Figure 10-8, except nCAS<3:0> are byte selects and are asserted with the same timing as nCS. When nCAS0 is low (asserted), D<7:0> will be used to transfer data. When nCAS1 is low, D<15:8> is used, and so on. During writes, all 32 data pins are actively driven by the SA-1100; they are not tristated regardless of the state of the individual nCAS pins.

Figure 10-9 shows the timing for SRAM writes.

Figure 10-9. SRAM Write Timing Diagram (4–Beat Burst)



In Figure 10-9, some of the parameters are defined as follows:

tAS = Address setup to nCS = 1 CPU cycle
tCES = nCS, nCAS setup to nWE = 2 memory clock cycles (4 CPU cycles)
tASW = Address setup to nWE low (asserted) = 1/2 memory cycle (1 CPU cycle)
[For A<25:5>, tASW=5 CPU cycles. For A<4:2>, tASW=1 CPU cycle for subsequent beats in a burst]
tDSWH = Write data setup to nWE high (deasserted) = 1/2 memory cycle + (RDN+1) memory cycles
tDH = Data hold after nWE high (deasserted) = 1/2 memory cycle (1 CPU cycle)
tCEH = nCS, nCAS held asserted after nWE deasserted = 1 memory clock cycle (2 CPU cycles)
tAH = Address hold after nWE deasserted = 1/2 memory cycle (1 CPU cycle)
nWE high time between burst beats = 1 memory cycle (2 CPU cycles)

10.4.5 FLASH EPROM Interface Overview

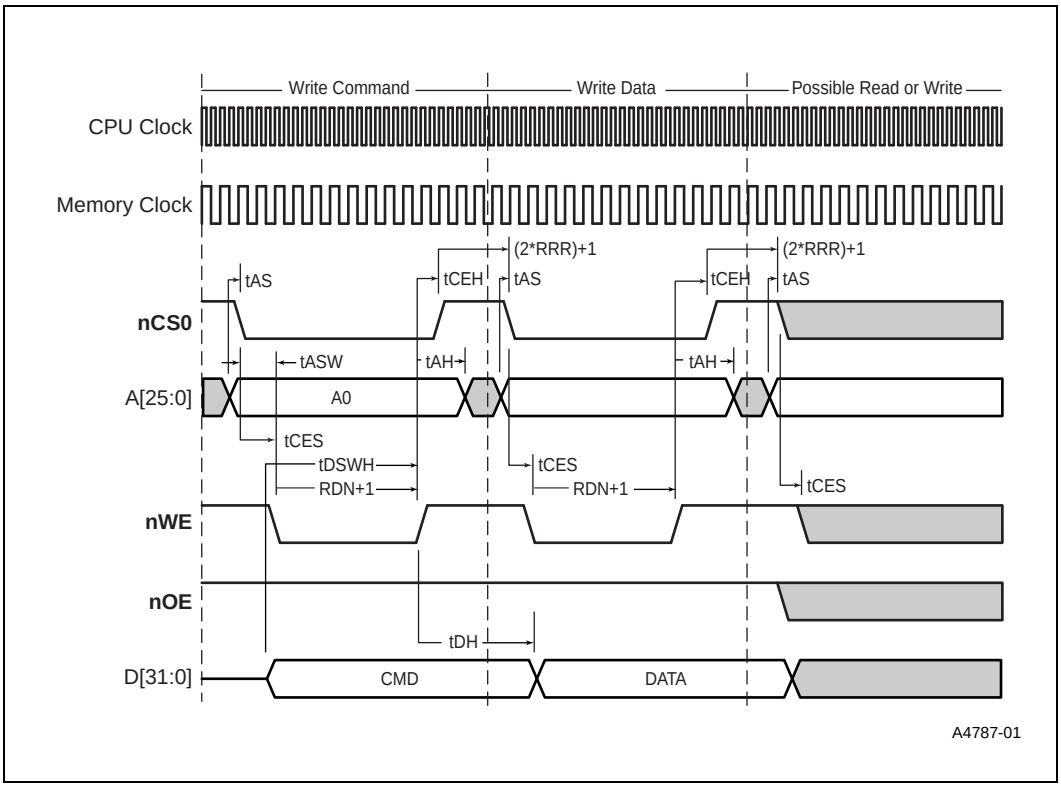
The SA-1100 provides an SRAM-like interface for access of Flash EPROM. The RDF fields in the MSCx registers are the latency for a read access. The RDN field controls the nWE low time during a write cycle. RRR is the time from nCS deassertion after a read to the start of a read from a different memory or after a write to another memory access. A single DRAM CBR refresh cycle may be inserted between words of a burst read from Flash memory. During the refresh cycle, nCS and nOE will be deasserted.

There are some requirements for writes to Flash memory. Flash memory space must be uncacheable and unbuffered. Writes must be exactly the width of the populated Flash devices on the data bus (no byte writes to a 32-bit bus or word writes to a 16-bit bus, and so on). Software is responsible for partitioning commands and data, and writing them out to Flash in the appropriate sequence.

10.4.6 FLASH EPROM Timing Diagrams and Parameters

Flash reads have the same timing as nonburst ROMs as shown in the preceding figures. Figure 10-10 shows the timing for Flash writes.

Figure 10-10. Flash Write Timing Diagram (2 Writes)



In Figure 10-10, some of the parameters are defined as follows:

tAS = Address setup to nCS = 1 CPU cycle
tCES = nCS setup to nWE = 2 memory clock cycles (4 CPU cycles)
tASW = Address setup to nWE low (asserted) = 2-1/2 memory cycles (5 CPU cycles)
tDSWH = Write data setup to nWE high (deasserted) = 1/2 memory cycle + (RDN+1) memory cycles
tDH = Data hold after nWE high = 1+1/2 memory cycle
tCEH = nCS held asserted after nWE deasserted = 1 memory clock cycle (2 CPU cycles)
tAH = Address hold after nWE deasserted = 1+1/2 memory cycle (3 CPU cycles)

10.5 General Memory BUS Timing

This section explains the boundary cases between DRAM, static, and refresh operations.

10.5.1 Static Access Followed by a DRAM Access

With a static memory access, nWE is deasserted 1 memory clock cycle prior to the deassertion of nCS. Then memory control will wait 2*RRR memory clock cycles (or 1, whichever is greater) before the assertion of nRAS for a DRAM access.

The SA-1100 always drives the data bus except while doing a read cycle (or while the alternate master mode is active). The delay from nOE asserted to data bus high-Z is approximately 0 ns. When nOE is deasserted, the data bus drives the same data that was already on the bus.

10.5.2 DRAM Access Followed by a Static Access

After a DRAM read cycle, the memory controller will wait TRP+1 memory cycles (or 2, whichever is greater) before nCS is asserted for a static memory access. nWE will be asserted 2 memory clock cycles after that for a total of TRP+3 memory clock cycles. For a static memory write after a DRAM write cycle, nWE will be asserted 3 memory clock cycles after nRAS is deasserted.

When nOE and nRAS are deasserted at the end of a DRAM ready cycle, the SA-1100 nCS<x> and nOE may be asserted for a static memory read, at which time the SA-1100 will stop driving in 0 ns. If the subsequent access is a static memory write, new data will be driven out TRP+1.5 memory clock cycles after the deassertion of nRAS and nOE. The minimum time between the end of a DRAM refresh cycle and nWE asserted is 3 memory clock cycles.

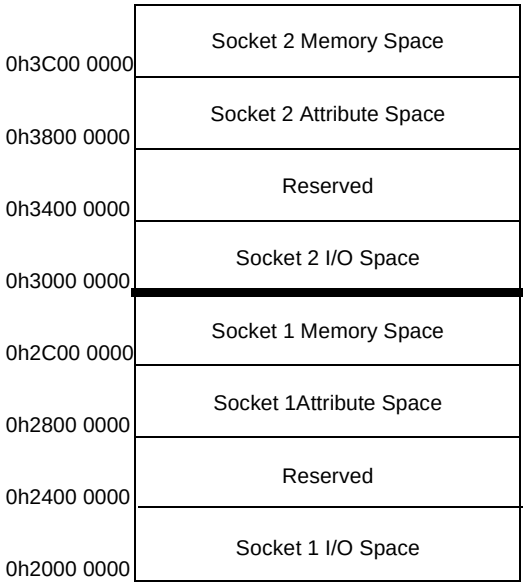
10.5.3 DRAM Access Followed by a Refresh Operation

At the end of a DRAM read/write cycle, nCAS will go high 1/2 to 1 memory clock cycles before nRAS goes high. For a subsequent refresh cycle, nCAS will go high TRP+1 memory clock cycles after the nRAS goes high. After that, nRAS will go high 2 memory clock cycles. In this case, TRP is used to hold off nCAS rather than just nRAS. There is no overlap (pipelining) between successive memory accesses.

10.6 PCMCIA Overview

The SA-1100 PCMCIA interface provides controls for one PCMCIA card slot with a PSKTSEL pin for support of a second slot. This 16-bit host interface supports 8- and 16-bit peripherals and handles common memory, I/O, and attribute memory accesses. The interface does not support the PCMCIA DMA protocol. The duration of each access is based on an internally generated clock that is programmed per address space in the MECR register. Figure 10-11 shows the memory map for the PCMCIA space.

Figure 10-11. PCMCIA Memory Map



The PCMCIA memory space is divided into eight partitions, four for each card slot. The four partitions for each card slot are common memory, I/O, attribute memory, and a reserved space. Each partition starts on a 64 Mbyte boundary. Pins A<25:0>, nPREG, and PSKTSEL are driven at the same time. nPCE1 and nPCE2 are driven at address time for memory and attribute accesses. For I/O accesses, their value depends on the value of nIOIS16 and thus will be valid a finite time after nIOIS16 is valid.

Common memory accesses assert the nPOE or nPWE control signals and are always 16-bit accesses with nPCE1 asserted for low byte access and nPCE2 asserted for high byte access. I/O accesses assert the nIOR or nIOW control signals and use the nIOIS16 input signal to determine the bus width of the transfer (8 or 16 bit). The SA-1100 uses nPCE2 to indicate to the expansion device that the upper half of the data bus, D<15:8>, will be used for the transfer and nPCE1 to indicate that the lower half of the data bus, D<7:0>, will be used. When nPCE2 is low, A<0> is ignored and an odd byte is transferred across D<15:8>. If nPCE2 is high and nPCE1 is low, then A<0> is used to determine whether the byte being transferred across D<7:0> is the odd byte or even byte. Transfers always start assuming a 16-bit bus. After the address is placed on the bus, an I/O device may respond with nIOIS16 indicating that it can perform the transfer in a single 16-bit transfer. If nIOIS16 is not asserted within the proper time, the address is assumed to be to two 8-bit registers and the transfer is completed as two 8-bit transfers on the low byte lane, D<7:0>, with nPCE2 deasserted, nPCE1 asserted, A<0> =0 for the first 8-bit transfer, and A<0> =1 for the second 8-bit transfer.

10.6.1 32-Bit Data Bus Operation

The SA-1100 PCMCIA interface supports the use of a 32-bit data bus. Because the PCMCIA 2.0 is 8- or 16-bit only, the 32-bit operation is outside the scope of the PCMCIA specification. This 32-bit mode is intended for use as a nonstandard expansion bus for communication with customer-designed logic. The operation is fairly simple; if a word read or write is performed to PCMCIA memory space, then the entire 32-bit bus is read or written. Normal PCMCIA operations should be performed using byte or half-word accesses only. Thirty-two bit accesses should be word aligned and only to "16-bit" space, as opposed to 8-bit space. Memory and attribute space is 16 bits by definition. However, I/O space may be 8- or 16-bit depending upon the state of the nIOIS16 input pin. Thirty-two bit accesses to I/O space require that the target assert nIOIS16.

For 32-bit accesses, the only size information present on the bus is the assertion of the nPCE1 and nPCE2 pins. This is the same information that is present during half-word accesses. As such, there is no way by looking at the SA-1100 pins to determine whether the access is a half-word or word. This information can be derived only though a user-defined address decode outside the SA-1100. The following table shows the operation of the PCMCIA interface and its relation to data width.

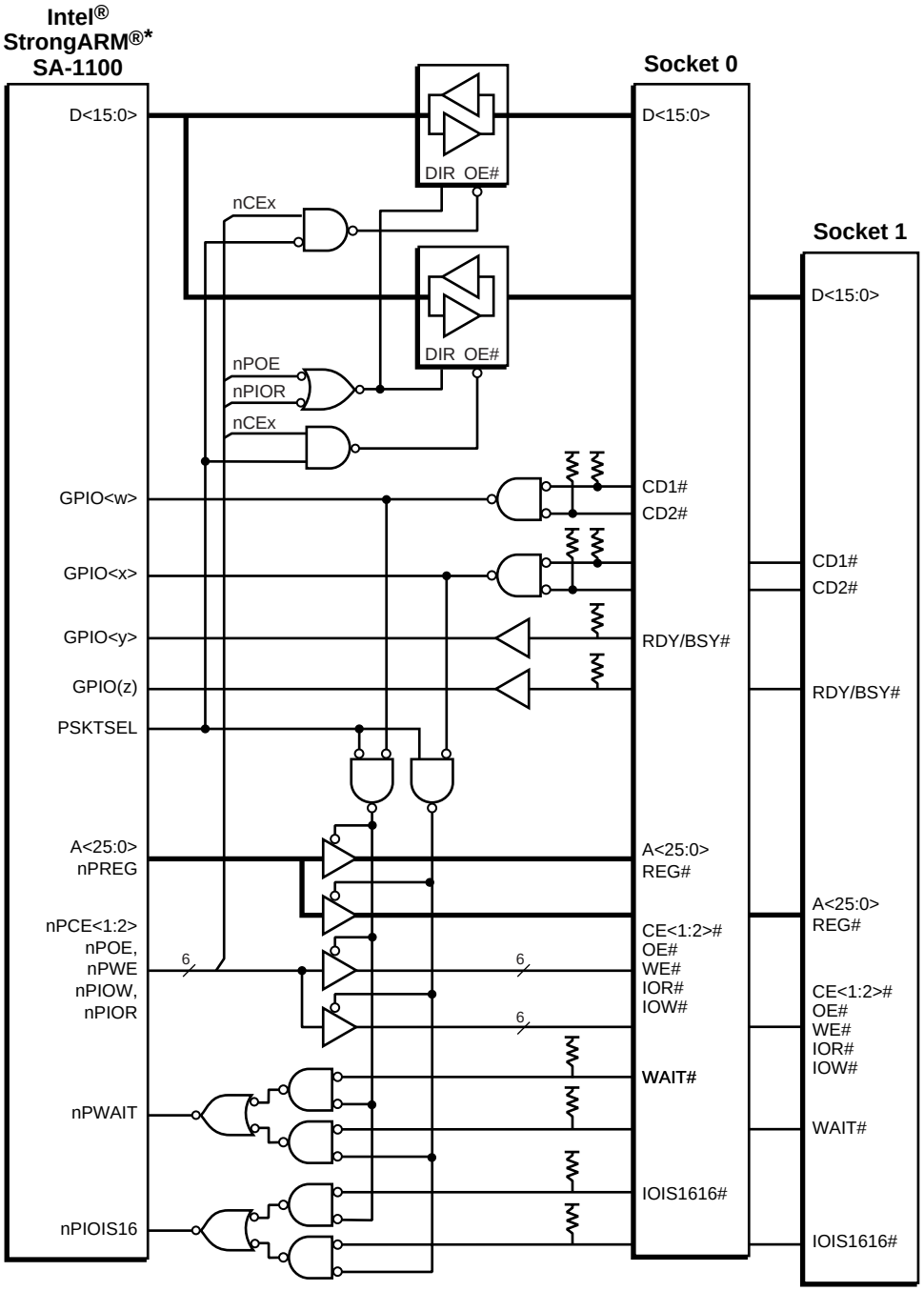
Access Type	Data Bus Width 1 = 16 Bit 0 = 8 Bit	Address (1:0)	Resulting Operation
Word	1	00	Word read or write, nPCE1 and nPCE2 asserted (low). nIOIS16 must be asserted for I/O space.
		1x	Undefined operation.
		x1	Undefined operation.
	0	xx	Undefined operation.
Half-word	1	x0 (even)	Single half-word access, nPCE1 and nPCE2 asserted (low). nIOIS16 must be asserted for I/O space.
		x1 (odd)	Undefined operation.
	0	x0 (even)	Two-byte accesses, both on the lower byte lane. Even access first (nPCE1 asserted and nPCE2 negated for both).
		x1 (odd)	Undefined operation.
Byte	1	x0 (even)	Load or store byte on the lower byte lane (nPCE1 asserted, nPCE2 negated).
		x1 (odd)	Load or store byte on the upper byte lane (nPCE1 negated, nPCE2 asserted).
	0	xx (even or odd)	Load or store byte on the low byte lane (nPCE2 negated and nPCE1 asserted).

10.6.2 External Logic for PCMCIA Implementation

The SA-1100 requires external logic to complete the PCMCIA socket interface. Figure 10-12 and Figure 10-13 show general solutions for a one- and two-socket configuration. Figure 10-14 shows a solution for the voltage-control circuit. These diagrams provide the logical connections necessary for support of 3 V and 5 V PCMCIA cards as well as hot insertion capability. For dual-voltage support, level shifting buffers are required for all signals. Hot insertion capability requires that each socket be electrically isolated from the other. If one or both of these features is not required, then some of the logic shown in these diagrams may be eliminated.

The pull-ups shown are included for compliance with the PCCARD xxx standard. Low power systems should remove power from these pull-ups during sleep to avoid unnecessary power consumption. The CD<2:1> signals have been “ORed” before being provided to the SA-1100. This signal is then routed into a GPIO pin for interrupt capability. Similarly, RDY/BSY is routed to a GPIO. The INPACK# signal is not used. In the data bus transceiver control logic, nCE1 should control the enable for the low byte lane and nCE2 should control the enable for the high byte lane.

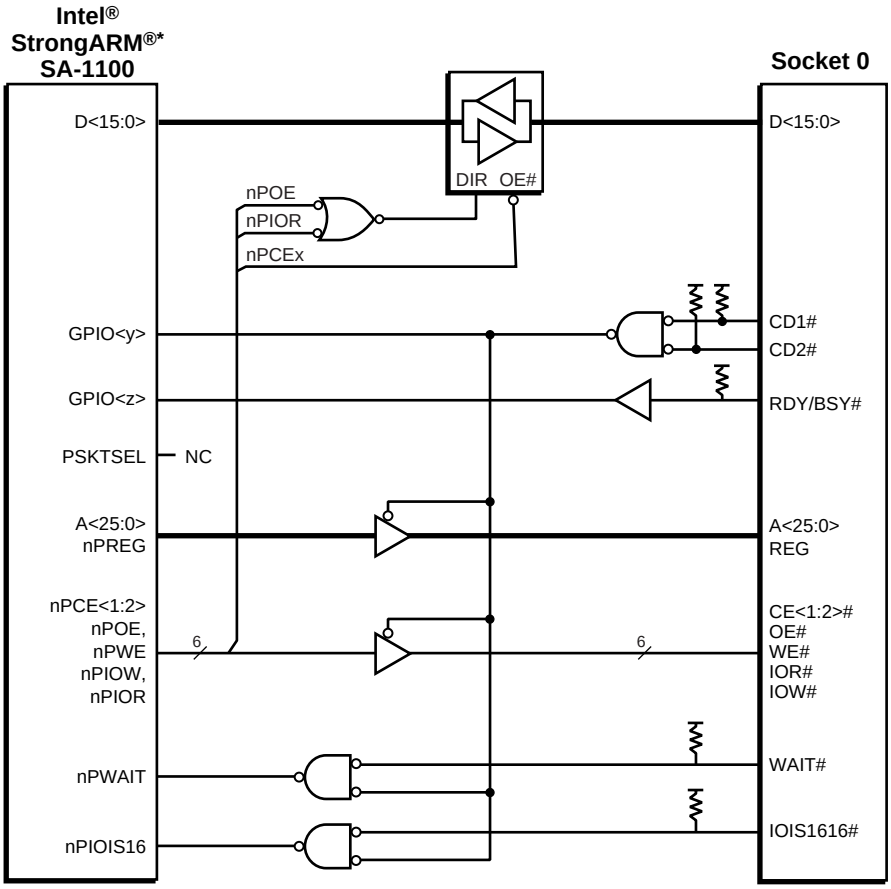
Figure 10-12. PCMCIA External Logic for a Two-Socket Configuration



* StrongARM is a registered trademark of ARM Limited.

A6840-01

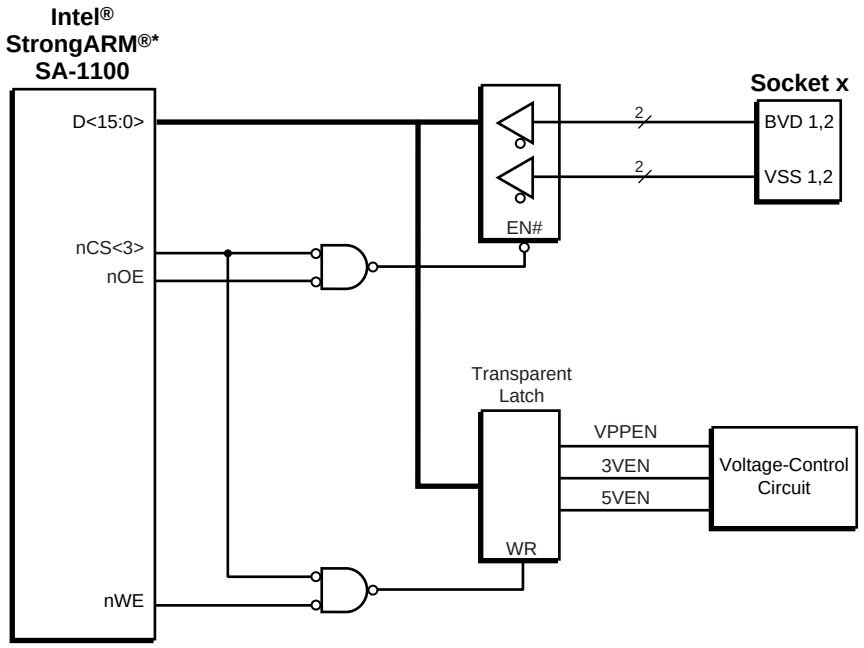
Figure 10-13. PCMCIA External Logic for a One-Socket Configuration



* StrongARM is a registered trademark of ARM Limited.

A6844-01

Figure 10-14. PCMCIA Voltage-Control Logic



* StrongARM is a registered trademark of ARM Limited.

A6845-01

The PCMCIA card voltage may be controlled through a set of discrete registers mapped into a static chip select. For example, Figure 10-14 shows mapping to chip select 3.

10.6.3 PCMCIA Interface Timing Diagrams and Parameters

Figure 10-15 shows a 16-bit access to a 16-bit memory or I/O device. The parameter, BS, is programmed in the MECR register. When common memory is accessed, the MECR:BSM1 or MECR:BSM2 field is used, depending on whether card socket 0 or 1 is addressed. MECR:BSIO1(2) is used for I/O accesses and MECR:BSA1(2) is used for access to attribute memory. Figure 10-15 and Figure 10-16 show the appropriate setting of BS_xx = 0b00001.

Figure 10-15. PCMCIA Memory or I/O 16-Bit Access

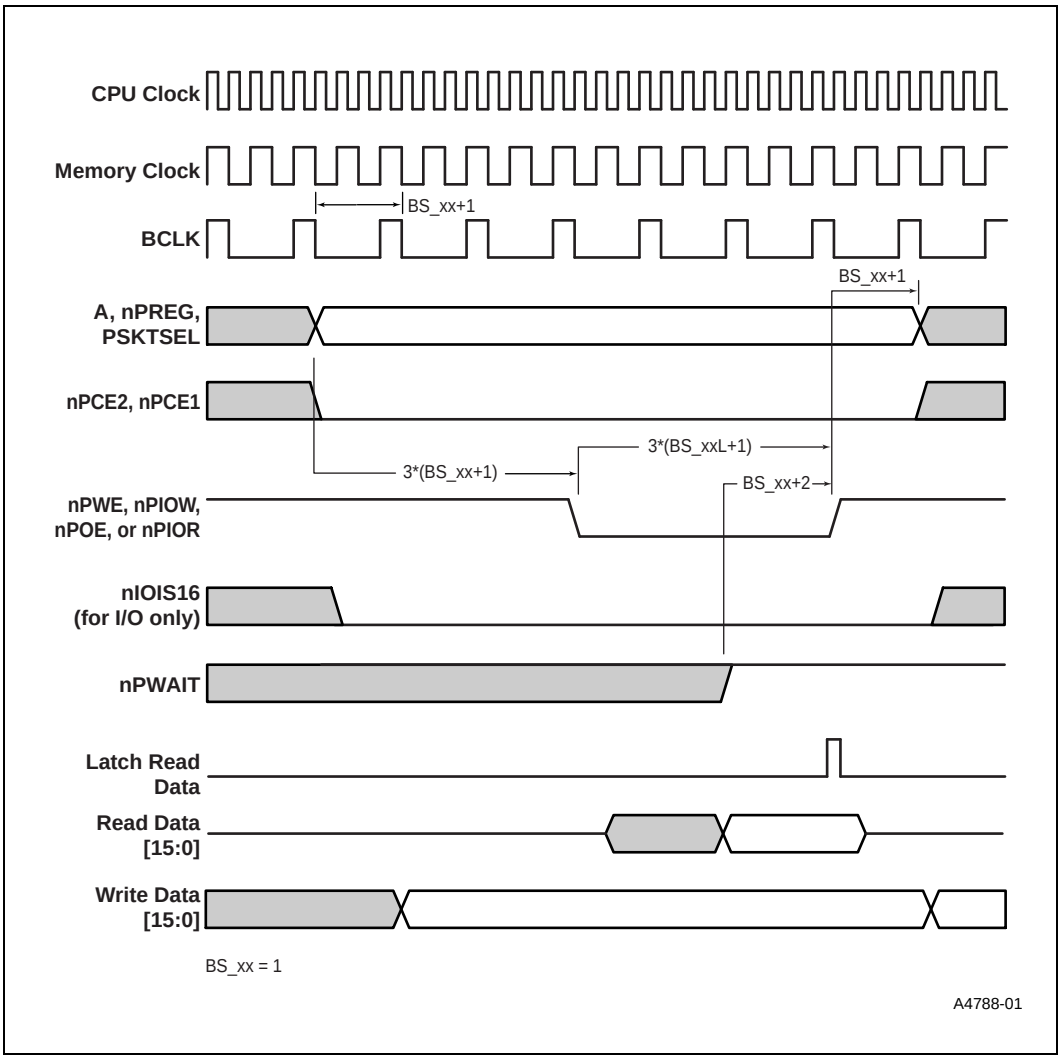
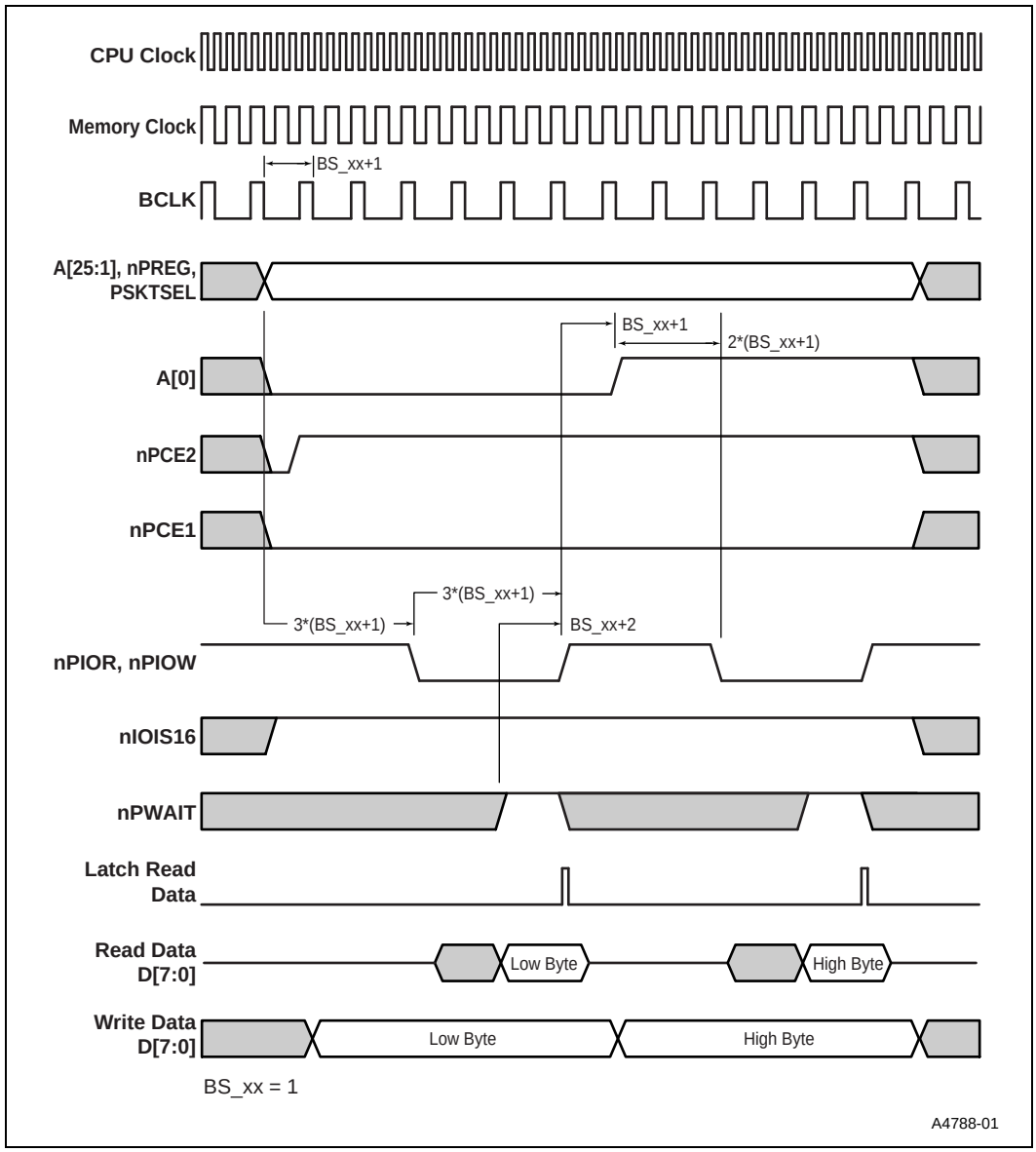


Figure 10-16. PCMCIA I/O 16-Bit Access to 8-Bit Device



Timing parameters are in CPU clock cycle units. All are minimums except as noted:

Address access time: 6*(BS_xx+1)
Command (nPOE, nPWE, nPIOR, nPIOW) assertion time: 3*(BS_xx+1)
Address setup to command assert: 3*(BS_xx+1)
Address hold after command deassertion: BS_xx+1
nPWAIT valid after command assertion (max): 2*(BS_xx+1) -1
Chip enable (nPCE1,2) setup to nPOE, nPWE assert: 3*(BS_xx+1)
Chip enable (nPCE1,2) setup to nPIOR, nPIOW assert: 3*(BS_xx+1) - (nIOIS 16 delay from address)
Chip enabled hold from command deassert: BS_xx+1

See Chapter 13, “AC Parameters” for actual AC timing.

10.7 Initialization of the Memory Interface

On power-on reset, the dynamic memory interface is disabled and the static interface for the boot ROM, connected to nCS0, is configured for the slowest nonburst ROM/Flash EPROM. The ROM_SEL pin determines the bus size of the boot ROM (nCS0).

Initialization software is responsible for setting up the memory interface configuration registers before enabling the DRAM interface by setting MDCNFG:DE3-0.

Most DRAMs require a wait period followed by a series of refresh cycles before the first memory access. The SA-1100 provides a mechanism for software to control these events. When a particular DRAM bank (bank n, selected by nRAS) is disabled (MDCNFG:DEn=0), a read from any address in that bank will trigger a CBR refresh cycle for all banks.

10.7.1 Flow of Events After Reset or Exiting Sleep Mode

On power-on reset, the memory controller is in the following state:
nRAS(3:0) = 0xF
nCAS(3:0) = 0xF
nCS(3:0) = 0xF
nOE = 1
nWE = 1
nPIOR = 1
nPIOW = 1
nPOE = 1
nPWE = 1

All DRAM banks disabled (MDCNFG:DE3:0 = 0).

Static interface set to slowest nonburst ROM/Flash timing.

(MSC0:SMCNFG0 field is initialized as follows:

RRR=0xF, RDN=0x1F, RDF=0x1F, RBW = not ROM_SEL, RT=0)

Upon exiting sleep mode, the memory controller is in a state similar to reset, except the **nCAS** and nRAS pins remain asserted to ensure that the DRAMs remain in a self-refresh state until the processor has been configured:
nRAS(3:0) = 0
nCAS(3:0) = 0
nCS(3:0) = 0xF
nOE = 1
nWE = 1
nPIOR = 1
nPIOW = 1
nPOE = 1
nPWE = 1

All DRAM banks disabled (MDCNFG:DE3:0 = 0).

Static interface set to slowest nonburst ROM/Flash timing.
(MSC0:SMCNFG0 field is initialized as follows:
RRR=0xF, RDN=0x1F, RDF=0x1F, RBW = not ROM_SEL, RT=0)

The following flow should be followed when coming out of reset, whether for sleep or power-up:

- Read boot ROM and write to memory configuration registers, but do not enable DRAM banks.
- If necessary, finish any DRAM power-up wait period (usually about 100 µs).
- If coming out of sleep, see Section 9.5, “Power Manager” on page 9-26 on how to release the nCAS and nRAS pins from their self-refresh state.
- If coming out of sleep, wait the DRAM-specific post-self-refresh precharge period before issuing a new DRAM transaction.
- If power-on reset, perform the number of initialization refreshes required by the specific DRAM part by reading disabled banks. A read from any disabled bank will refresh all four banks.
- Enable DRAM banks by setting MDCNFG:DE3:0.

10.8 Alternate Memory Bus Master Mode

The SA-1100 supports the existence of an alternate master on the memory bus. The alternate master is given control of the memory bus (address, data, RAS, CAS, and static controls) using a hardware handshake. This handshake is performed through MBREQ and MBGNT, which are invoked through the alternate functions on GPIO<22> and GPIO<21> respectively. When the alternate master wants to take control of the memory bus, it asserts MBREQ (GPIO<22>). The SA-1100 will then complete any pending or in-progress memory operation and any outstanding DRAM refresh cycle and then assert MBGNT (GPIO<21>). When the alternate master asserts MBGNT, the SA-1100 will tristate the memory bus pins (A<25:0>, D<31:0>, nCS<3:0>, nOE, NWE, nRAS<3:0>, nCAS<3:0>).

During the tristate period, both MBREQ and MBGNT remain high and an external device may take control of the tristated pins. It is recommended that the external device drive all the pins even if some are not actually used. This will prevent floating inputs and the crossover current associated with them. Note that during the tristate period, the SA-1100 is unable to perform DRAM refresh cycles. The alternate master must assume the responsibility for DRAM integrity during this period. It is recommended that the system be designed such that the period of alternate mastership is limited to much less than the refresh period, or that the alternate master implement a refresh counter making it capable of performing refresh at the proper intervals.

To give up the bus, the alternate master negates MBREQ. The SA-1100 will then negate MBGNT and begin driving the bus. If the refresh counter inside the SA-1100 requested a refresh cycle during the alternate master tenure, then that refresh cycle is run first, followed by any other bus transactions that stalled during that period. This mode is set up by writing to the following registers:

- GPIO pin direction register to program GIO<21> as an output and GPIO<22> as an input.
- GPIO alternate function register to program GPIO<21> and GPIO<22> to their alternate function.
- Test unit control register (TUCR) to set bit 10.

Peripheral Control Module 11

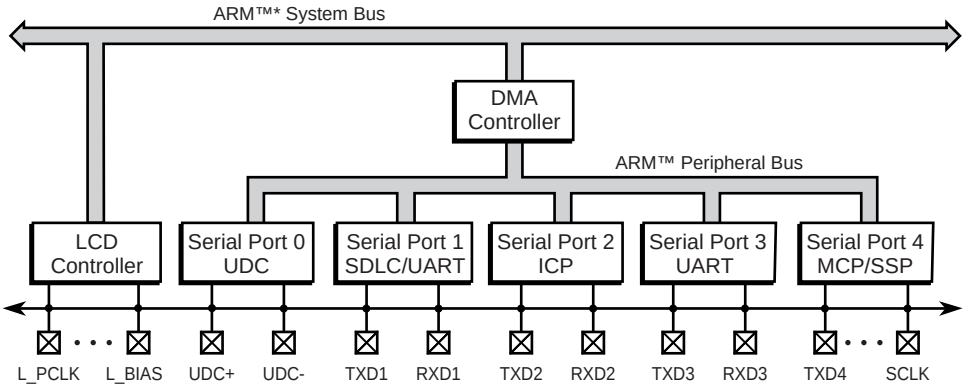
This chapter describes the peripheral control units that are integrated within the Intel® StrongARM® SA-1100 Microprocessor (SA-1100) and the DMA controller that services them. The peripheral units include one parallel data port to drive an LCD display, one synchronous serial port, and four asynchronous serial ports that implement different serial protocol standards. Each section includes a description of the unit's operation and the control, data, and status registers used to configure the unit. The DMA controller acts as the gateway to the peripheral units. It provides DMA access to these units and control and address decode for programmed I/O accesses between the processor and registers inside the units. Note that the LCD controller contains its own high bandwidth DMA controller that is connected to the ARM™ system bus and is used to read pixel and palette information from the off-chip frame buffer.

11.1 Read/Write Interface

The ARM system bus, shown in Figure 11-1, is a high-performance synchronous bus that connects the peripheral control module to the SA-1100 CPU and to the external memory controller. The DMA connects the ARM system bus to the ARM peripheral bus. The ARM peripheral bus implements a standard asynchronous protocol that is used by all peripherals designed for ARM chips. This standard allows a single library of peripherals to be developed for the entire ARM family of CPUs, providing a means to quickly spin off new chip implementations that contain different peripheral mixes for target applications. Note that the LCD controller interfaces to the ARM system bus because its throughput requirement is much higher than that of any other serial peripheral. Placing the LCD on the ARM system bus allows faster synchronous transfers to be made between the external frame buffer and the LCD controller. Additionally, the LCD controller contains its own dual-channel DMA controller to supply frame buffer data to the unit.

Although the ARM peripheral bus supports 32 bits of data, the register size (width) implemented for each peripheral is equal to the maximum data size that must be coherently read or written by the CPU and DMA. This minimizes the size of the peripheral while providing the necessary memory throughput for the unit. Although the peripherals' register sizes vary, the ARM peripheral bus does not support byte or half-word accesses. Only word accesses are allowed. Table 11-1 shows the register width, DMA port size, and DMA burst size of each of the six peripherals (and the PPC) implemented on the SA-1100.

Figure 11-1. Peripheral Control Module Block Diagram



* ARM is a trademark of ARM Limited.

A6833-01

Table 11-1. Peripheral Control Modules' Register Width and DMA Port Size

Peripheral		Register Width / DMA Port Size	DMA Burst Size
LCD controller		32	4 words
Serial port 0: UDC		8	8 bytes
Serial port 1:	UART	8	4 bytes
	SDLC	8	4 bytes
Serial port 2: ICP	UART	8	4 bytes
	HSSP	8	8 bytes
Serial port 3: UART		8	4 bytes
Serial port 4:	MCP	16	8 bytes
	SSP	16	8 bytes
Peripheral pin controller (PPC)		32	N/A

11.2 Memory Organization

Several of the serial ports contain more than one serial engine. Each individual engine is self-contained (no shared logic or registers) and implements a separate serial protocol. Serial ports 1, 2, and 4 each contain two separate serial engines, totalling eight separate serial engines within all five serial ports. Each of the eight serial engines, including the peripheral pin controller (PPC), has been allocated a separate 64 Kbyte block on-chip memory space in which its registers reside. Although the register width of individual units varies, each register is right justified on word boundaries. All register accesses via the CPU must be performed using word reads and writes. This chapter includes a summary of individual peripheral registers. See Appendix A, "Register Summary" for a complete summary of all on-chip registers.

Table 11-2 shows the base address for each of the peripheral control units.

Table 11-2. Peripheral Units' Base Addresses

Peripheral	Serial Protocol	Base Address
LCD Controller		0h B010 0000
Serial Port 0	USB	0h 8000 0000
Serial Port 1	UART	0h 8001 0000
	SDLC	0h 8002 0000
Serial Port 2 (ICP)	UART	0h 8003 0000
	HSSP	0h 8004 0000
Serial Port 3	UART	0h 8005 0000
Serial Port 4	MPC	0h 8006 0000
	SSP	0h 8007 0000
Peripheral Pin Controller (PPC)[1]		0h 9006 0000

[1] The PPC does not support DMA requests.

11.3 Interrupts

Each peripheral unit interfaces to the interrupt controller within the system control module. The interrupt controller contains a 32-bit interrupt pending register, which when read, informs the user of all the units on the SA-1100 that are currently generating an unmasked interrupt. Once the user determines which unit is causing the interrupt, the unit's status registers can be read to determine the exact cause of the interrupt. This mechanism provides a two-level approach to identify the source of any interrupt from the hundreds of possible interrupt sources that exist on the SA-1100.

Each of the peripheral units generate either one or two interrupts that correspond to specific interrupt pending bits within the interrupt controller. Serial ports 1 and 4 each contain two independent serial engines. Although each peripheral uses only one set of pins for serial communication, the user may choose to use both serial engines within serial ports 1 and 4 by assigning one of the two protocols to communicate off-chip by taking control of GPIO pins. Because the two engines within serial ports 1 and 4 can operate at the same time, these two units are assigned two separate interrupt request numbers within the interrupt controller's pending register. Table 11-3 shows the interrupt level for each of the peripheral control units.

Table 11-3. Peripheral Units' Interrupt Numbers

Peripheral		Interrupt Number
LCD controller		12
Serial port 0: USB		13
Serial port 1:	SDLC	14
	UART	15
Serial port 2: ICP		16
Serial port 3: UART		17
Serial port 4:	MCP	18
	SSP	19

11.4 Peripheral Pins

Each peripheral has a number of dedicated pins with which to communicate to off-chip devices. The six peripherals of the SA-1100 use a total of 24 pins: the LCD uses twelve pins; serial port 4 four pins; and serial port 0 through 3 each use two pins. Many applications may not require the use of all six of the SA-1100's peripherals. To provide maximum flexibility, the pins associated with any unused peripheral (except serial port 0) can be used as general-purpose digital input/output pins that are noninterruptible. When a peripheral is disabled, the peripheral pin controller (PPC) automatically takes control of the peripheral's pin direction and pin state. A user can sample input pin state by reading the PPC pin state register (PPSR) and control the state of an output pin by writing to it. Pin direction is established by configuring the PPC pin direction register (PPDR). Table 11-4 shows a list of the pins associated with the peripheral units.

Table 11-4. Dedicated Peripheral Pins

Peripheral	GPIO Pin	Function
LCD Controller	L_PCLK	Pixel clock
	L_LCLK	Line clock/horizontal sync pulse
	L_FCLK	Frame clock/vertical sync pulse
	L_BIAS	A/C bias signal
	LDD<7:0>	Pixel data
Serial port 0: USB	UDC+	Positive differential receiver
	UDC-	Negative differential receiver
Serial port 1: SDLC/UART	TXD1	Serial transmit data
	RXD1	Serial receive data
Serial port 2: ICP	TXD2	Serial transmit data
	RXD2	Serial receive data
Serial port 3: UART	TXD3	Serial transmit data
	RXD3	Serial receive data
Serial port 4: MPC/SSP	TXD4	Serial transmit data
	RXD4	Serial receive data
	SCLK	Serial clock
	SFRM	Serial frame clock

11.5 Use of the GPIO Pins for Alternate Functions

Each of the SA-1100's six peripheral units has a number of dedicated pins that can be used to drive an LCD display, communicate serially with off-chip devices, or be used as general-purpose digital input/output pins. Each of the peripherals, except serial port 0 and 2, also has programming options that allow the unit to take over control of one or more GPIO pins from the system control module to be used for various special functions. Several control bits must be programmed to enable GPIO use by peripheral units. First, the user must enable the special function either within the peripheral unit or within the peripheral pin controller (PPC). Second, the user must enable the GPIO pin to communicate to the peripheral and select the pin's direction by programming the GPIO alternate function register (GAFR) and GPIO pin direction register (GPDR), respectively. See Section 9.1, "General-Purpose I/O" on page 9-1 for a description of these GPIO registers. Table 11-5 shows the GPIO pins that can be used for alternate peripheral pin functions.

Table 11-5. Peripheral Unit GPIO Pin Assignment

Peripheral	GPIO Pin	Function
LCD Controller	GPIO<2>	LDD<8> pin for dual-panel color mode.
	GPIO<3>	LDD<9> pin for dual-panel color mode.
	GPIO<4>	LDD<10> pin for dual-panel color mode.
	GPIO<5>	LDD<11> pin for dual-panel color mode.
	GPIO<6>	LDD<12> pin for dual-panel color mode.
	GPIO<7>	LDD<13> pin for dual-panel color mode.
	GPIO<8>	LDD<14> pin for dual-panel color mode.
	GPIO<9>	LDD<15> pin for dual-panel color mode.
Serial port 0: USB	N/A	None.
Serial port 1: SDLC/UART	GPIO<14>	Transmit pin for UART when SDLC and UART both needed.
	GPIO<15>	Receive pin for UART when SDLC and UART both needed.
	GPIO<16>	Sample clock input/output to SDLC.
	GPIO<17>	Toggle to drive external tristate for SDLC transmit packets.
	GPIO<18>	Sample clock input to UART.
Serial port 2: ICP	N/A	None.
Serial port 3: UART	GPIO<20>	Sample clock input to UART.
Serial port 4: MPC/SSP	GPIO<10>	Transmit pin for SSP when MCP and SSP both needed.
	GPIO<11>	Receive pin for SSP when MCP and SSP both needed.
	GPIO<12>	SCLK pin for SSP when MCP and SSP both needed.
	GPIO<13>	SFRM pin for SSP when MCP and SSP both needed.
	GPIO<19>	Clock input pin for SSP to drive the frame and sample rates when other than nonmultiple of 3.6864 MHz needed.
	GPIO<21>	Clock input pin for MCP to drive the frame and sample rates when other than 12 Mbps needed.

11.6 DMA Controller

The DMA controller consists of six independent DMA channels. Each channel can be configured to service any of the serial controllers. Two channels are required to service a full-duplex serial controller. The DMA controller is intended to relieve the processor of the interrupt overhead in servicing these ports with programmed I/O. If desired, any or all peripherals (except the UDC) may be serviced with programmed I/O instead of DMA. Each peripheral is capable of requesting processor service through its own interrupt lines or through a DMA request.

The DMA controller consists of a set of configuration and control registers for each channel and a common data transfer engine that services the active channel. Channels are serviced in a fixed priority sequence if the DMA receives multiple requests. Each channel is serviced in increments of that device's burst size and delivered in the granularity of that device's port width (byte or half-word). The burst size and port width for each device is programmed in the channel registers and is based on the device's FIFO depth and bandwidth needs. When multiple channels are actively executing, each channel is serviced with a burst of data after which the DMA controller may perform a context switch to another active channel. The DMA controller performs context switches based on whether a channel is active, whether its target device is currently requesting service (the FIFO is half-empty), and where that channel lies in the priority scheme.

Data transfers are performed between a device (one of the serial controllers) and memory (ROM, RAM, Flash, SRAM, or DRAM). DMA transfers to and from PCMCIA space are not permitted. During a write, a burst of data is read from memory as words into a buffer inside the DMA controller. That data is then written to the device according to the device's port width and the state of the endian bit (E). During a read, data is read from the device according to the device's port width and then sent to memory as words. The organization of the bytes inside that word is determined again by the endian bit (E).

The control registers for each channel include two starting address registers and two transfer count registers. These registers should be programmed by the system at the start of the transfer. The registers control two rotating buffers for use during a transfer. These buffers, designated buffer A and buffer B, can be chained together so that when a transfer to (or from) one buffer completes, the transfer to (or from) the other begins immediately. By interrogating the status information in the channel control/status register, the user can safely update the address pointer and transfer count of the inactive buffer.

11.6.1 DMA Register Definitions

Each DMA channel is supported by six 32-bit registers as part of the DMA controller hardware. These registers are the DMA device address register (DDARn), DMA control/status register (DCSRn), DMA buffer A start address (DBSAn), DMA buffer B start address (DBSBn), DMA buffer A transfer count (DBTAn), and DMA buffer B transfer count (DBTBn). (The n is a value from 0 to 5 and is the channel number.) A register summary including physical addresses is provided at the end of this section.

11.6.1.1 DMA Device Address Register (DDARn)

The DDARn is a 32-bit read/write register containing channel information regarding the target device. Writes to this register are blocked if the RUN bit in the DCSRn is one. The following figure shows the format for this register; question marks indicate that the values are unknown at reset.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read	DA 31	DA 30	DA 29	DA 28	DA 27	DA 26	DA 25	DA 24	DA 23	DA 22	DA 21	DA 20	DA 19	DA 18	DA 17	DA 16
Reset	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DA 15	DA 14	DA 13	DA 12	DA 11	DA 10	DA 9	DA 8	DS 3	DS 2	DS 1	DS 0	DW	BS	E	RW
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	Name	Description
0	RW	Device data transfer direction (read/write). 0 = Transfer is a write (memory to device). 1 = Transfer is a read (device to memory).
1	E	Device endianess. 0 = Byte ordering is little endian. 1 = Byte ordering is big endian.
2	BS	Device burst size. 0 = Four datums per burst. 1 = Eight datums per burst.
3	DW	Device datum width. 0 = Datum size is one byte. 1 = Datum size is one half-word.
7..4	DS<3:0>	Device select. This field is programmed to point to the desired device.
31..8	DA<31:8>	Device address field. This field is a partial address of the data port of the device currently being serviced. [1]

[1] "Partial" means that certain bits in the address are assumed to be zero. The **DA<31:8>** field is constructed as follows:

DA<31:28> = Device port address 31:28.

Device port address 27:22 is assumed to be zero.

DA<27:8> = Device port address 21:2.

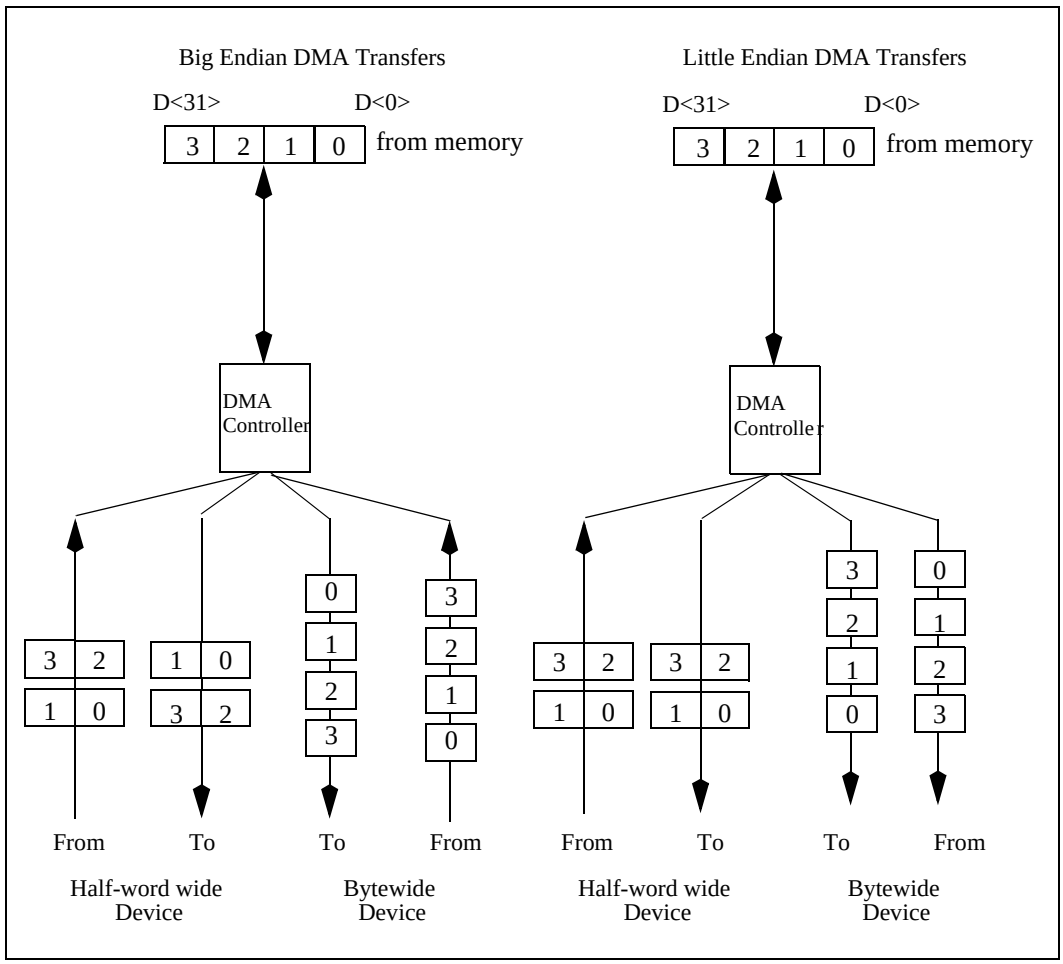
Device port address 1:0 is assumed to be zero.

The value written to the device select DS<3:0> field specifies which DMA request this channel responds to. The device datum width (DW) field value is fixed for each device type and indicates whether the device's data port is one or two bytes wide. If the datum width is programmed incorrectly for a particular device select, then the results are unpredictable.

The device burst size (BS) field value is fixed for each device type. It indicates how many beats of the datum width are transferred each time the device requests service. This value is chosen based on the FIFO size of the particular device. If the burst size is programmed incorrectly for a particular device select, then the results are unpredictable.

The device endianess E field value indicates the byte ordering within a word when data is read from or written to memory. If the E bit is zero, then memory is assumed to be little endian. If the bit is one, then memory is assumed to be big endian. The following figure shows big and little endian DMA transfers.

Figure 11-2. Big and Little Edian DMA Transfers



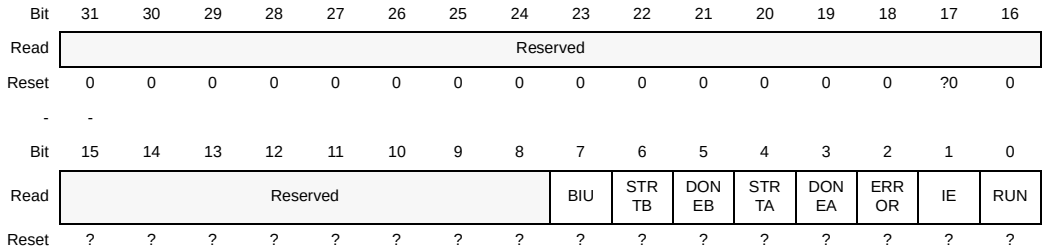
The device transfer direction (RW) field indicates the direction of the transfer. A zero indicates that the transfer is a write (with respect to the device) and that the flow of data will be from memory to the device. If the RW field is programmed to a one, then the transfer is a read and the flow of data will be from the device to memory. The transfer direction is fixed for each device type. If the burst size is programmed incorrectly for a particular device select, then the results are unpredictable.

Table 11-6. Valid Settings for the DDARn Register

Unit Name	Function	Device Address	DDAR Fields					
			DA<31:8>	DS<3:0>	DW	BS	E	RW
Serial port 0	UDC transmit	0x 8000 0028	0x80000A	0000	0	1	0/1	0
	UDC receive	0x 8000 0028	0x80000A	0001	0	1	0/1	1
Serial port 1	SDLC transmit	0x 8002 0078	0x80801E	0010	0	0	0/1	0
	SDLC receive	0x 8002 0078	0x80801E	0011	0	0	0/1	1
	UART transmit	0x 8001 0014	0x804005	0100	0	0	0/1	0
	UART receive	0x 8001 0014	0x804005	0101	0	0	0/1	1
Serial port 2	HSSP transmit	0x 8004 006C	0x801001B	0110	0	1	0/1	0
	HSSP receive	0x 8004 006C	0x801001B	0111	0	1	0/1	1
	UART transmit	0x 8003 0014	0x80C005	0110	0	0	0/1	0
	UART receive	0x 8003 0014	0x80C005	0111	0	0	0/1	1
Serial port 3	UART transmit	0x 8005 0014	0x814005	1000	0	0	0/1	0
	UART receive	0x 8005 0014	0x814005	1001	0	0	0/1	1
Serial port 4	MCP transmit (audio)	0x 8006 0008	0x818002	1010	1	0	0/1	0
	MCP receive (audio)	0x 8006 0008	0x818002	1011	1	0	0/1	1
	MCP transmit (telecom)	0x 8006 000C	0x818003	1100	1	0	0/1	0
	MCP receive (telecom)	0x 8006 000C	0x818003	1101	1	0	0/1	1
	SSP transmit	0x 8007 006C	0x81C01B	1110	1	0	0/1	0
	SSP receive	0x 8007 006C	0x81C01B	1111	1	0	0/1	1

11.6.1.2 DMA Control/Status Register (DCSRn)

The DCSRn is a 32-bit read/write register that contains control and status bits for the channel. The following figure shows the format for this register; question marks indicate that the values are unknown at reset.



Bit	Name	Description
0	RUN	Run bit. This is a control bit and is set by the user to indicate that the device address register has been loaded. No transfer will occur on this channel unless this bit is set. Clearing the RUN bit on an active channel acts as a pause to that channel. Operation can then be resumed by again setting the RUN bit.
1	IE	Interrupt enable. This bit enables interrupts to be passed onto the interrupt controller. An interrupt is the “OR” of the DONEA, DONEB, and ERROR bits.
2	ERROR	Transfer error bit. ERROR is a status bit and is set to indicate that a memory error has occurred. It can generate an interrupt if the IE bit is set. ERROR is cleared by software through setting the RUN bit.
3	DONEA	Buffer A done. This bit is a status bit and indicates that the transfer into or out of buffer A has completed. It is cleared by writing a one to it or by setting the STRTA bit. DONEA can generate an interrupt if IE is set.
4	STRTA	Buffer A transfer start. This bit is a control bit and is written by the user. It causes the buffer A transfer to begin. This bit is functional only if the RUN bit is set.
5	DONEB	This bit is a status bit and indicates that the transfer into or out of buffer B has completed. It is cleared by writing a one to it or by setting the STRTB bit. DONEB can generate an interrupt if IE is set.
6	STRTB	Buffer B transfer start. This bit is a control bit and is written by the processor. It causes the buffer B transfer to begin. This bit is functional only if the RUN bit is set.
7	BIU	Buffer in use. BIU is a status bit and may be read to indicate which buffer (A or B) is active . This bit is toggled by the DMA controller when DONEA or DONEB are set. This bit is cleared by all reset sources (hard, sleep, watchdog, or software).
8..31	—	Reserved. These bits are reserved and read as zeros. Writes to this field have no effect.

The RUN bit is the channel enable. It should be written to a one when the channel is ready for a transfer. It can also be used to pause the channel in the middle of a transfer; when it is set to a one again, the channel will resume from the current pointer value using the current active buffer. If the RUN bit is cleared in the middle of a burst, the burst will complete before the channel is paused. The DDAR may be written only when RUN is zero.

The IE bit is the interrupt enable for the channel. An interrupt is generated if the DONEA, DONEB, or ERROR bits are set and the IE bit is set. The interrupt is negated when all of these status bits are cleared.

The ERROR bit is set if the DMA controller is incorrectly programmed and points to reserved memory space. No error is generated for references to nonexistent external memory. If enabled, ERROR generates a channel interrupt.

The DONEA bit is a status bit set by the DMA controller to indicate that the transfer to or from buffer A has completed. If enabled, DONEA causes a channel interrupt.

The STRTA bit is written by the user to start the channel transfer to or from buffer A. When DONEA is set, STRTA is cleared. The immediate action resulting from setting STRTA is dependent on the state of the BIU bit.

The DONEB bit is a status bit set by the DMA controller to indicate that the transfer to or from buffer B has completed. If enabled, DONEB will cause a channel interrupt.

The STRTB bit is written by the user to start the channel transfer to or from buffer B. When DONEB is set, STRTB is cleared. The immediate action resulting from setting STRTB is dependent on the state of the BIU bit.

The BIU bit indicates the current buffer-in-use (A or B). If BIU is a zero, buffer A is in use. If BIU is a one, buffer B is in use.The setting of DONEA or DONEB toggles the BIU bit. This bit is never cleared except on reset (either hardware, software, or sleep). For this reason, the processor must interrogate this bit before programming the channel for a new transfer. If both STRTA and STRTB are set at the same time, the first buffer serviced depends on the state of BIU.

11.6.1.3 DMA Buffer A Start Address Register (DBSAn)

The DBSAn is a 32-bit read/write register that contains the starting memory address for buffer A. This register may be written only when STRTA is zero.

11.6.1.4 DMA Buffer A Transfer Count Register (DBTAn)

The DBTAn is a 32-bit read/write register that contains the current transfer count in bytes for buffer A. This register may be written only when the STRTA bit for this channel is a zero. The following figure shows the format of this register; question marks indicate that the values are unknown at reset.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Read	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Reserved			TCA 12	TCA 11	TCA 10	TCA 9	TCA 8	TCA 7	TCA 6	TCA 5	TCA 4	TCA 3	TCA 2	TCA 1	TCA 0
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

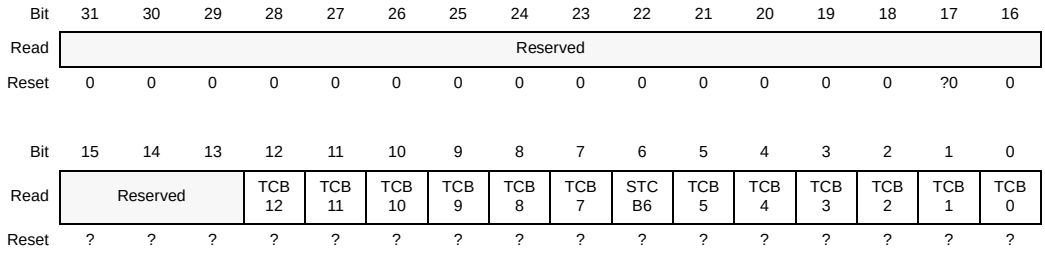
Bit	Name	Description
0..12	TCA<12:0>	Transfer count (buffer A). This field is a 13-bit value and contains the current transfer count (in bytes) for the transfer to or from buffer A. The maximum value programmed via this transfer count is 8 Kbyte.
13..31	—	Reserved. These bits are reserved and read as zeros. Writes to this field have no effect.

11.6.1.5 DMA Buffer B Start Address Register (DBSBn)

The DBSBn is a 32-bit read/write register that contains the starting memory address for buffer B. This register may be written only while STRTB in the DCSR is zero.

11.6.1.6 DMA Buffer B Transfer Count Register (DBTBn)

The DBTBn is a 32-bit read/write register that contains the current transfer count in bytes for buffer B. This register may be written only when the STRTB bit for this channel is a zero. The following figure shows the format of this register; question marks indicate that the values are unknown at reset.



Bit	Name	Description
0..12	TCB<12:0>	Transfer count (buffer B). This field is a 13-bit value and contains the current transfer count (in bytes) for the transfer to or from buffer B. The maximum value programmed via this transfer count is 8 Kbyte.
13..31	—	Reserved. These bits are reserved and read as zeros. Writes to this field have no effect.

11.6.2 DMA Operation

The DMA controller provides dynamic context switching between active channels on a demand basis. A context switch may occur when a channel completes a command or when a particular burst (portion of a transfer) has been completed. For example, if the FIFO in a particular transmit serial controller is full and cannot accept more data, that channel may be switched out of the active context in favor of another channel that is requesting service. An active channel may actually go idle many times as the device is serviced. Channels are serviced in a fixed priority with channel 0 being the highest and channel 5 being the lowest.

11.6.3 DMA Register List

The following table lists the registers contained within the DMA controller:

Physical Address	Register Name	Symbol
Channel 0 Registers		
0h B000 0000	DMA device address register.	DDAR0
0h B000 0004	DMA control/status register 0. Write ones to set.	DCSR0
0h B000 0008	Write ones to clear.	
0h B000 000C	Read only.	
0h B000 0010	DMA buffer A start address 0.	DBSA0
0h B000 0014	DMA buffer A transfer count 0.	DBTA0
0h B000 0018	DMA buffer B start address 0.	DBSB0
0h B000 001C	DMA buffer B transfer count 0.	DBTB0
Channel 1 Registers		
0h B000 0020	DMA device address register 1.	DDAR1
0h B000 0024	DMA control/status register 1. Write ones to set.	DCSR1
0h B000 0028	Write ones to clear.	
0h B000 002C	Read only.	
0h B000 0030	DMA buffer A start address 1.	DBSA1
0h B000 0034	DMA buffer A transfer count 1.	DBTA1
0h B000 0038	DMA buffer B start address 1.	DBSB1
0h B000 003C	DMA buffer B transfer count 1.	DBTB1
Channel 2 Registers		
0h B000 0040	DMA device address register 2	DDAR2
0h B000 0044	DMA control/status register 2. Write ones to set.	DCSR2
0h B000 0048	Write ones to clear.	
0h B000 004C	Read only.	
0h B000 0050	DMA buffer A start address 2.	DBSA2
0h B000 0054	DMA buffer A transfer count 2.	DBTA2
0h B000 0058	DMA buffer B start address 2.	DBSB2
0h B000 005C	DMA buffer B transfer count 2.	DBTB2
Channel 3 Registers		
0h B000 0060	DMA device address register 3.	DDAR3
0h B000 0064	DMA control/status register 3. Write ones to set.	DCSR3
0h B000 0068	Write ones to clear.	
0h B000 006C	Read only.	

Physical Address	Register Name	Symbol
0h B000 0070	DMA buffer A start address 3.	DBSA3
0h B000 0074	DMA buffer A transfer count 3.	DBTA3
0h B000 0078	DMA buffer B start address 3.	DBSB3
0h B000 007C	DMA buffer B transfer count 3.	DBTB3
Channel 4 Registers		
0h B000 0080	DMA device address register 4.	DDAR4
0h B000 0084	DMA control/status register 4. Write ones to set.	DCSR4
0h B000 0088	Write ones to clear.	
0h B000 008C	Read only.	
0h B000 0090	DMA buffer A start address 4.	DBSA4
0h B000 0094	DMA buffer A transfer count 4.	DBTA4
0h B000 0098	DMA buffer B start address 4.	DBSB4
0h B000 009C	DMA buffer B transfer count 4.	DBTB4
Channel 5 Registers		
0h B000 00A0	DMA device address register 5.	DDAR5
0h B000 00A4	DMA control/status register 5. Write ones to set.	DCSR5
0h B000 00A8	Write ones to clear.	
0h B000 00AC	Read only.	
0h B000 00B0	DMA buffer A start address 5.	DBSA5
0h B000 00B4	DMA buffer A transfer count 5.	DBTA5
0h B000 00B8	DMA buffer B start address 5.	DBSB5
0h B000 00BC	DMA buffer B transfer count 5.	DBTB5

11.7 LCD Controller

The SA-1100's LCD controller has three types of displays:

Passive Color Mode — Supports a total of 3375 possible colors, allowing any 256 colors to be displayed each frame.

Active Color Mode — Supports up to 65536 colors (16-bit).

Passive Monochrome Mode — Supports 15 gray-scale levels.

Display sizes up to 1024 x 1024 pixels are supported. However, the size of encoded pixel data within the frame buffer limits the maximum size screen the LCD can drive due to memory bus bandwidth. The LCD controller also supports single- or dual-panel displays. Encoded pixel data is stored in external memory in a frame buffer in 4-, 8-, 12-, or 16-bit increments and is loaded into a 5-entry FIFO (32 bits per entry) on a demand basis using the LCD's own dedicated dual-channel DMA controller. One channel is used for single-panel displays and two are used for dual-panel displays.

Frame buffer data contains encoded pixel values that are used by the LCD controller as pointers to index into a 256-entry x 12-bit wide palette. Monochrome palette entries are 4 bits wide; color palette entries are 12 bits wide. Encoded pixel data from the frame buffer, which is 4 bits wide, addresses the top 16 locations of the palette; 8-bit pixel data accesses any of the 256 entries within the palette. When passive color 12-bit pixel mode is enabled, the color pixel values bypass the palette and are fed directly to the LCD's dither logic. When active color 16-bit pixel mode is enabled, the pixel value not only bypasses the palette, but also bypasses the dither logic and is sent directly to the LCD's data pins.

Once the 4- or 8-bit encoded pixel value is used to select a palette entry, the value programmed within the entry is transferred to the dither logic, which uses a patented space- and time-based dithering algorithm to produce the pixel data that is output to the screen. Dithering causes individual pixels to be turned off on each frame at varying rates to produce the 15 levels of gray for monochrome screens and 15 levels each for the red, green, and blue pixel components for color screens, providing a total of 3375 colors (256 colors are available on each frame). The data output from the dither logic is placed in a 19-entry pin data FIFO before it is placed out on the LCD's pins and driven to the display using pixel clock.

Depending on the type of panel used, the LCD controller is programmed to use either 4-, 8-, or 16-pixel data output pins. Single-panel monochrome displays use either four or eight data pins to output 4 or 8 pixels for each pixel clock; single-panel color displays use eight pins to output 2-2/3 pixels each pixel clock (8 pins / 3 colors/pixel = 2-2/3 pixels per clock). The LCD controller also supports dual-panel mode, which causes the LCD controller's data lines to be split into two groups, one to drive the top half and one to drive the bottom half of the screen. For dual-panel displays, the number of pixel data output pins is doubled, allowing twice as many pixels to be output each pixel clock to the two halves of the screen.

In active color display mode, the LCD controller can drive TFT displays. The LCD's line clock pin functions as a horizontal sync (HSYNC) signal, the frame clock pin functions as a vertical sync (VSYNC) signal, and the ac bias pin functions as an output enable (OE) signal. In TFT mode, the LCD's dither logic is bypassed, sending selected palette entries (12 bits each) directly to the LCD's data output pins. Additionally, 16-bit pixels can be used that bypass both the palette and the dither logic.

The LCD controller can be configured in active color display mode and used with an external DAC (and optionally an external palette) to drive a video monitor. Note that only monitors that implement the RGB data format can be used; the LCD controller does not support the NTSC standard.

When the LCD controller is disabled, control of its pins is given to the peripheral pin controller (PPC) to be used as general-purpose digital input/output pins that are noninterruptible. The LCD controller's pins include:

- **LDD<7:0>**
  Data lines used to transmit either four or eight data values at a time to the LCD display. For monochrome displays, each pin value represents a pixel; for passive color, groupings of three pin values represent one pixel (red, green, and blue data values). In single-panel monochrome mode, LDD<3:0> pins are used. For double-pixel data, single-panel monochrome, dual-panel monochrome, single-panel color, and active color modes LDD<7:0> are used.
- **GPIO<9:2>**
  When dual-panel color or 16-bit TFT operation is programmed, GPIO pins are used as the additional, required LCD data lines to output pixel data to the screen.
- **L_PCLK**
  Pixel clock used by the LCD display to clock the pixel data into the line shift register. In passive mode, pixel clock transitions only when valid data is available on the data pins. In active mode, pixel clock transitions continuously and the ac bias pin is used as an output to signal when data is available on the LCD's data pins.
- **L_LCLK**
  Line clock used by the LCD display to signal the end of a line of pixels that transfers the line data from the shift register to the screen and increment the line pointers. Also, it is used by TFT displays as the horizontal synchronization signal.
- **L_FCLK**
  Frame clock used by the LCD displays to signal the start of a new frame of pixels that resets the line pointers to the top of the screen. Also, it is used by TFT displays as the vertical synchronization signal.
- **L_BIAS**
  AC bias used to signal the LCD display to switch the polarity of the power supplies to the row and column axis of the screen to counteract DC offset. In TFT mode, it is used as the output enable to signal when data should be latched from the data pins using the pixel clock.

The pixel clock frequency is derived from the output of the on-chip PLL that is used to clock the CPU (CCLK) and is programmable from CCLK/6 to CCLK/514. Each time new data is supplied to the LCD data pins, the pixel clock is toggled to latch the data into the LCD display's serial shifter. The line clock toggles after all pixels in a line have been transmitted to the LCD driver and a programmable number of pixel clock wait states have elapsed both at the beginning and end of each line. In passive mode, the frame clock is asserted during the first line of the screen. In active mode, the frame clock is asserted at the beginning of each frame after a programmable number of line clock wait states occur. In passive display mode, the pixel clock does not transition when the line clock is asserted. However, in active display mode, the pixel clock transitions continuously and the ac bias bin is used as an output enable to signal when valid pixels are present on the LCD's data lines. In passive mode, the ac bias pin can be configured to transition each time a programmable number of line clocks have elapsed to signal the display to reverse the polarity of its voltage to counteract DC offset in the screen.

11.7.1 LCD Controller Operation

The LCD controller supports a variety of user-programmable options including display type and size, frame buffer, encoded pixel size, and output data width. Although all programmable combinations are possible, the selection of displays available within the market dictate which combinations of these programmable options are practical. The type of external memory system implemented by the user limits the bandwidth of the LCD's DMA controller, which, in turn, limits the size and type of screen that can be controlled. The user must also determine the maximum bandwidth of the SA-1100's external bus that the LCD is allowed to use without negatively affecting all other functions that the SA-1100 must perform. Note that the LCD's DMA engine has the highest priority on the SA-1100's internal data bus structure (ARM system bus) and can "starve" other masters on the bus, including the CPU.

The following sections describe individual functional blocks within the LCD controller, frame buffer and palette memory organization, and the LCD's DMA controller. The sections are arranged in order of data flow, starting with the off-chip frame buffer and ending with the pins that interface to the LCD display.

11.7.1.1 DMA to Memory Interface

Palette RAM and encoded pixel data are stored in off-chip memory (usually DRAM) in the frame buffer and are transferred to the LCD controller's 5-entry x 32-bit wide input FIFO, on a demand basis, using the LCD controller's dedicated DMA controller. The LCD controller is on the ARM system bus (ASB) rather than the ARM peripheral bus (APB), where all other peripherals are located, because it is a higher speed synchronous bus that is able to maintain the data rate required for demanding displays, such as dual-panel color. The LCD's DMA contains two channels that transfer data from external memory to the input FIFO. One channel is used for single-panel displays and two are used for dual-panel displays.

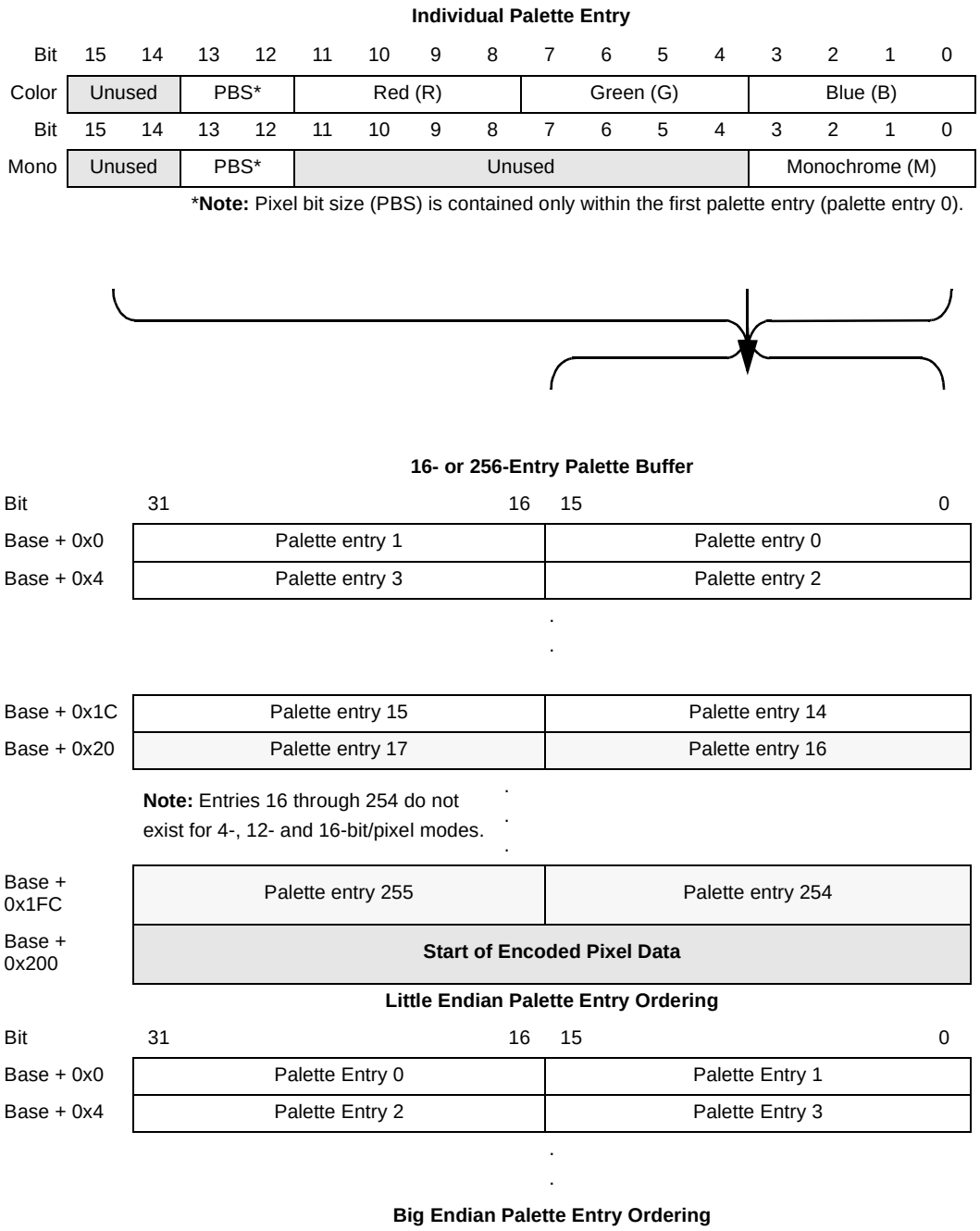
The LCD controller issues a service request to the DMA after it has been initialized and enabled. The DMA automatically performs four word transfers, filling all but one entry of the FIFO. Values are fetched from the bottom of the FIFO, one entry at a time, and each 32-bit value is unpacked into individual pixel encodings, of 4, 8, 12, or 16 bits each. After the value is removed from the bottom of the FIFO, the entry is invalidated and all data in the FIFO is transferred down one entry. When four of the five entries are empty, a service request is issued to the DMA. If the DMA is not able to keep the FIFO filled with enough pixel data due to insufficient external memory access speed and the FIFO is emptied, the FIFO underrun status bit is set and an interrupt request is made.

11.7.1.2 Frame Buffer

The frame buffer is in an off-chip memory area used to supply enough encoded pixel values to fill the entire screen one or more times. At the start or lowest order address of the LCD controller's frame buffer is either a 32- or 512-byte buffer used to store the lookup palette data for each frame. A 32-byte buffer is used to load the top 16 entries of the palette for 4-, 12-, or 16-bit pixel encodings, and a 512-byte buffer is used to load the entire 256-entry palette for 8-bit pixel encodings. Note that the LCD's on-chip palette is not used for 12- and 16-bit pixel encodings; the PBS field must be programmed to select 12- and 16-bit pixel mode and the remainder of the 32 bytes at the top of the frame buffer must be zero-filled even though the data is not used.

Each time a new frame is fetched from the frame buffer, the LCD controller's palette is first loaded with the data contained within the palette buffer. Each of the 16 or 256 palette entries is stored in adjacent half-words. Figure 11-3 shows the palette-entry organization for little and big endian memory organization. The user can select how the LCD views the ordering of frame buffer palette/pixel entries by programming the big/little endian select (BLE) bit in LCD control register 0. In little endian mode, palette entries are ordered starting with the least significant half-word, followed by the most significant. In big endian mode, palette entries are ordered starting with the most significant half-word, followed by the least significant. Note that the ordering of the 4-bit R, G, B, and monochrome pixel data (and the PBS field) does not change between big and little endian modes; only the relative positioning of the individual 16-bit palette entries changes.

Figure 11-3. Palette Buffer Format



The first palette entry (palette entry 0) also contains an extra field that is used to synchronously configure the LCD controller at the beginning of each frame. Bits 12 and 13 of the first palette entry contain a field that is used to select the number of bits per pixel that is to be used in the next frame (see Figure 11-3). The pixel bit size (PBS) bit-field is decoded by the LCD to correctly unpack pixel data into nibbles, bytes, 12-bit values, or half-words, and by the palette to tell it how many address bits are contained in the pixel data it is supplied, configuring the palette size to 16 or 256 entries. Note that 12/16-bit pixel mode bypasses the LCD palette and supplies 12-bit values directly to the dither logic when passive mode is enabled, or 16-bit values directly to the output FIFOs when active mode is enabled. The following table shows the encoding of the PBS bit field.

Bit	Name	Description
13..12	PBS	Pixel bit size. 0x – 4 bits per pixel, 16-entry palette, 32 bytes of palette buffer transferred each frame to palette. 01 – 8 bits per pixel, 256-entry palette, 512 bytes of palette buffer transferred each frame to palette. 10 – 12 bits per pixel in passive mode (PAS=0), 16 bits per pixel in active mode (PAS=1). Palette unused, however, 32 bytes of “dummy” palette data is transferred each frame to palette. Palette data must be zero-filled. 11 – Reserved. **Note**: Two 4-bit pixels are packed into each byte, and 12-bit pixels are right justified on half-word boundaries.

Following the palette buffer is the pixel data buffer that contains one encoded pixel value for each of the pixels present on the display. The number of pixel data values depends on the size of the screen (1024 x 768 = 786,432 encoded pixel values). Figure 11-4 through Figure 11-7 show the memory organization within the frame buffer for each size pixel encoding. Note that for 4-bit encodings, 2 pixels are placed into each byte, and for 12-bit encodings the value is right- justified within a half-word. These figures show the encoded pixel organization for little endian memory organization. The user can select how the LCD views the ordering of frame buffer pixel entries by programming the big/little endian select (BLE) bit in LCD control register 0. In big endian mode, pixel entries are ordered starting with the most significant nibble, byte, or half-word and ending with the least significant.

Figure 11-4. 4 Bits Per Pixel Data Memory Organization (Little Endian)

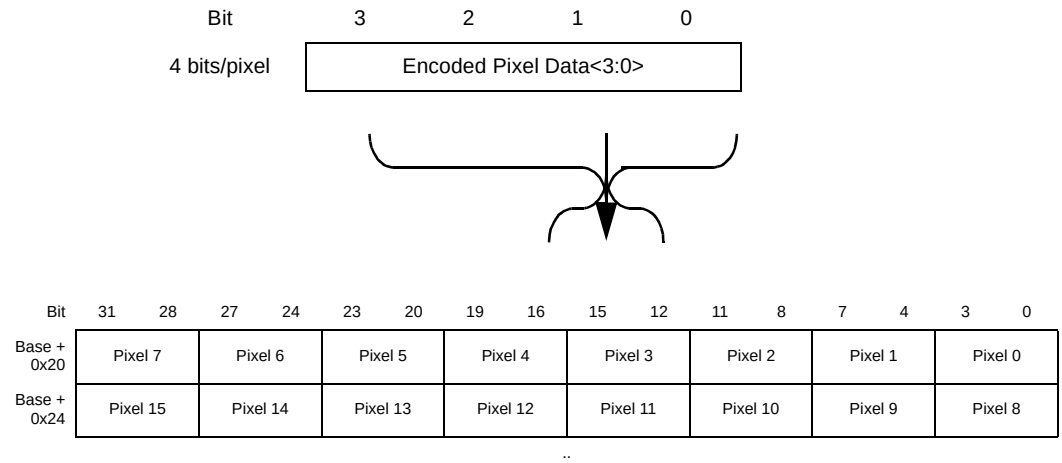


Figure 11-5. 8-Bits Per Pixel Data Memory Organization (Little Endian)

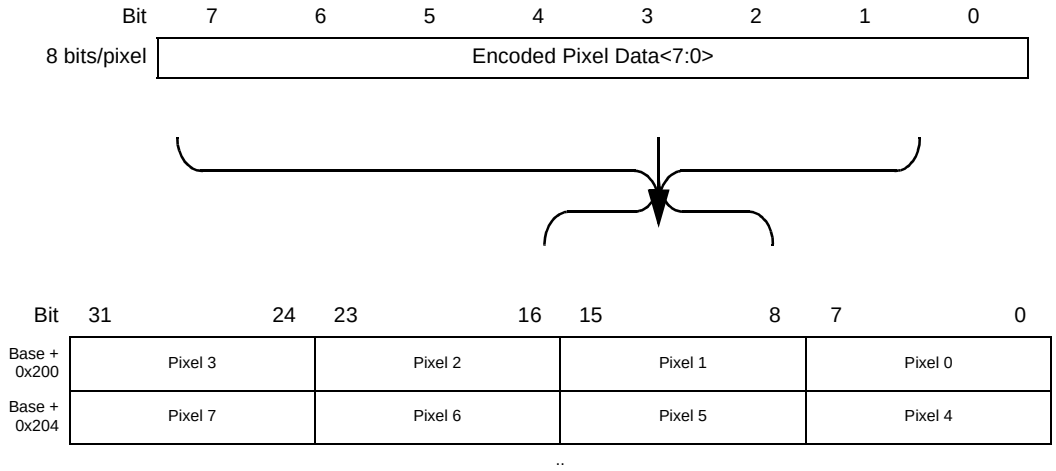


Figure 11-6. 12-Bits Per Pixel Data Memory Organization (Passive Mode Only)

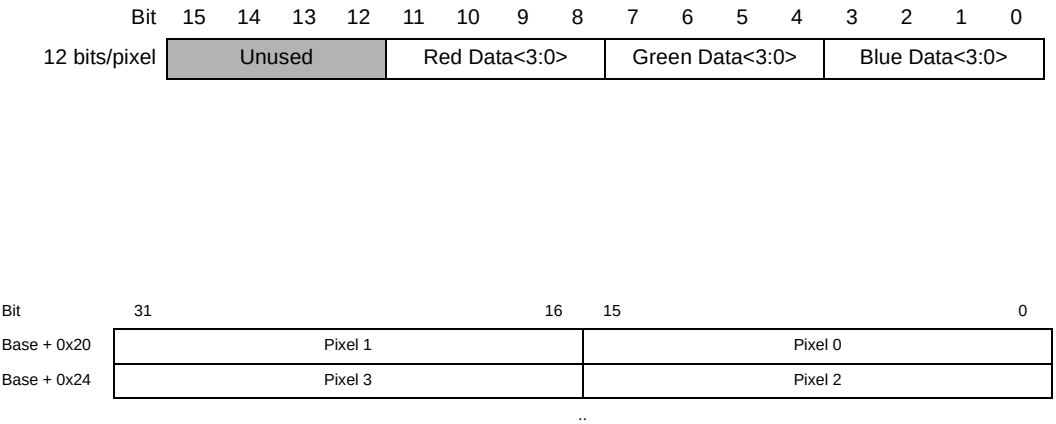
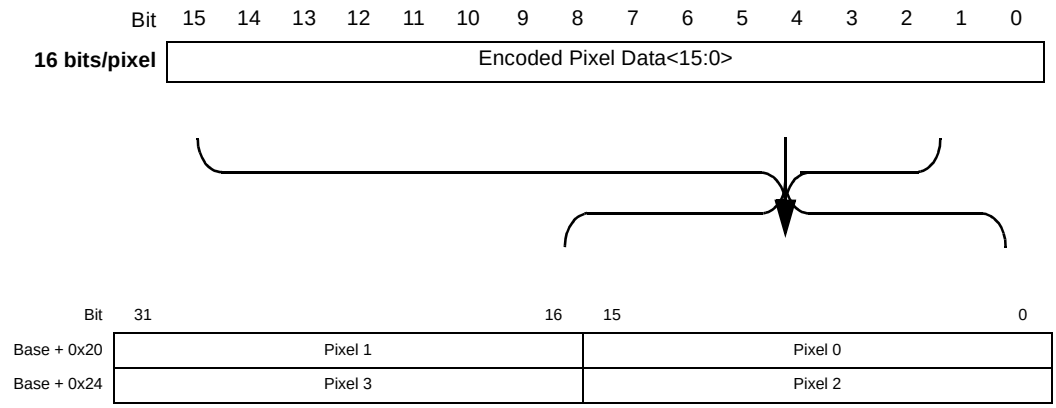


Figure 11-7. 16-Bits Per Pixel Data Memory Organization (Active Mode Only)



In dual-panel mode, pixels are presented to two halves of the screen at the same time (upper and lower). A second DMA channel and input FIFO exist to support dual-panel operation. The DMA channels alternate service requests when filling the two input FIFOs. The palette buffer is implemented in DMA channel 1, but not channel 2; the base address points to the top of the encoded pixel values for channel 2. The DMA controller contains a base and current address pointer register. The end address is calculated automatically by the LCD using the display information such as pixels per line, lines per frame, single- or dual-panel mode, color or monochrome mode, and bits per pixel, which are programmed by the user.

The base address of both DMA channels must be configured such that the least significant four address bits are all zero (for example, address bits 3 through 0 must be zero). This requirement limits the base address of the frame buffer to start at even 4-word (or 16-byte) intervals.

The frame buffer must contain an even multiple of 16 pixels for every line and must be aligned on a quadword boundary. Many LCD displays are a multiple of 16 pixels wide; however, most passive LCD displays are not and will ignore extra pixels at the end of each line. Thus for these types of displays that do not use an even multiple of 16 encoded pixel values, the user must adjust the start address for each line by adding between 1 and 15 “dummy” pixel values to the end of the previous line. For example, if the screen that is being driven is 107 pixels wide, and 4-bits/pixel mode is used, each line is 107 pixels or nibbles in length (53.5 bytes). The next nearest 16-pixel boundary occurs at 112 pixels or nibbles (56 bytes). Thus, the user must start each new line in the frame buffer at multiples of 56 bytes by adding an extra 5 “dummy” pixels per line (2.5 bytes). The user must ensure that the panel being controlled does indeed ignore extra pixel clocks at the end of each line when a panel with line widths that are non-multiple of 16 pixels are used.

The user must add extra space at the end of the frame buffer. The LCD’s DMA may overshoot the end of the frame buffer by one burst cycle (4-word read). The LCD’s DMA reads these extra values, but they are flushed from the input FIFO each time the frame clock is pulsed. The user must ensure that the four words immediately following the end of the frame buffer reside in legal memory space (do not cause a bus error if read). Since the LCD does not alter this memory (only reads are performed), these locations can be used for data storage unrelated to the LCD.

The following equations are used to calculate the total frame buffer size in bytes that is accessed by the DMA based on varying pixel size encodings and screen sizes. The first term in the equations represents the size of the palette buffer, the second term is the add-on for the DMA overshoot at the end of the frame buffer, and the third term is the size required for the encoded pixel values. Note that for dual-panel mode, the frame buffer size is equally distributed between the two DMA channels and that DMA channel 2’s buffer is either 32 or 512 bytes smaller (no palette buffer; that is, the first term in the equations is deleted).

4 bits/pixel: $$FrameBufferSize = 32 + 16 + \left(\frac{Line(sXColumns)}{2}\right) + (2(nXLines))$$

8 bits/pixel: $$FrameBufferSize = 512 + 16 + (Line(sXColumns)) + (nXLines)$$

12 or 16 bits/pixel: $$FrameBufferSize = 32 + 16 + 2(Line(sXColumns))$$

Where n = 0 to 15 and is the number of extra “dummy” pixels required per line to make pixels/line an even multiple of sixteen.

Note: The base address of the frame buffer must start on even 4-word boundaries (the four least significant address bits <3:0> must be zero).

11.7.1.3 Input FIFO

Data from the LCD's DMA is directed either to the palette or the input FIFO. The direction of data flow is switched whenever the LCD controller is first enabled and by each frame pulse. After the LCD controller is configured and enabled, the first 32 (4-, 12-, and 16-bits/pixel) or 512 (8-bit/pixel) bytes supplied by the DMA are sent to the palette. All subsequent encoded pixel data is sent to the FIFO. After an entire frame of pixels has been processed, the frame clock pin is pulsed to denote the start of the next frame. This signal is also used to change the direction of DMA input data from the FIFO back to the palette. A modulus of 8 (4-, 12-, and 16-bits/pixel) or 128 (8-bits/pixel) is used to count when loading the palette RAM, depending on the pixel bit size shown above. A 7-bit counter is loaded each time a frame clock pulse occurs or the LCD is enabled, and is decremented each time a word is stored to the palette (two palette entries). When the counter wraps around to zero, the data input from the DMA is switched back to the FIFO.

The LCD controller contains a 5-entry x 32-bit wide input FIFO that is used to store encoded pixels fetched from the frame buffer. The FIFO signals a service request to the DMA whenever four entries of the FIFO are empty. In turn, the DMA automatically fills the FIFO with a 4-word burst.

Pixel data from the frame buffer remains packed within individual 32-bit words when it is loaded into the FIFO. The LCD controller's port size is 32 bits wide to accommodate the heavy data flow from the frame buffer. Depending on the number of bits per pixel, as words are taken from the bottom of the FIFO, they are unpacked and supplied to the lookup palette in nibbles (4 bits/pixel) or bytes (8 bits/pixel) to the dither logic (12 bits/pixel), or directly to the pins in half-word increments (16 bits/pixel).

Each time a word is taken from the bottom of the FIFO, the entry is invalidated and all data in the FIFO moves down one position. When four entries are empty, a service request is issued to the DMA.

11.7.1.4 Lookup Palette

The encoded pixel data taken from the bottom entry of the input FIFO is used as an address to index and select individual palette locations. Four-bit pixel encodings address 16 locations and 8-bit pixel encodings select any of the 256 palette entries. Note that the user may program 1, 2, and 3 bits/pixel as well by zeroing out the upper 3, 2 or 1 bits of each encoded pixel value in the frame buffer, respectively. However, for 1, 2, and 3 bits/pixel, the encoded pixel size remains at 4 bits within the frame buffer and within the LCD controller's input FIFO.

Once a palette entry is selected by the encoded pixel value, the contents of the entry is sent to the color/gray-scale space/time base dither circuit. In color mode, the value within the palette is made up of three 4-bit fields, one for each color component – red, green, and blue. In monochrome mode, only one 4-bit value is present (see Figure 11-3). For both modes, the 4-bit values represent one of 15 intensity levels. For color operation, an individual frame is limited to a selection of 256 colors (the number of palette entries). However, the LCD controller is capable of generating a total of 3375 colors (15 levels per color ^ 3 colors = 3375). When 12 or 16 bits per pixel mode is enabled, the palette is bypassed. For passive displays, 12-bit pixels are sent directly to the dither logic; for active displays, 16-bit pixels are sent to the output FIFO to be driven directly to the LCD's data pins.

11.7.1.5 Color/Gray-Scale Dithering

For passive displays, entries selected from the lookup palette are sent to the color/gray-scale space/time base dither generator. Each 4-bit value is used to select one of 15 intensity levels. Note that two of the 16 dither values are identical (always high). The color/gray intensity is controlled by turning individual pixels on and off at varying periodic rates. For some screens, more intense colors/grays are produced by making the average time the pixel is high longer than the average time it is low, while other screens produce more intense colors/grays when the average time the pixel is low is longer. The user should program the palette appropriately depending on whether a one on the pixel line turns the pixel on or off. The dither generator also uses the intensity of adjacent pixels in its calculations to give the screen image a smooth appearance. The proprietary dither algorithm is optimized to provide a range of intensity values that match the eye's visual perception of color/gray gradations. In color mode, three separate dither blocks are used to process the three color components: red, green, and blue. Table 11-7 summarizes the duty cycle and resultant intensity level for all 15 color/gray-scale levels.

Table 11-7. Color/Gray-Scale Intensities and Modulation Rates

Dither Value (4-Bit Value from Palette)	Intensity (0% Is Black)	Modulation Rate (Ratio of ON to ON+OFF Pixels)
0000	0.0%	0
0001	11.1%	1/9
0010	20.0%	1/5
0011	26.7%	4/15
0100	33.3%	3/9
0101	40.0%	2/5
0110	44.4%	4/9
0111	50.0%	1/2
1000	55.6%	5/9
1001	60.0%	3/5
1010	66.6%	6/9
1011	73.3%	11/15
1100	80.0%	4/5
1101	88.9%	8/9
1110	100.0%	1
1111	100.0%	1

11.7.1.6 Output FIFO

The LCD controller contains a 19-entry x 16-bit wide output FIFO that is used to store pixel pin data before it is driven out to the pins. Each time a modulated pixel value is output from the dither generator, it is placed into a serial shifter. The size of the shifter is controlled by programming the color/monochrome select and single- and dual-panel, double pixel data, and passive/active select bits in the LCD's control registers and the pixel bit size within palette entry 0 in the frame buffer. The shifter can be configured to be 4, 8, or 16 bits wide. Four pins are used for single-panel monochrome screens; 8 pins are used for single- and dual-panel monochrome screens as well as single-panel color displays; 12 pins are used for active displays; and 16 pins are used for dual-panel color and active displays. Once the correct number of pixels have been placed within the shifter (4-, 8-, or 16-pixel values), the value is transferred to the top of the output FIFO. The value is then transferred down until it reaches the last empty location within the FIFO. Each time a value is taken from the bottom of the FIFO, the entry is invalidated and all data in the FIFO moves down one position.

11.7.1.7 LCD Controller Pins

Pixel data is removed from the bottom of the output FIFO and is driven in parallel onto the LCD's data lines on the edge selected by the pixel clock polarity (PCP) bit. For a 4-bit wide bus, data is driven onto the LCD data lines LDD<3:0> starting with the most significant bit. For an 8-bit wide bus, data is driven onto LDD<7:0>; for a 12-bit bus GPIO<5:2> and LDD<7:0>; and for a 16-bit bus GPIO<9:2> and LDD<7:0>. In monochrome dual-panel mode, the pixels for the upper half of the screen are driven onto LDD<3:0> and the lower half to LDD<7:4>. In color dual-panel mode, the upper panel pixels are driven onto LDD<7:0> and the lower panel pixels to GPIO<9:2>. Note that for a 4-bit wide bus, data is output via the LDD<3:0> pins and the LCD<7:4> pins are held low by the LCD controller. The user cannot use this pins as GPIOs in this mode. However, for a 12-bit wide bus, the user is free to use GPIO<9:6> as general- purpose I/O signals.

When an entire line of pixels has been output to the LCD controller screen, the line clock pin (L_LCLK) is toggled. Likewise, when an entire frame of pixels has been output to the LCD controller screen, the frame clock pin (L_FCLK) is toggled. To prevent a dc charge from building within a passive display, its power and ground supplies must be switched periodically. The LCD controller signals the display to switch the polarity by toggling the ac bias pin (L_BIAS). The user can control the frequency of the bias pin by programming the number of line clock transitions between each toggle.

When active display mode is enabled, the timing of the pixel, line, and frame clocks and the ac bias pin changes. The pixel clock transitions continuously in this mode as long as the LCD is enabled. The ac bias pin functions as an output enable. When it is asserted, the display latches data from the LCD's pins using the pixel clock. The line clock pin is used as the horizontal synchronization signal (HSYNC) and the frame clock as the vertical synchronization signal (VSYNC). The timing of the line and frame clock pins is programmable to support both passive and active mode. Programming options include: waitstate insertion both at the beginning and end of each line and frame; pixel clock; line clock; frame clock; output enable signal polarity; and frame clock pulse width.

When the LCD controller is disabled, control of all 12 of its pins is relinquished to the peripheral pin controller (PPC) unit to be used as general-purpose digital I/O pins that are noninterruptible. See the section 11.13 on page 184 for a description of the programming and operation of the PPC unit.

11.7.2 LCD Controller Register Definitions

The LCD controller contains four control registers, four DMA address registers, and one status register. The control registers contain bit fields to enable and disable the LCD controller; to define the height and width of the screen being controlled; and to indicate single- versus dual-panel display mode, color versus monochrome mode, passive versus active display, polarity of the control pins, pulse width of the line and frame clocks, pixel clock and ac bias pin frequency. AC bias pin toggles per interrupt the number of waitstates to insert before and after each line, after each frame, and various interrupt masks. An additional control field exists to tune the DMA's performance based on the type of memory system in which the SA-1100 is used. This field controls the placement of a minimum delay between each LCD DMA request to ensure enough bus bandwidth is given to other ARM system bus masters for accesses.

The DMA address registers are used to define the base addresses of the off-chip frame buffers and to which address the DMA is currently pointing. Both of these registers exist for DMA channels 1 and 2.

The status registers contain bits that signal input and output FIFO overrun and underrun errors, DMA bus errors, when the DMA base address can be reprogrammed, when the last active frame has completed after the LCD is disabled, and each time the ac bias pin has toggled a programmed number of times. Each of these hardware-detected events signals an interrupt request to the interrupt controller.

11.7.3 LCD Controller Control Register 0

LCD controller control register 0 (LCCR0) contains 10 bit fields that are used to control various functions within the LCD controller.

11.7.3.1 LCD Enable (LEN)

The LCD enable (LEN) bit is used to enable and disable all LCD controller operation. When LEN=0, the LCD controller is disabled and control of all 12 of its pins is given to the peripheral pin controller (PPC) unit to be used as general-purpose I/O (noninterruptible). When LEN=1, the LCD controller is enabled. Note that all other control registers should be initialized before setting LEN. The user can program LCCR0 last, and configure all 10 bit fields at the same time via a word write to the register. If the user clears LEN while the LCD controller is enabled, it will complete transmission of the current frame before being disabled. Completion of the current frame is signalled by the LCD when it sets the LCD disable done flag (LDD) within the LCD status register that generates an interrupt request. The user should use a read-modify-write procedure to clear LEN because the other bit-fields within LCCR0 continue to be used by the LCD controller after LEN is cleared until the frame that is currently in progress completes. When the LCD controller is disabled, control of all 12 of its pins is given to the peripheral pin controller (PPC) so that they may be used for general-purpose input and output (noninterruptible). See the Section 11.13, “Peripheral Pin Controller (PPC)” on page 11-184 for a description of the PPC.

11.7.3.2 Color/Monochrome Select (CMS)

The color/monochrome select (CMS) bit selects whether the LCD controller operates in color or monochrome mode. When CMS=0, color mode is selected, palette entries are 12 bits wide (4 bits per color), 8 data pins are enabled for single-panel mode, 16 data pins are enabled for dual-panel mode (GPIO pins 2..9 are used as the extra 8 data output pins), and all three dither blocks are used, one each for the red, green, and blue pixel components. When CMS=1, monochrome mode is selected, palette entries are 4 bits wide (15 levels of gray-scale), 4 or 8 data pins are enabled for single-panel mode, and 8 data pins are enabled for dual-panel mode.

11.7.3.3 Single-/Dual-Panel Select (SDS)

In passive mode (PAS=0), the single-/dual-panel select (SDS) bit is used to select the type of display control that is implemented by the LCD screen. When SDS=0, single-panel operation is selected (pixels presented to screen a line at a time), and when SDS=1, dual-panel operation is selected (pixels presented to screen two lines at a time). Single-panel LCD drivers have one line/row shifter and driver for pixels, and one line pointer; dual-panel LCD controller drivers have two line/row shifters (one for the top half of the screen, one for the bottom), and two line pointers (one for the top half of the screen, one for the bottom). When dual-panel mode is programmed, both of the LCD controller’s DMA channels are used. DMA channel 1 is used to load the palette RAM from the frame buffer and to drive the upper half of the display, and DMA channel 2 drives the lower half. The two channels alternate when fetching data for both halves of the screen, placing encoded pixel values within the two separate input FIFOs. When programming dual-panel operation, the user must perform the following sequence in order: disable the LCD (LEN=0), program dual-panel mode (SDS=0->1), write the upper panel DMA base address, write the lower panel DMA base address, and enable the LCD (LEN=0->1). When dual-panel operation is enabled, the LCD controller doubles its pin uses; thus, for monochrome screens 8 pins are used, and for color screens, 16 pins are used.

Table 11-8 shows the LCD data pins and GPIO pins used for each mode of operation and the ordering of pixels delivered to a screen for each mode of operation. Figure 11-8 shows the LCD data pin pixel ordering. Note that when dual-panel color operation is enabled, the user must configure GPIO pins 2 through 9 as outputs by setting bits 2..9 within the GPIO pin direction register (GPDR) and GPIO alternate function register (GAFR). See the Section 9.1, "General-Purpose I/O" on page 9-1 for configuration information. Also note that SDS is ignored in active mode (PAS=1).

Table 11-8. LCD Controller Data Pin Utilization

Color/ Monochrome Panel	Single/ Dual Panel	Passive/ Active Panel	Screen Portion	Pins
Monochrome	Single	Passive	Whole	LDD<3:0>
Monochrome	Single	Passive	Whole	LDD<7:0>[1]
Monochrome	Dual	Passive	Top	LDD<3:0>
			Bottom	LDD<7:4>
Color	Single	Passive	Whole	LDD<7:0>
Color	Dual	Passive	Top	LDD<7:0>
			Bottom	GPIO<9:2>
Color	Single	Active	Whole	GPIO<9:2>, LDD<7:0>

[1] Double-pixel data mode (DPD) = 1.

Figure 11-8. LCD Data-Pin Pixel Ordering

Top Left Corner of Screen

	Column 0	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8
Row 0	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>
Row 1	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>
Row 2	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>
Row 3	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>

Passive Monochrome Single-Panel Display Pixel Ordering

Top Left Corner of Screen

	Column 0	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8
Row 0	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<4>	LDD<5>	LDD<6>	LDD<7>	LDD<0>
Row 1	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<4>	LDD<5>	LDD<6>	LDD<7>	LDD<0>
Row 2	LD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<4>	LDD<5>	LDD<6>	LDD<7>	LDD<0>
Row 3	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<4>	LDD<5>	LDD<6>	LDD<7>	LDD<0>

Passive Monochrome Single-Panel Double-Pixel Display Pixel Ordering

Top Left Corner of Screen

	Column 0	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8
Row 0	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>
Row 1	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>	LDD<1>	LDD<2>	LDD<3>	LDD<0>
⋮									
Row n/2	LDD<4>	LDD<5>	LDD<6>	LDD<7>	LDD<4>	LDD<5>	LDD<6>	LDD<7>	LDD<4>
Row n/2+1	LDD<4>	LDD<5>	LDD<6>	LDD<7>	LDD<4>	LDD<5>	LDD<6>	LDD<7>	LDD<4>

n = # of rows **Passive Monochrome Dual-Panel Display Pixel Ordering**

Top Left Corner of Screen

	Column 0 Red	Column 0 Green	Column 0 Blue	Column 1 Red	Column 1 Green	Column 1 Blue	Column 2 Red	Column 2 Green	Column 2 Blue
Row 0	LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>	LDD<7>
Row 1	LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>	LDD<7>
Row 2	LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>	LDD<7>
Row 3	LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>	LDD<7>

Passive Color Single-Panel Display Pixel Ordering

Top Left Corner of Screen

	Column 0 Red	Column 0 Green	Column 2 Green	Column 2 Blue	Column 4 Blue	Column 5 Red	Column 5 Green
Row 0	LDD<7>	LDD<6>	LDD<0>	LDD<7>	LDD<1>	LDD<0>	LDD<7>
Row 1	LDD<7>	LDD<6>	LDD<0>	LDD<7>	LDD<1>	LDD<0>	LDD<7>
⋮							
Row n/2	GPIO<9>	GPIO<8>	GPIO<2>	GPIO<9>	GPIO<3>	GPIO<2>	GPIO<9>
Row n/2+1	GPIO<9>	GPIO<8>	GPIO<2>	GPIO<9>	GPIO<3>	GPIO<2>	GPIO<9>

n = # of rows **Passive Color Dual-Panel Display Pixel Ordering**

11.7.3.4 LCD Disable Done Interrupt Mask (LDM)

The LCD disable done interrupt mask (LDM) bit is used to mask or enable interrupt requests that are asserted after the LCD is disabled and the frame currently being output to the pins has completed. When LDM=0, the interrupt is enabled, and whenever the LCD disable done (LDD) status bit within the LCD status register (LCSR) is set (one), an interrupt request is made to the interrupt controller. When LDM=1, the interrupt is masked and the state of the LDD status bit is ignored by the interrupt controller. Note that programming LDM=1 does not affect the current state of LDD or the LCD controller's ability to set and clear LDD; it only blocks the generation of the interrupt request. This interrupt is particularly useful when the user needs to ensure the LCD has been disabled and the current frame that is being output to the pins has completed, before entering sleep mode. If the user disables the LCD, but does not need to enter sleep mode, this interrupt can be masked using LDM.

11.7.3.5 Base Address Update Interrupt Mask (BAM)

The base address update interrupt mask (BAM) bit is used to mask or enable interrupt requests that are asserted at the beginning of each frame when the LCD's base address pointer is transferred to the current address pointer within the LCD's DMA. When BAM=0, the interrupt is enabled, and whenever the base address update (BAU) status bit within the LCD status register (LCSR) is set (one) an interrupt request is made to the interrupt controller. When BAM=1, the interrupt is masked and the state of the BAU status bit is ignored by the interrupt controller. Note that programming BAM=1 does not affect the current state of BAU or the LCD controller's ability to set and clear BAU; it only blocks the generation of the interrupt request. Note that this interrupt mask is particularly useful when the user wishes to enter idle mode to turn off the CPU and to display the same image (the off-chip frame buffer data does not change). By masking the BAU interrupt, the SA-1100 is not forced out of idle mode at the end of each frame.

11.7.3.6 Error Interrupt Mask (ERM)

The error interrupt mask (ERM) bit is used to mask or enable interrupt requests that are asserted whenever a bus error or input/output FIFO over/underrun error occurs. When ERM=0, all error interrupts are enabled, and whenever the bus error (BER) status bit or any of the input/output FIFO over/underrun (IOL, IUL, IOU, IUU, OOL, OUL, OOU, OUU) status bits within the LCD status register (LCSR) are set (one), an interrupt request is made to the interrupt controller. When ERM=1, error interrupts are masked; the state of all of the error status bits (BER, IOL, IUL, IOU, IUU, OOL, OUL, OOU, OUU) are ignored by the interrupt controller. Note that programming ERM=1 does not affect the current state of these status bits or the LCD controller's ability to set and clear them; it only blocks the generation of the interrupt requests.

11.7.3.7 Passive/Active Display Select (PAS)

The passive/active display select (PAS) bit selects whether the LCD controller operates in passive (STN) or active (TFT) display control mode. When PAS=0, passive or STN mode is selected, all LCD data flow operates normally (including the use of the LCD's dither logic), and all LCD controller pin timing operates as described in the preceding sections.

When PAS=1, active or TFT mode is selected. For 4- and 8-bit per pixel modes, pixel data is transferred via the DMA from off-chip memory to the input FIFO, is unpacked and used to select an entry from the palette, just like passive mode. However, the value read from the palette bypasses the LCD's dither logic, and is sent directly to the output FIFO to be output on the LCD's data pins. This 12-bit value output to the pins represents 4 bits of red, 4 bits of green, and 4 bits of blue data. For 12- and 16-bit pixel encoding mode, the pixel size within the frame buffer is increased to 16 bits.

Thus two 16-bit values are packed into each word in the frame buffer. Each 16-bit value is transferred via the DMA from off-chip memory to the input FIFO. Unlike 4- and 8-bit per pixel modes, the 16-bit value bypasses both the palette and the dither logic, and is placed directly in the output FIFO to be output on the LCD's data pins. Increasing the size of the pixel representation allows a total of 64K colors to be generated. This 16-bit value output to the pins can be organized into one of three RGB color formats: 6 bits of red, 5 bits of green, and 5 bits of blue data; 5 bits of red, 6 bits of green, and 5 bits of blue data; 5 bits of red, 5 bits of green, and 6 bits of blue data, as specified by the user. Note that the pin timing of the LCD changes when active mode is selected. Timing of each pin is described in subsequent bit-field sections for both passive and active mode. Additionally, the LCD controller can be configured in active color display mode and used with an external DAC and optionally an external palette to drive a video monitor. Note that only monitors that implement the RGB data format can be used; the LCD controller does not support the NTSC standard.

Figure 11-9 shows which bits within each frame buffer entry (for 16-bit/pixel mode) and which bits within a selected palette entry (for 4- and 8-bit/pixel mode) are sent to the individual LCD data pins. In active mode, GPIO pins 2..9 are also used. Note that the user must configure GPIO pins 2..5 as outputs (for 4- and 8-bit/pixel mode), and GPIO pins 2..9 as outputs (for 16-bit/pixel mode) by setting the appropriate bits within the GPIO pin direction register (GPDR) and GPIO alternate function register (GAFR). See the General-Purpose I/O section for configuration information. If GPDR<6:9> = GAFR<6:9> = 4'hF in 4- or 8-bit/pixel mode, then GPIO<6:9> are pulled low during LCD operation in active mode. However, the user is free to clear GAFR<6:9>, allowing the GPIO unit to assume control of these pins to be used as normal digital I/Os. In general, the user may clear any number of GAFR bits 2..9, to allow the GPIO unit to assume control of unused GPIO pins for normal digital I/O depending on the required number of data pins.

If the panel that is being controlled contains more data pin inputs than 16, the user may still use the SA-1100's LCD controller, but the panel will be limited to a total of 64 K colors. If the user wishes to maintain the panel's full range of colors and increase the granularity of the spectrum, the LCD's 16 data pins should be interfaced to the panel's most significant R, G, and B pixel data input pins and the least significant R, G, and B data pins should be tied either high or low. If instead, the user wishes to maintain the granularity of the spectrum and limit the overall range of colors possible, the LCD's 16 data pins should be interfaced to the panel's least significant R, G, and B pixel data input pins and the most significant data pins should again be tied either high or low.

Figure 11-9. Frame Buffer/Palette Bits Output to LCD Data Pins in Active Mode

16-Bit/Pixel Mode

Frame Buffer Entry

	R<5>	R<4>	R<3>	R<2>	R<1>	R<0>	G<4>	G<3>	G<2>	G<1>	G<0>	B<4>	B<3>	B<2>	B<1>	B<0>
	R<4>	R<3>	R<2>	R<1>	R<0>	G<5>	G<4>	G<3>	G<2>	G<1>	G<0>	B<4>	B<3>	B<2>	B<1>	B<0>
	R<4>	R<3>	R<2>	R<1>	R<0>	G<4>	G<3>	G<2>	G<1>	G<0>	B<5>	B<4>	B<3>	B<2>	B<1>	B<0>
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data Pin	GPIO <9>	GPIO <8>	GPIO <7>	GPIO <6>	GPIO <5>	GPIO <4>	GPIO <3>	GPIO <2>	LDD <7>	LDD <6>	LDD <5>	LDD <4>	LDD <3>	LDD <2>	LDD <1>	LDD <0>

4- or 8-Bit/Pixel Mode

Selected Palette Entry

					R<3>	R<2>	R<1>	R<0>	G<3>	G<2>	G<1>	G<0>	B<3>	B<2>	B<1>	B<0>
Bit	VSS[1]	VSS	VSS	VSS	11	10	9	8	7	6	5	4	3	2	1	0
Data Pin	GPIO <9>	GPIO <8>	GPIO <7>	GPIO <6>	GPIO <5>	GPIO <4>	GPIO <3>	GPIO <2>	LDD <7>	LDD <6>	LDD <5>	LDD <4>	LDD <3>	LDD <2>	LDD <1>	LDD <0>

[1] GPIO pins 6..0 are grounded by the LCD in this mode. However, if GAFR bit 6..9 are cleared within the system control module, these pins can be used as normal GPIO pins.

11.7.3.8 Big/Little Endian Select (BLE)

The big/little endian select (BLE) bit selects whether the LCD controller views external memory organization of the frame buffer as big or little endian. When BLE=0, little endian mode is selected and pixel data is organized within the off-chip frame buffer as shown in Figure 11-4 through Figure 11-7. Pixels are packed into words starting with the least-significant nibble, byte, or half-word. When BLE=1, big endian mode is selected and pixel data is organized in memory starting with the most significant nibble, byte, or half-word. When BLE=1, palette entries are packed into half-words starting with the most significant half-word. Note that BLE does not affect the ordering of the 4-bit red/green/blue bit fields, the 4-bit monochrome field within each 16-bit palette entry, or the 2-bit pixel bit size (PBS) field contained with palette entry 0.

11.7.3.9 Double-Pixel Data (DPD) Pin Mode

The double-pixel data (DPD) pin mode bit selects whether four or eight data pins are used to output pixel data to the LCD screen in single-panel monochrome mode. When DPD=0, LDD<3:0> pins are used to output 4-pixel values each pixel clock transition; when DPD=1, LDD<7:0> pins are used to output 8-pixel values each pixel clock. See the following table and figure for a comparison of how the LCD's data pins are used in each of its display modes. Note that DPD does not affect dual-panel monochrome mode nor any of the color modes.

11.7.3.10 Palette DMA Request Delay (PDD)

The 8-bit palette DMA request delay (PDD) field is used to select the minimum number of memory controller clock cycles (half the frequency of the CPU clock) to wait between the servicing of each DMA request issued while the on-chip palette is loaded. When the palette is loaded at the beginning of every frame, either 32 or 512 bytes of data must be accessed by the LCD's DMA. Since the LCD's DMA is the highest priority master on the ARM system bus, other masters (such as the CPU) will be denied access to the bus and may be starved. Using PDD allows other masters to gain access of the bus in between palette DMA loads, so that they are not locked from accessing the bus for an unacceptable period of time. Note that PDD does not apply to normal input FIFO DMA requests for frame buffer information since these DMA requests do not occur back-to-back. The input FIFO DMA request rate is a function of the rate at which pixels are displayed on the screen.

After a palette DMA burst cycle has completed, the value contained within PDD is loaded to a down counter that disables the palette from issuing another DMA request until the counter decrements to zero. This counter ensures that the LCD's DMA does not fully consume the bandwidth of the SA-1100's system bus. Once the counter reaches zero, any pending or future DMA requests by the palette cause the DMA to arbitrate for the ARM system bus (ASB). Once the DMA burst cycle has completed, the process starts over and the value in PDD is loaded to the counter to create another waitstate period, which disables the palette from issuing a DMA request. PDD can be programmed with a value that causes the FIFO to wait between 0 to 255 memory clock cycles after the completion of one DMA request to the start of the next request. When PDD=8'h00, the FIFO DMA request delay function is disabled. Note that waitstates are not inserted between DMA burst cycles that are used to fill the input FIFO with pixel data.

The following table shows the location of all 10 bit-fields located in LCD control register 0 (LCCR0). The user must program the control bits within all other control registers before setting LEN=1 (a word write can be used to configure LCCR0 while setting LEN after all other control registers have been programmed), and also must disable the LCD controller when changing the state of any control bit within the LCD controller. Note that writes to reserved bits are ignored and reads return zeros.

Address: 0h B010 0000 **LCCR0: LCD Control Register 0** **Read/Write**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved												PDD<7:4>			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PDD<3:0>				Reserved		DPD	BLE	PAS	Res.	ERM	BAM	LDM	SDS	CMS	LEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	LEN	LCD controller enable. 0 – LCD controller disabled. Control of L_PCLK, L_LCLK, L_FCLK, L_BIAS, and the LDD<7:0> pins is given to the PPC unit to be used as general-purpose I/O pins. 1 – LCD controller enabled.
1	CMS	Color/monochrome select. 0 – Color operation enabled. 1 – Monochrome operation enabled.
2	SDS	Single-/dual-panel display select. 0 – Single-panel display enabled. LDD<3:0> used for monochrome, LDD<7:0> used for color. 1 – Dual-panel display enabled. LDD<7:0> used for monochrome, LDD<7:0> and GPIO<9:2> used for color (user must also program GPDR and GAFR registers within the GPIO unit). Note: SDS is ignored in active mode (PAS=1). For dual-panel operation, the user must disable the LCD, set SDS, program the upper panel DMA base address, program the lower panel DMA base address, and enable the LCD.
3	LDM	LCD disable done mask. 0 – LCD disable done condition generates an interrupt (state of LDD status sent to the interrupt controller). 1 – LCD disable done condition does not generate an interrupt (LDD status bit ignored).
4	BAM	Base address update mask. 0 – Base address update condition generates an interrupt (state of BAU status sent to the interrupt controller). 1 – Base address update condition does not generate an interrupt (BAU status bit ignored).
5	ERM	Error mask. 0 – Bus error and FIFO over/underrun errors generate an interrupt (state of BER, IOL, IUL, IOU, IUU, OOL, OUL, OUU status sent to the interrupt controller). 1 – Bus error and FIFO over/underrun errors do not generate an interrupt (BER, IOL, IUL, IOU, IUU, OOL, OUL, OOU, OUU status bits ignored).
6	—	Reserved.

Bit	Name	Description
7	PAS	Passive/active display select. 0 – Passive or STN display operation enabled. Dither logic is enabled. 1 – Active or TFT display operation enable. Dither logic bypassed, pin timing changes to support continuous pixel clock, output enable, VSYNC, HSYNC signals.
8	BLE	Big/little endian select. 0 – Little endian operation is selected, half-word palette buffer data is packed into individual words of memory starting with the least significant half-word, and frame buffer pixel data is packed into individual words of memory starting with the least significant nibble, byte, or half-word. 1 – Big endian operation is selected, half-word palette buffer data is packed into individual words of memory starting with the most significant half-word, and frame buffer pixel data is packed into individual words of memory starting with the most significant nibble, byte, or half-word.
9	DPD	Double-pixel data pin mode. 0 – In single-panel monochrome operation, four pixels are presented to **LDD<3:0>** each pixel clock. 1 – In single-panel monochrome operation, eight pixels are presented to **LDD<7:0>** each pixel clock. **Note:** This bit is ignored in all other modes of operation except for single-panel monochrome.
11..10	—	Reserved.
19..12	PDD	Palette DMA request delay. Value (from 0 to 255) used to specify the number of memory controller clocks (half the speed of the CPU clock). The on-chip palette DMA request should be disabled after each DMA transfer to the palette. The clock count starts after the last write of each burst cycle. While the counter is decrementing, all DMA requests from the palette are masked. When the counter reaches zero, any pending or subsequent DMA requests are allowed to generate a 4-word burst. Programming PDD=8h'00 disables this function.
31..20	—	Reserved.

11.7.4 LCD Controller Control Register 1

LCD controller control register 1 (LCCR1) contains four bit fields that are used as modulus values for a collection of down counters, each of which performs a different function to control the timing of several of the LCD's pins.

11.7.4.1 Pixels Per Line (PPL)

The pixels per line (PPL) bit-field is used to specify the number of pixels in each line or row on the screen. PPL is a 10-bit value that represents between 16 and 1024 pixels per line. PPL is used to count the correct number of pixel clocks that must occur before the line clock can be asserted. The user should program PPL with the desired number of pixels per line minus 16. Note that the bottom four bits of PPL are not implemented and therefore are not writable. Reads of these bits return zeros because the LCD controller only supports displays that are a multiple of 16 pixels wide.

Many displays exist that are not a multiple of 16, but are able to ignore added pixels at the end of each line. For example, if the display being controlled is 250 pixels wide, the nearest greater multiple of 16 is 256. The user should program PPL to 256-16 = 240 (10'h0F0). In this case, the user must also add the appropriate number of "dummy" pixel values (between 1 and 15) to the frame buffer. Again, for a 250 pixel wide display, and if 4-bit/pixel mode is used, each line is 250 nibbles or 125 bytes in length. The next nearest pixel boundary occurs at 256 pixels or nibbles (128 bytes). Thus the user must start each new line in the frame buffer at multiples of 128 bytes by adding an extra 6 "dummy" pixels per line (3 bytes). Note that the user must also ensure that the display that is being controlled will ignore any additional pixel clocks at the end of each line because these "dummy" pixel values will be output to the display and the pixel clock will continue to transition until the PPL+16 value is reached.

11.7.4.2 Horizontal Sync Pulse Width (HSW)

The 6-bit horizontal sync pulse width (HSW) field is used to specify the pulse width of the line clock in passive mode or horizontal synchronization pulse in active mode. L_LCLK is asserted each time a line or row of pixels is output to the display and a programmable number of pixel clock waitstates have elapsed. When line clock is asserted, the value in HSW is transferred to a 6-bit down counter, which uses the programmed pixel clock frequency to decrement. When the counter reaches zero, the line clock is negated. HSW can be programmed to generate a line clock pulse width ranging from 1 to 64 pixel clock periods. The user should program HSW with the desired number of pixel clocks minus one. Note that the pixel clock does not transition during the line clock pulse in passive display mode, but does transition in active display mode. Also note that the polarity (active and inactive state) of the line clock pin is programmed using the horizontal sync polarity (HSP) bit in LCCR3.

11.7.4.3 End-of-Line Pixel Clock Wait Count (ELW)

The 8-bit end-of-line pixel clock wait count (ELW) field is used to specify the number of "dummy" pixel clocks to insert at the end of each line or row of pixels before pulsing the line clock pin. Once a complete line of pixels is transmitted to the LCD driver, the value in ELW is used to count the number of pixel clocks to wait before pulsing the line clock. ELW generates a wait period ranging from 1 to 256 pixel clock cycles. The user should program ELW with the desired number of pixel clocks minus one. Note that the pixel clock pin, L_PCLK, does not transition during the these "dummy" pixel clock cycles in passive display mode (pixel clock transitions continuously in active display mode).

11.7.4.4 Beginning-of-Line Pixel Clock Wait Count (BLW)

The 8-bit beginning-of-line pixel clock wait count (BLW) field is used to specify the number of "dummy" pixel clocks to insert at the beginning of each line or row of pixels. After the line clock for the previous line has been negated, the value in BLW is used to count the number of pixel clocks to wait before starting to output the first set of pixels in the next line. BLW generates a wait period ranging from 1 to 256 pixel clock cycles. The user should program BLW with the desired number of pixel clocks minus one. Note that the pixel clock pin, L_PCLK, does *not* transition during these "dummy" pixel clock cycles in passive display mode (pixel clock transitions continuously in active display mode).

The following table shows the location of the four bit fields located in LCD control register 1 (LCCR1). The LCD controller must be disabled (LEN=0) when changing the state of any field within this register.

Address: 0h B010 0020 **LCCR1: LCD Controller Control Register 1** **Read/Write**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	BLW								ELW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	HSW						PPL<9:4>						PPL<3:0>			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
9..0	PPL	Pixels per line. Value (from 1 to 1024). Used to specify number of pixels contained within each line on the LCD display. Pixels/line = (PPL+16). Note that **PPL<3:0>** are not implemented but return zeros when read.
15..10	HSW	Horizontal sync pulse width. Value (from 1 to 64). Used to specify number of pixel clock periods to pulse the line clock at the end of each line. HSYNC pulse width = (HSW+1). Note that pixel clock is held in its inactive state during the generation of the line clock in passive display mode and is permitted to transition in active display mode.
23..16	ELW	End-of-line pixel clock wait count. Value (from 1 to 256). Used to specify number of pixel clock periods to add to the end of a line transmission before line clock is asserted. EOL = (ELW+1). Note that pixel clock is held in its inactive state during the end-of-line wait period in passive display mode and is permitted to transition in active display mode.
31..24	BLW	Beginning-of-line pixel clock wait count. Value (from 1 to 256). Used to specify number of pixel clock periods to add to the beginning of a line transmission before the first set of pixels is output to the display. BOL wait = (BLW+1). Note that pixel clock is held in its inactive state during the beginning-of-line wait period in passive display mode and is permitted to transition in active display mode.

11.7.5 LCD Controller Control Register 2

LCD controller control register 2 (LCCR2) contains four bit fields that are used as modulus values for a collection of down counters, each of which performs a different function to control the timing of several of the LCD's pins.

11.7.5.1 Lines Per Panel (LPP)

The lines per panel (LPP) bit field is used to specify the number of lines or rows present on the LCD panel being controlled. In single-panel mode, it represents the total number of lines for the entire LCD display. In dual-panel mode, it represents half the number of lines of the entire LCD display because it is split into two panels. LPP is a 10-bit value that represents between 1 and 1024 lines per screen. The user should program LPP with the desired height of the display minus one. LPP is used to count the correct number of line clocks that must occur before the frame clock can be pulsed.

The LCD's DMA may overshoot the end of frame buffer by one burst cycle (4-word read). The LCD's DMA reads these extra values but they are flushed from the input FIFO each time the frame clock is pulsed. The user must ensure that the four words immediately following the end of the frame buffer reside in legal memory space (do not cause a bus error if read). Because the LCD does not alter this memory (only reads are performed), these locations can be used for data storage unrelated to the LCD.

11.7.5.2 Vertical Sync Pulse Width (VSW)

The 6-bit vertical sync pulse width (VSW) field is used to specify the pulse width of the vertical synchronization pulse in active mode, or is used to add extra "dummy" line clock waitstates between the end and beginning of frame in passive mode.

In active mode (PAS=1), L_FCLK is used to generate the vertical sync signal and is asserted each time the last line or row of pixels for a frame is output to the display and a programmable number of line clock waitstates have elapsed as specified by ELW. When L_FCLK is asserted, the value in VSW is transferred to a 6-bit down counter, which uses the line clock frequency to decrement. When the counter reaches zero, L_FCLK is negated. VSW can be programmed to generate a vertical sync pulse width ranging from 1 to 64 line clock periods. The user should program VSW with the desired number of line clocks minus one. Note that the line clock does not transition during generation of the vertical sync pulse. Also note that the polarity (active and inactive state) of the L_FCLK pin is programmed using the frame clock polarity (FCP) bit in LCCR3.

In passive mode (PAS=0), VSW does not affect the timing of the L_FCLK pin, but rather can be used to add extra line clock waitstates between the end of each frame and the beginning of the next frame. When the last line clock of a frame is negated, the value in VSW is transferred to a 6-bit down counter that uses the line clock frequency to decrement. When the counter reaches zero, the next frame is permitted to begin. VSW can be programmed to generate from 1 to 64 dummy line clock periods between each frame in passive mode. The user should program VSW properly to ensure that enough waitstates occur between frames such that the LCD's DMA is able to fully load the on-chip palette, as well as allowing a sufficient number of encoded pixel values to be input from the frame buffer, to be processed by the dither logic, and placed in the output FIFO, ready to be output to the LCD's data pins. The number of waitstates required is system dependent. The factors that determine the number of waitstates include: palette buffer size (32 or 512 bytes), memory system speed (number of waitstates, burst speed, number of beats), and what value is programmed in the palette DMA request delay (PDD) bit-field in LCCR0. Note that the line clock pin *does* transition during the insertion of the line clock waitstate periods.

VSW does not affect generation of the frame clock signal in passive mode. Passive LCD displays require that the frame clock is active on the rising edge of the first line clock pulse of each frame, with adequate setup and hold time. To meet this requirement, the LCD controller's frame clock pin is asserted on the rising edge of the first pixel clock for each frame. The frame clock remains asserted for the remainder of the first line as pixels are output to the display and it is then negated on the rising edge of the first pixel clock of the second line of each frame.

11.7.5.3 End-of-Frame Line Clock Wait Count (EFW)

The 8-bit end-of-frame line clock wait count (EFW) field is used in active mode (PAS=1) to specify the number of line clocks to insert at the end of each frame. Once a complete frame of pixels is transmitted to the LCD display, the value in EFW is used to count the number of line clock periods to wait. After the count has elapsed, the VSYNC (L_FCLK) signal is pulsed. EFW generates a wait period ranging from 0 to 255 line clock cycles (setting EFW=8'h00 disables the EOF wait count). Note that the line clock pin, L_LCLK, does not transition during the generation of the EFW line clock periods.

In passive mode, EFW should be set to zero such that no end-of-frame waitstates are generated. VSW should be used exclusively in passive mode to insert line clock waitstates to allow the LCD's DMA to fill the palette and process a number of pixels before the start of the next frame.

11.7.5.4 Beginning-of-Frame Line Clock Wait Count (BFW)

The 8-bit beginning-of-frame line clock wait count (BFW) field is used in active mode (PAS + 1) to specify the number of line clocks to insert at the beginning of each frame. The BFW count starts just after the VSYNC signal for the previous frame has been negated. After this has occurred, the value in BFW is used to count the number of line clock periods to insert before starting to output pixels in the next frame. BFW generates a wait period ranging from 0 to 255 extra line clock cycles (BFW=8'h00 disables the BOF wait count). Note that the line clock pin, L_LCLK, does transition during the generation of the BFW line clock wait periods.

In passive mode, BFW should be set to zero such that no beginning-of-frame waitstates are generated. VSW should be used exclusively in passive mode to insert line clock waitstates to allow the LCD's DMA to fill the palette and process a number of pixels before the start of the next frame.

The following table shows the location of the four bit fields located in LCD control register 2 (LCCR2). The LCD controller must be disabled (LEN=0) when changing the state of any field within this register.

Address: 0h B010 0024 **LCCR2: LCD Controller Control Register 2** **Read/Write**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	BFW								EFW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

-	-															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	VSW						LPP									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
9..0	LPP	Lines per panel. Value (from 1 to 1024). Used to specify number of lines per panel. For single-panel mode, this represents the total number of lines on the LCD display; for dual-panel mode, this represents half the number of lines on the whole LCD display. Lines/panel = (LPP+1).
15..10	VSW	Vertical sync pulse width. In active mode (PAS=1), value (from 1 to 64). Used to specify number of line clock periods to pulse the L_FCLK pin at the end of each frame after the end-of-frame wait (EFW) period elapses. Frame clock used as VSYNC signal in active mode. In passive mode (PAS=0), value (from 1 to 64). Used to specify number of extra line clock periods to insert after the end-of-frame. Note that the width of L_FCLK is not affected by VSW in passive mode and that line clock does not transition during the insertion of the extra line clock periods. Also note that both EFW and BFW should be set to zero in passive mode. VSYNC width = (VSW+1).
23..16	EFW	End-of-frame line clock wait count. In active mode (PAS=1), value (from 0 to 255). Used to specify number of line clock periods to add to the end of each frame. Note that line clock does transition during the insertion of the extra line clock periods. EFW should be cleared to zero (disabled) in passive mode.
31..24	BFW	Beginning-of-frame line clock wait count. In active mode (PAS=1), value (from 0 to 255). Used to specify number of line clock periods to add to the beginning of a frame before the first set of pixels is output to the display. Note that line clock does transition during the insertion of the extra line clock periods. BFW should be cleared to zero (disabled) in passive mode.

11.7.6 LCD Controller Control Register 3

LCD controller control register 3 (LCCR3) contains seven different bit fields that are used to control various functions within the LCD controller.

11.7.6.1 Pixel Clock Divider (PCD)

The 8-bit pixel clock divider (PCD) field is used to select the frequency of the pixel clock. PCD can be any value from 1 to 225 (0 is illegal) and is used to generate a range of pixel clock frequencies from CCLK/6 to CCLK/514 (where CCLK is the programmed frequency of the CPU clock). The pixel clock frequency should be adjusted to meet the required screen refresh rate. The refresh rate depends on: the number of pixels for the target display; whether single- or dual-panel mode is selected; whether monochrome or color mode is selected; the number of pixel clock waitstates programmed at the beginning and end of each line; the number of line clocks inserted at the beginning and end of each frame; the width of the VSYNC signal in active mode or VSW line clocks inserted in passive mode; and the width of the frame clock or HSYNC signal. All of these factors alter the time duration from one frame transmission to the next. Different display manufacturers require different frame refresh rates depending on the physical characteristics of the display. PCD is used to alter the pixel clock frequency in order to meet these requirements. The frequency of the pixel clock for a set PCD value or the required PCD value to yield a target pixel clock frequency can be calculated using the two following equations. Note that programming PCD = 8’h00 is illegal.:

$$PixelClock = \frac{CCLK}{2(PCD + 2)}$$

$$PCD = \frac{CCLK}{2(PixelClock)} - 2$$

11.7.6.2 AC Bias Pin Frequency (ACB)

The 8-bit ac bias frequency (ACB) field is used to specify the number of line clock periods to count between each toggle of the ac bias pin (L_BIAS). In passive mode, after the LCD controller is enabled, the value in ACB is loaded to an 8-bit down counter and the counter begins to decrement using the line clock. When the counter reaches zero, it stops, the state of L_BIAS is reversed, and the whole procedure starts again. The number of line clocks between each ac bias pin transition ranges from 1 to 256. The user should program ACB with the desired number of line clocks minus one.

This pin is used by the LCD display to periodically reverse the polarity of the power supplied to the screen to eliminate dc offset. If the LCD display being controlled has its own internal means of switching its power supply, ACB should be set to its maximum value to reduce power consumption (8’hFF). Note that the ACB bit field has no effect on L_BIAS in active mode. Because the pixel clock transitions continuously in active mode, the ac bias pin is used as an output enable signal. It is asserted automatically by the LCD controller in active mode whenever pixel data is driven out to the data pins to signal the display when it may latch pixels using the pixel clock.

11.7.6.3 AC Bias Pin Transitions Per Interrupt (API)

The 4-bit ac bias pin transitions per interrupt (API) field is used to specify the number of L_BIAS pin transitions to count before setting the ac bias count status (ACS) bit in the LCD controller status register that signals an interrupt request. After the LCD controller is enabled, the value in API is loaded to a 4-bit down counter and the counter decrements each time the ac bias pin is inverted. When the counter reaches zero, it stops and the ac bias count (ABC) bit is set in the status register. Once ABC is set, the 4-bit down counter is reloaded with the value in API, and is disabled until ABC is cleared. When ABC is cleared by the CPU, the down counter is enabled and again decrements each time the ac bias pin is inverted. The number of ac bias pin transitions between each interrupt request ranges from 0 to 15. Note that programming API=4'h0 disables the ac bias pin transitions per interrupt function.

In active mode, L_BIAS is used as an output enable signal. However, signalling of the API interrupt may still occur. The ACB bit field can be used to count line clock pulses in active mode. When the programmed number of line clock pulses occurs, an internal signal is transitioned that is used to decrement the 4-bit counter used by the API interrupt logic. Once this internal signal transitions the programmed number of times, as specified by API, an interrupt is generated. The user should program API to zero if the API interrupt function is not required in active mode (PAS = 1).

11.7.6.4 Vertical Sync Polarity (VSP)

The vertical sync polarity (VSP) bit is used to select the active and inactive states of the vertical sync signal in active display mode (PAS = 1), and the frame clock signal in passive display mode. When VSP=0, the L_FCLK pin is active high and inactive low. When VSP=1, the L_FCLK pin is active low and inactive high. In active display mode, the L_FCLK pin is forced to its inactive state while pixels are transmitted during the frame. After the end of the frame and a programmable number of line clocks periods occur (controlled by EFW), the L_FCLK pin is forced to its active state for a programmable number of line clocks (controlled by VSW), and is then again forced to its inactive state. In passive display mode, the L_FCLK pin is forced to its inactive state during the transmission of the second line of each frame through to the end of the frame. Frame clock is then forced to its active state on the rising edge of the first pixel clock of each frame. Frame clock remains active during the transmission of the entire first line of pixels in the frame and is then forced back to its inactive state on the rising edge of the first pixel clock of the second line of the frame.

11.7.6.5 Horizontal Sync Polarity (HSP)

The horizontal sync polarity (HSP) bit is used to select the active and inactive states of the horizontal sync signal in active display mode, and the line clock signal in passive display mode. When HSP=0, the L_LCLK pin is active high and inactive low. When HSP=1, the L_LCLK pin is active low and inactive high. Both in active and passive display modes, the L_FCLK pin is forced to its inactive state whenever pixels are transmitted After the end of each line and a programmable number of pixel clock periods occur (controlled by ELW), the L_FCLK pin is forced to its active state for a programmable number of line clocks (controlled by HSW), and is then again forced to its inactive state.

11.7.6.6 Pixel Clock Polarity (PCP)

The pixel clock polarity (PCP) bit is used to select which edge of the pixel clock data is driven out onto the LCD's data pins. When PCP=0, data is driven onto the LCD's data pins on the rising edge of the L_PCLK pin. When PCP=1, data is driven onto the LCD's data pins on the falling edge of the L_PCLK pin.

11.7.6.7 Output Enable Polarity (OEP)

The output enable polarity (OEP) bit is used to select the active and inactive states of the output enable signal in active display mode. In this mode, the ac bias pin is used as an enable that signals the off-chip device when data is actively being driven out using the pixel clock. The pixel clock continuously toggles during operation of active mode (PAS=1). When OEP=0, the L_BIAS pin is active high and inactive low. When OEP=1, the L_BIAS pin is active low and inactive high. In active display mode, data is driven onto the LCD's data pins on the programmed edge of the L_PCLK pin when L_BIAS is in its active state. Note that OEP does not affect L_BIAS in passive display mode.

The following table shows the location of the seven different bit fields located in LCD controller control register 3 (LCCR3). The LCD controller must be disabled (LEN=0) when changing the state of any field within this register. Note that writes to reserved bits are ignored and reads return zeros.

Address: 0h B010 0028 LCCR3: LCD Controller Control Register 3 Read/Write

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved								OEP	PCP	HSP	VSP	API			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-	-															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ACB								PCD							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
7..0	PCD	Pixel clock divisor. Value (from 0 to 255). Used to specify the frequency of the pixel clock based on the CPU clock (CCLK) frequency. Pixel clock frequency can range from CCLK/6 to CCLK/514. Pixel Clock Frequency = CCLK/2(PCD+2). Note that PCD must be programmed with a value of 1 or greater (PCD = 8'h00 is illegal).
15..8	ACB	AC bias pin frequency. Value (from 1 to 256). Used to specify the number of line clocks to count before transitioning the ac bias pin in passive mode (PAS=0). This pin is used to periodically invert the polarity of the power supply to prevent dc charge buildup within the display. If the passive display that is being controlled does not need to use L_BIAS, the user should program ACB to its maximum value (8'hFF) to conserve power. Note that ACB is ignored in active mode (PAS = 1). Number of line clocks/toggle of the L_BIAS pin = (ACB+1).
19..16	API	AC bias pin transitions per interrupt. Value (from 0 to 15). Used to specify the number of ac bias pin transitions to count before setting the line count status (ABC) bit, signalling an interrupt request. Counter frozen when ABC is set and is restarted when ABC is cleared by software. This function is disabled when API=4'h0.
20	VSP	Vertical sync polarity. 0 – L_FCLK pin is active high and inactive low. 1 – L_FCLK pin is active low and inactive high. Active mode: Vertical sync pulse active between frames, after end-of-frame wait period. Passive mode: Frame clock active during first line of each frame.

Bit	Name	Description
21	HSP	Horizontal sync polarity. 0 – L_LCLK pin is active high and inactive low. 1 – L_LCLK pin is active low and inactive high. Active and passive mode: horizontal sync pulse/line clock active between lines, after end-of-line wait period.
22	PCP	Pixel clock polarity. 0 – Data is driven on the LCD's data pins on the rising edge of L_PCLK. 1 – Data is driven on the LCD's data pins on the falling edge of L_PCLK.
23	OEP	Output enable polarity. 0 – L_BIAS pin is active high and inactive low in active display mode and parallel data input mode. 1 – L_BIAS pin is active low and inactive high in active display mode and parallel data input mode. In active display mode, data is driven out to the LCD's data pins on programmed pixel clock edge when ac bias pin is active. Note that OEP is ignored in passive display mode.
31..24	—	Reserved.

11.7.7 LCD Controller DMA Registers

The LCD controller has two fully independent DMA channels used to transfer frame buffer data for each frame displayed from off-chip memory to the LCD's palette RAM and the input FIFO. DMA channel 1 is used for single-panel display mode and the upper screen in dual-panel mode. DMA channel 2 is used exclusively for the lower screen in dual-panel mode. Both DMA channels contain a base address pointer and current address pointer register. The LCD's DMA engine has the highest priority to gain mastership of the SA-1100's internal ARM system bus. The LCD is given highest priority to prevent other masters from starving the LCD screen (including the CPU).

The two DMA channels use a separate set of base address and current address pointers. The user must initialize the base address pointer registers before enabling the LCD. Once enabled, the base address is transferred to the current address pointer.

After the LCD is enabled, the input FIFO requests a DMA transfer and the DMA makes a 4-word burst access from off-chip memory using the address contained within the current address pointer. The pointer is incremented by 4 (bytes) each time a word is read from memory (bit 2 of the pointer is incremented). Each of the 4 words from the burst is loaded into the top of the input FIFO. The LCD then takes one value at a time from the bottom of the FIFO, unpacks it into individual encoded pixel values, and uses the values to index into the palette. Each time the input FIFO contains four empty entries, another DMA request is made and another 4-word burst is performed. To calculate the frame buffer end address, the DMA controller uses the values programmed in the pixels per line (PPL), lines per panel (LPP), single/dual screen select (SDS), color/monochrome select (CMS) bit fields, and double pixel data (DPD) bit fields within the control registers as well as the pixel bit size (PBS) field contained within the first entry of the palette buffer from the off-chip frame buffer. When the current address pointer reaches the calculated end of buffer address, the value in the base address pointer is again transferred to the current address pointer.

11.7.8 DMA Channel 1 Base Address Register

DMA channel 1 base address register (DBAR1) is a 32-bit register that is used to specify the base address of the off-chip frame buffer for DMA channel 1. The base address pointer register can be both read and written. Addresses programmed in the base address register must be aligned on quadword boundaries; the least significant four bits (DBAR1<3:0>) must always be written with zeros. The user must initialize the base address register before enabling the LCD, and can also write a new value to it while the LCD is enabled to allow a new frame buffer to be used for the next frame. The user can change the state of DBAR1 while the LCD controller is active just after the base address update (BAU) status bit is set with the LCD's status register, which generates an interrupt request. This status bit indicates that the value in the base address pointer has transferred to the current address pointer register and that it is safe to write a new base address value. DMA channel 1 is used to transfer frame buffer data from off-chip memory to the LCD's input FIFO and the palette RAM for single-panel mode, and for the top half of the screen in dual-panel mode. For dual-panel operation, the user must perform the following sequence in order: disable the LCD (LEN=0), program dual panel mode (SDS= 0 → 1), write the upper panel DMA base address, write the lower panel DMA base address, enable the LCD (LEN= 0 → 1). Note that DBAR1 is not reset and must be initialized before enabling the LCD; question marks indicate that the values are unknown at reset.

Address: 0h B010 0010 **DBAR1: DMA Channel 1 Base Address Register** **Read/Write**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DMA Channel 1 Base Address Pointer															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DMA Channel 1 Base Address Pointer															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	Name	Description
31..0	DBAR1	DMA channel 1 base address pointer. Used to specify the base address of the frame buffer within off-chip memory. Value in DBAR1 is transferred to current address pointer register 1 when LCD is first enabled (LEN= 0 → 1) and when the current address pointer value equals the end-of-frame buffer. DBAR1 should be written only when the LCD is disabled or immediately after an interrupt is generated by the setting of the base address update (BAU) status bit. The base address must be on a quadword boundary; the user must always write bits 0 through 3 to zero.

11.7.9 DMA Channel 1 Current Address Register

DMA channel 1 current address register (DCAR1) is a 32-bit read-only register that is used by DMA channel 1 to keep track of the address of the DMA transfer currently in progress or the address of the next DMA transfer. Any time the LCD is first enabled (LEN= 0 → 1) or the value in the current address pointer register equals the calculated end address value, the contents of the base address pointer register is transferred to the current address pointer. This register can be read to determine the approximate line that the LCD controller is currently processing and driving out to the display. It is also useful to read this register just before writing the DMA's base address pointer to ensure that the end of frame is not about to occur, which means that the base address pointer is about to be transferred to the current address pointer. Note that DCAR1 is a read-only register that is not reset and is not initialized until the LCD is first enabled, causing the contents of the base address register to be transferred to it; question marks indicate that the values are unknown at reset.

Address: 0h B010 0014 **DCAR1: DMA Channel 1 Current Address Register** **Read-Only**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DMA Channel 1 Current Address Pointer															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

-	-															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DMA Channel 1 Current Address Pointer															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	Name	Description
31..0	DCAR1	DMA channel 1 current address pointer. Read-only register. Continuously reflects the current address that DMA channel 1 is transferring from or will use in the next transfer. Base address register is transferred to this register whenever the LCD is enabled (LEN= 0 → 1) and when the current address is equal to the calculated end address of the buffer.

11.7.10 DMA Channel 2 Base and Current Address Registers

DMA channel 2's base and current address registers (DBAR2 and DCAR2) function exactly like DMA channel 1's except that they are used exclusively for dual-panel operation. (See the preceding sections.) When SDS=1, DMA channel 2 is used to supply frame buffer data to the lower half of the display. Note that the palette buffer, which resides within the first 16 or 256 entries of the frame buffer, is utilized only by DMA channel 1. The user should not place palette entries into the frame buffer for DMA channel 2. The base address for channel 2 points to the first encoded pixel values for the lower half of the display. For dual-panel operation, the user must perform the following sequence in order: disable the LCD (LEN=0), program dual-panel mode (SDS= 0 → 1), write the upper panel DMA base address, write the lower DMA base address and enable the LCD (LEN= 0 → 1). The following figures show the format of these registers; question marks indicate that the values are unknown at reset.

Address: 0h B010 0018 **DBAR2: DMA Channel 2 Base Address Register** **Read/Write**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DMA Channel 2 Base Address Pointer															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DMA Channel 2 Base Address Pointer															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	Name	Description
31..0	DBAR2	DMA channel 2 base address pointer. Used to specify the base address of the frame buffer within off-chip memory for the lower half of the display in dual-panel operation. Value in DBAR2 is transferred to current address pointer register 2 when LCD is first enabled (LEN= 0 → 1) and when the current address pointer value reaches the end-of-frame buffer. DBAR2 should be written only when the LCD is disabled or immediately after an interrupt is generated by setting the base address update status (BAU) bit. The base address must be on a quadword boundary. The user must always write bits 0 through 3 to zero.

Address: 0h B010 001C **DCAR2: DMA Channel 2 Current Address Register** **Read-Only**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DMA Channel 2 Current Address Pointer															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DMA Channel 2 Current Address Pointer															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bit	Name	Description
31..0	DCAR2	DMA channel 2 current address pointer. Read-only register. Continuously reflects the current address that DMA channel 2 is transferring from or will use in the next transfer. Base address register is transferred to this register whenever the LCD is first enabled and when the current address is equal to the calculated end address of the buffer.

11.7.11 LCD Controller Status Register

The LCD controller status register (LCSR) contains bits that signal overrun and underrun errors for both the input and output FIFOs, ac bias pin transition count, LCD disabled, DMA base update ready, and DMA transfer bus error conditions. Each of these hardware-detected events signal an interrupt request to the interrupt controller.

Each of the LCD's status bits signal an interrupt request as long as the bit is set. Once the bit is cleared, the interrupt is cleared. Read/write bits are called status bits (read-only bits are called flags). Status bits are referred to as "sticky" (once set by hardware, they must be cleared by software). Writing a one to a sticky status bit clears it; writing a zero has no effect. Read-only flags are set and cleared by hardware; writes have no effect. The user has the ability to mask all LCD interrupts by clearing bit 12 within the interrupt controller mask register (ICMR). See the Section 9.2, "Interrupt Controller" on page 9-11.

11.7.11.1 LCD Disable Done Flag (LDD) (read/write, maskable interrupt)

The LCD disable done flag (LDD) is set after the LCD has been disabled and the frame that is active finishes being output to the LCD's data pins. When the LCD is disabled by clearing the LCD enable bit (LEN= 0 $\rightarrow$ 1) in LCCR0, the LCD allows the current frame to complete before it is disabled. After the last set of pixels is clocked out onto the LCD's data pins by the pixel clock, the LCD is disabled, LDD is set, and an interrupt request is made to the interrupt controller if it is unmasked (LDM=0). This interrupt is useful to allow an orderly shutdown of the LCD controller before the user places the SA-1100 into sleep mode.

11.7.11.2 Base Address Update Flag (BAU) (read-only, maskable interrupt)

The base address update flag (BAU) is a read-only bit that is set after the contents of the DMA base address register 1 are transferred to the DMA current address register 1 and is cleared when DMA base address register 1 is written. The value in the base address register is transferred to the current address register when the LCD is first enabled by writing a one to LEN (LEN= 0 $\rightarrow$ 1) and when the current address pointer equals the end address value calculated by the LCD controller. When BAU is set, an interrupt request is made to the interrupt controller if it is unmasked (BAM = 0). This interrupt allows the user to program the DMA with a new base address value to alternate between two or more frame buffer locations. When dual-panel mode is enabled (SDS=1), both DMA channels are enabled, and BAU is set only after *both* channels' base address registers are transferred to their corresponding current address registers (1 and 2) and is cleared when DMA base address register 2 (lower panel) is written. Therefore, the user must always update the DMA base address register 1 (upper panel) first in dual-panel mode.

11.7.11.3 Bus Error Status (BER) (read/write, maskable interrupt)

The bus error status (BER) bit is set when a DMA transfer causes a bus error to occur on the ARM system bus. A bus error is signalled when the DMA controller attempts to access a reserved or nonexistent memory space. When this occurs, the SA-1100's memory controller returns zeros for the read. It asserts the bus error signal to the LCD's DMA, which in turn, causes the BER bit to be set and an interrupt request is made to the interrupt controller if it is unmasked (ERM = 0). The DMA is not disabled as a result of the bus error and operation continues as normal. If a DMA access causes a bus error, zeros are returned by the memory controller, which causes a palette entry to be filled with zeros (highest intensity color or black), or if pixel data is being DMAed, the LCD accesses the first location of the palette RAM one or more times.

11.7.11.4 AC Bias Count Status (ABC) (read/write, nonmaskable interrupt)

The ac bias count status (ABC) bit it set each time the ac bias pin (L_BIAS) transitions a particular number of times as specified by the ac bias pin transitions per interrupt (API) field in LCCR3. If API is programmed with a nonzero value, a counter is loaded with the value in API and is decremented each time the L_BIAS pin reverses state. When the counter reaches zero, the ABC bit is set, which signals an interrupt request to the interrupt controller. The counter reloads using the value in API, but does not start to decrement again until ABC is cleared by the user.

11.7.11.5 Input FIFO Overrun Lower Panel Status (IOL) (read/write, maskable interrupt)

The input FIFO overrun lower panel status (IOL) bit is set when the LCD's DMA channel 2 attempts to place data into the lower panel's input FIFO after it has been completely filled. It is cleared by writing a one to the bit. This bit is used only in dual-panel mode (SDS=1). When this bit is set, an interrupt request is made to the interrupt controller if it is unmasked (ERM=0).

11.7.11.6 Input FIFO Underrun Lower Panel Status (IUL) (read/write, maskable interrupt)

The input FIFO underrun lower panel status (IUL) bit is set when the lower panel's input FIFO is completely empty and the LCD's pixel unpacking logic attempts to fetch data from the FIFO. It is cleared by writing a one to the bit. This bit is used only in dual-panel mode (SDS=1). When this bit is set, an interrupt request is made to the interrupt controller if it is unmasked (ERM=0).

11.7.11.7 Input FIFO Overrun Upper Panel Status (IOU) (read/write, maskable interrupt)

The input FIFO overrun upper panel status (IOU) bit is set when the LCD's DMA channel 1 attempts to place data into the upper panel's input FIFO after it has been completely filled. It is cleared by writing a one to the bit. This bit is used in single-panel mode (SDS=0) and dual-panel mode (SDS=1). When this bit is set, an interrupt request is made to the interrupt controller if it is unmasked (ERM=0).

11.7.11.8 Input FIFO Underrun Upper Panel Status (IUU) (read/write, maskable interrupt)

The input FIFO underrun upper panel status (IUU) bit is set when the upper panel's input FIFO is completely empty and the LCD's pixel unpacking logic attempts to fetch data from the FIFO. It is cleared by writing a one to the bit. This bit is used in single-panel mode (SDS=0) and dual-panel mode (SDS=1). When this bit is set, an interrupt request is made to the interrupt controller if it is unmasked (ERM=0).

11.7.11.9 Output FIFO Overrun Lower Panel Status (OOL) (read/write, maskable interrupt)

The output FIFO overrun lower panel status (OOL) bit is set when the LCD's dither logic attempts to place data into the lower panel's output FIFO after it has been completely filled. It is cleared by writing a one to the bit. This bit is used only in dual-panel mode (SDS=1). When this bit is set, an interrupt request is made to the interrupt controller if it is unmasked (ERM = 0).

11.7.11.10 Output FIFO Underrun Lower Panel Status (OUL) (read/write, maskable interrupt)

The output FIFO underrun lower panel status (OUL) bit is set when the lower panel's output FIFO is completely empty and the LCD's data pin driver logic attempts to fetch data from the FIFO. It is cleared by writing a one to the bit. This bit is used only in dual-panel mode (SDS=1). When this bit is set, an interrupt request is made to the interrupt controller if it is unmasked (ERM=0).

11.7.11.11 Output FIFO Overrun Upper Panel Status (OOU) (read/write, maskable interrupt)

The output FIFO overrun upper panel status (OOU) bit is set when the LCD's dither logic attempts to place data into the upper panel's output FIFO after it has been completely filled. It is cleared by writing a one to the bit. This bit is used in single-panel mode (SDS=0) and dual-panel mode (SDS=1). When this bit is set, an interrupt request is made to the interrupt controller if it is unmasked (ERM=0).

11.7.11.12 Output FIFO Underrun Upper Panel Status (OUU) (read/write, maskable interrupt)

The output FIFO underrun upper panel status (OUU) bit is set when the upper panel's output FIFO is completely empty and the LCD's data pin driver logic attempts to fetch data from the FIFO. It is cleared by writing a one to the bit. This bit is used in single-panel mode (SDS=0) and dual-panel mode (SDS=1). When this bit is set, an interrupt request is made to the interrupt controller if it is unmasked (ERM=0).

The following table shows the location of the status and flag bits in LCSR. For reserved bits, writes are ignored and reads return zero. Set status bits should be cleared by software before enabling both the LCD controller and interrupt controller.

Address: 0h B010 0004 — **LCSR: LCD Status Register** — **Read/Write & Read-Only**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				OUU	OOU	OUL	OOL	IUU	IOU	IUL	IOL	ABC	BER	BAU	LFD
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	Name	Description
0	LDD	LCD disable done flag. 0 – LCD has not been disabled and the last active frame completed. 1 – LCD has been disabled and the last active frame has just completed.
1	BAU	Base address update flag (read-only). 0 – Base address has been written and has not yet been transferred to the current address register. 1 – Base address has been transferred to the current address register, triggered either by enabling the LCD or when the current address pointer equals the end address value calculated by the LCD.

Bit	Name	Description
2	BER	Bus error status. 0 – DMA has not attempted an access to reserved/nonexistent memory space. 1 – DMA has attempted an access to a reserved/nonexistent location in external memory. The errant DMA read returns zeros.
3	ABC	AC bias count status. 0 – AC bias transition counter has not decremented to zero, or API is programmed to all zeros. 1 – AC bias transition counter has decremented to zero, indicating that the L_BIAS pin has transitioned the number of times specified by the API control bit field. Counter is reloaded with the value in API but is disabled until the user clears ABC.
4	IOL	Input FIFO overrun lower panel status. 0 – Input FIFO for the lower panel display has not overrun. 1 – DMA attempted to place data into the input FIFO for the lower panel after it has been filled.
5	IUL	Input FIFO underrun lower panel status. 0 – Input FIFO for the lower panel display has not underrun. 1 – DMA not supplying data to input FIFO for the lower panel at a sufficient rate. FIFO has completely emptied; pixel unpacking logic has attempted to take added data from the FIFO.
6	IOU	Input FIFO overrun upper panel status. 0 - Input FIFO for the upper or whole panel display has not overrun. 1 - DMA attempted to place data into the input FIFO for the upper or whole panel after it has been filled.
7	IUU	Input FIFO underrun upper panel status. 0 – Input FIFO for the upper or whole panel display has not underrun. 1 – DMA not supplying data to input FIFO for the upper or whole panel at a sufficient rate. FIFO has completely emptied; pixel unpacking logic has attempted to take added data from the FIFO.
8	OOL	Output FIFO overrun lower panel status. 0 – Output FIFO for the lower panel display has not overrun. 1 – Dither logic attempted to place data into the output FIFO for the lower panel after it had been filled.
9	OUL	Output FIFO underrun lower panel status. 0 – Output FIFO for the lower panel display has not underrun. 1 – LCD dither logic not supplying data to output FIFO for the lower panel at a sufficient rate. FIFO has completely emptied and data pin driver logic has attempted to take added data from the FIFO.
10	OOU	Output FIFO overrun upper panel status. 0 – Output FIFO for the upper or whole panel display has not overrun. 1 – Dither logic attempted to place data into the output FIFO for the upper or whole panel after it had been filled.
11	OUU	Output FIFO underrun upper panel status. 0 – Output FIFO for the upper or whole panel display has not underrun. 1 – LCD dither logic not supplying data to output FIFO for the upper or whole panel at a sufficient rate. FIFO has completely emptied and data pin driver logic has attempted to take added data from the FIFO.
31..12	—	Reserved.

11.7.12 LCD Controller Register Locations

Table 11-9 shows the registers associated with the LCD controller and the physical addresses used to access them.

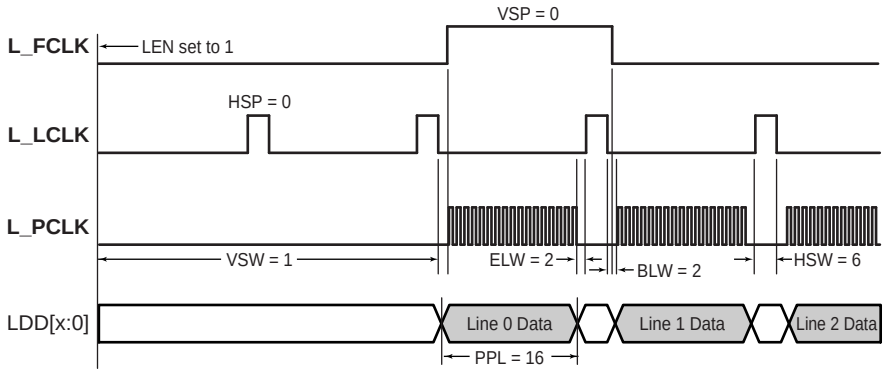
Figure 11-34 to Figure 11-38 describe the LCD controller timing parameters.

Table 11-9. LCD Controller Control, DMA, and Status Register Locations

Address	Name	Description
0hB010 0000	LCCR0	LCD controller control register 0
0hB010 0004	LCSR	LCD controller status register 1
0hB010 0008 – 0h B010 000C	—	Reserved
0hB010 0010	DBAR1	DMA channel 1 base address register
0hB010 0014	DCAR1	DMA channel 1 current address register
0hB010 0018	DBAR2	DMA channel 2 base address register
0hB010 001C	DCAR2	DMA channel 2 current address register
0hB010 0020	LCCR1	LCD controller control register 1
0hB010 0024	LCCR2	LCD controller control register 2
0hB010 0028	LCCR3	LCD controller control register 3
0hB010 002C – 0hB010 FFFF	—	Reserved

11.7.13 LCD Controller Pin Timing Diagrams

Figure 11-10. Passive Mode Beginning-of-Frame Timing

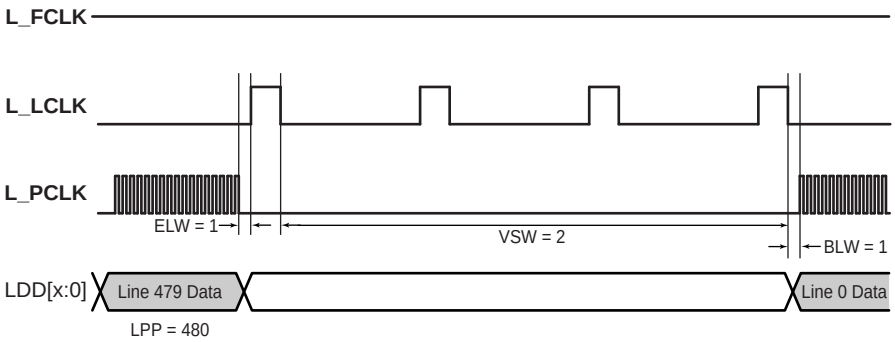


Notes:

LEN - LCD enable:
0 - LCD is disabled.
1 - LCD is enabled.
VSP - Vertical sync polarity:
0 - Frame clock is active high, inactive low.
1 - Frame clock is active low, inactive high.
VSW - Vertical Sync Pulse Width:
1 to 64 horizontal sync clock periods to assert the vertical sync signal (hsync transitions).
HSP - Horizontal sync polarity:
0 - Line clock is active high, inactive low.
1 - Line clock is active low, inactive high.
ELW - End-of-line pixel clock wait count:
1 to 256 "dummy" pixel clock periods to wait after last pixel in line before asserting line clock (pixel clock does not transition).
BLW - Beginning-of-line pixel clock wait count:
1 to 256 "dummy" pixel clock periods to wait after line clock negated before asserting pixel clocks (pixel clock does not transition).
HSW - Horizontal sync pulse width:
0 to 64 "dummy" pixel clock periods to assert the line clock (pixel clock does not transition).
PPL - Pixels per line:
16 to 1024 pixels per line on the screen (must be programmed on 16 pixel multiples).
Frame clock asserted on first pixel clock of each frame, and is negated one "dummy" pixel clock period before the first pixel clock of the 2nd line.

A4790-01

Figure 11-11. Passive Mode End-of-Frame Timing

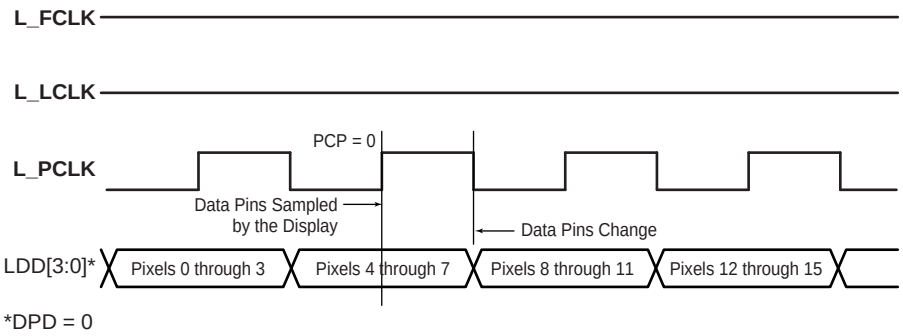


Notes:

BLW - Beginning-of-line pixel clock wait count:
0 to 256 "dummy" pixel clock periods to wait after line clock is negated before asserting pixel clocks (pixel clock does not transition).
VSW - Vertical sync pulse width:
In passive mode, 1 to 64 line clock periods to wait between the end of one frame and the beginning of the next frame (line clock transitions).
ELW - End-of-line pixel clock wait count:
1 to 256 "dummy" pixel clock periods to wait after last pixel in line before asserting line clock (pixel clock does not transition).
LPP - Lines per panel:
1 to 1024 lines per panel.

A4791-01

Figure 11-12. Passive Mode Pixel Clock and Data Pin Timing



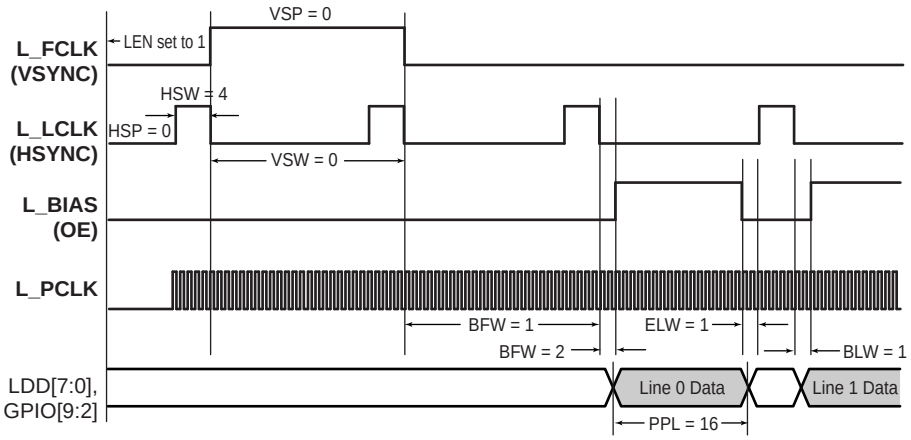
Notes:

PCP - Pixel clock polarity:
0 - Pixels sampled from data pins on rising edge of pixel clock.
1 - Pixels sampled from data pins on falling edge of pixel clock.

DPD - Dual pixel data mode:
0 - 4 data pins are used in single-panel monochrome mode.
1 - 8 data pins are used in single-panel monochrome mode.

A4792-01

Figure 11-13. Active Mode Timing

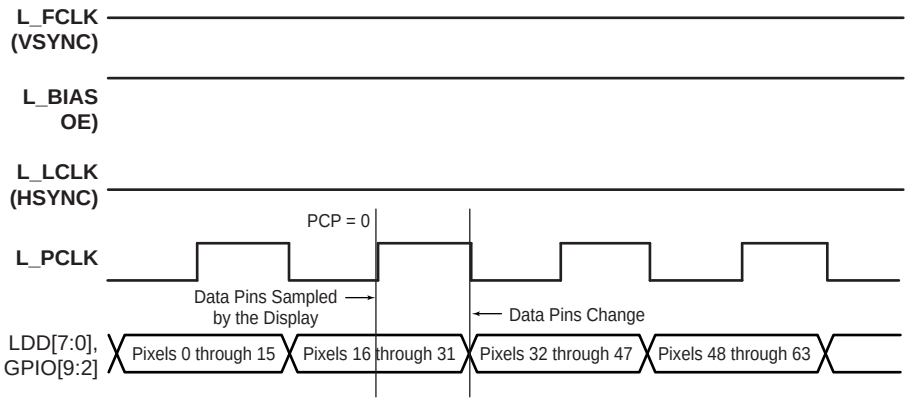


Notes:

LEN - LCD enable:
0 - LCD is disabled.
1 - LCD is enabled.
VSP - Vertical sync polarity:
0 - Vertical sync clock is active high, inactive low.
1 - Vertical sync clock is active low, inactive high.
VSW - Vertical sync width:
1 to 64 horizontal sync clock periods to assert the vertical sync signal (hsync transitions).
HSW - Horizontal sync pulse width:
1 to 64 pixel clock periods to assert the line clock (pixel clock transitions).
HSP - Horizontal sync polarity:
0 - Horizontal sync clock is active high, inactive low.
1 - Horizontal sync clock is active low, inactive high.
BFW - Beginning-of-frame horizontal sync clock wait count:
0 to 255 horizontal sync clock periods to wait at the beginning of each frame (hsync transitions).
BLW - Beginning-of-line pixel clock wait count:
1 to 256 pixel clock periods to wait after line clock negated before asserting pixel clocks (pixel clock transitions).
ELW - End-of-line pixel clock wait count:
1 to 256 pixel clock periods to wait after last pixel in line before asserting line clock (pixel clock transitions).
PPL - Pixels per line:
1 to 1024 pixels per line on screen.

A4793-01

Figure 11-14. Active Mode Pixel Clock and Data Pin Timing



Notes:

PCP - Pixel clock polarity:
0 - Pixels sampled from data pins on rising edge of pixel clock.
1 - Pixels sampled from data pins on falling edge of pixel clock.

A4794-01

11.8 Serial Port 0 – USB Device Controller

This section describes the implementation-specific options of the USB protocol for a device controller as it applies to serial port 0, such as number, type, and function of the endpoints, interrupts to the CPU, transmit/receive FIFO interface, and so on. It is assumed that the user has a working knowledge of the USB standard. The UDC is USB-compliant and supports all standard device requests issued by the host. For programmer convenience, summaries of UDC operation are provided as well as quick reference tables. However, the user should refer to the *Universal Serial Bus Specification*, Revision 1.0[1] for a full description of the USB protocol and its operation.

Serial port 0 is a universal serial bus device controller (UDC) that supports three endpoints and can operate half-duplex at a baud rate of 12 Mbps (slave only, not a host or hub controller).

The serial information transmitted by the UDC contains layers of communication protocols, the most basic of which are fields. UDC fields include: sync, packet identifier, address, endpoint, frame number, data, and CRC fields. Fields are used to produce packets. Depending on the function of a packet, a different combination and number of fields are used. Packet types include: token, start of frame, data, and handshake packets. Packets are then assembled into groups to produce frames. These frames or transactions fall into four groups: bulk, control, interrupt, and isochronous. (The UDC supports only bulk and control.) Endpoint 0, by default, is used only to communicate control transactions to configure the UDC after it is reset or hooked up (physically connected to an active USB host or hub). Endpoint 0's responsibilities include: connection, address assignment, endpoint configuration, bus enumeration, and disconnect. Endpoint 1 is used to perform bulk OUT data transactions and receiving data from the USB host; endpoint 2 is used to perform bulk IN data transactions and transmitting data to the USB host.

The UDC uses two separate FIFOs to buffer incoming and outgoing data to or from the host (16-entry x 8-bit for transmitting, and 20-entry x 8-bit for receiving). The FIFOs can be filled or emptied either by the DMA or the CPU, with service requests being signalled when either FIFO is half-full or empty. Interrupts are signalled when the receive FIFO experiences an overrun and the transmit FIFO experiences an underrun. The control endpoint 0 has an additional 8-entry x 8-bit FIFO that can only be read or written by processor reads and writes.

The external pins dedicated to this interface are UDC+ and UDC-. The USB protocol uses differential signalling between the two pins for half-duplex data transmission. A 1.5-Kohm pull-up resistor is required to be connected to the USB cable's D+ signal to pull the UDC+ pin high when not driven. This signifies the UDC is a high-speed, 12-Mbps device and provides the correct polarity for data transmission. Using differential signalling allows multiple states to be transmitted on the serial bus. These states are combined to transmit data as well as various bus conditions, including: idle, resume, start of packet, end of packet, disconnect, connect, and reset.

11.8.1 USB Operation

Following a reset of the SA-1100 or whenever the UDC is attached to a USB bus, all endpoints are automatically configured by the UDC and the UDC is forced to use the USB default address of zero. The host then assigns the UDC a unique address. At this point, the UDC is under the host's control and responds to its commands that are transmitted to endpoint 0 using control transactions. Endpoint 1 is used to perform bulk OUT data transactions, receiving data from the USB host, and endpoint 2 bulk IN data transactions, transmitting data to the USB host.

The following sections provide details of the USB protocol in a bottom-up fashion starting with signalling levels.

1. The latest revision of the *Universal Serial Bus Specification Revision 1.0* can be accessed via the World Wide Web Internet site at: http://www.teleport.com/~usb/

11.8.1.1 Signalling Levels

USB uses differential signalling to encode data and to communicate various bus conditions. The USB specification refers to the J and K data states to differentiate between high- and low-speed transmission. Because the UDC supports only 12-Mbps transmission, references are made only to actual data state 0 and actual data state 1.

Four distinct states are represented using differential data by decoding the polarity of the UDC+ and UDC- pins. Two of the four states are used to represent data. A one is represented when UDC+ is high and UDC- is low; a zero is represented when UDC+ is low and UDC- is high. The remaining two states and pairings of the four encodings are further decoded to represent the current state of the USB bus. Table 11-10 shows how seven different bus states are represented using differential signalling.

Table 11-10. USB Bus States

Bus State	UDC+/UDC- Pin Levels
Idle	UDC+ high, UDC- low (same as a 1).
Resume	UDC+ low, UDC- high (same as a 0).
Start of Packet	Transition from idle to resume.
End of Packet	UDC+ AND UDC- low for 2-bit times followed by an idle for 1-bit time.
Disconnect	UDC+ AND UDC- below single-ended low threshold for more than 2.5 µs. (Disconnect is the static bus condition that results when no device is plugged into a hub port.)
Connect	UDC+ OR UDC- high for more than 2.5 µs.
Reset	UDC+ AND UDC- low for more than 2.5 µs. (Reset is driven by the host controller and sensed by a device controller.)

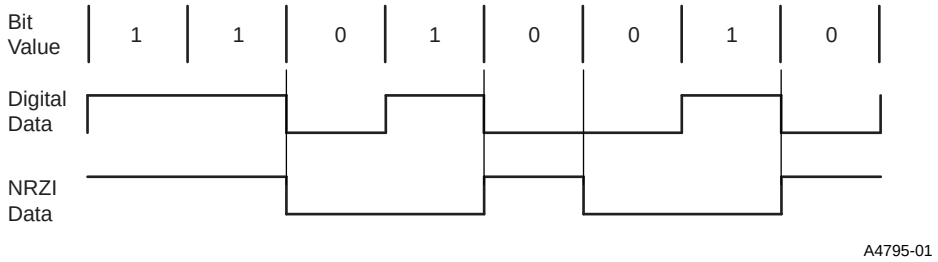
Hosts and hubs have pull-down resistors on both the D+ and D- lines. When a device is not attached to the cable, the pull-down resistors cause D+ and D- to be pulled down below the single-ended low threshold of the host or hub. This creates a state called single-ended zero (SE0). A disconnect is detected by the host when an SE0 persists for more than 2.5 µs (30-bit times). When the UDC is connected to the USB cable, the pull-up resistor on the UDC+ pin causes D+ to be pulled above the single-ended high threshold level. After 2.5 µs elapse, the host detects a connect.

After this point, the bus is in the idle state because UDC+ is high and UDC- is low. A start of packet is signalled by transitioning the bus from the idle to the resume state (a 1 to 0 transition). The beginning of each USB packet begins with a sync field, which starts with the 1-to-0 transition (see the Section 11.8.1.1, “Signalling Levels” on page 11-57). After the packet data has been transferred, an end of packet is signalled by pulling both UDC+ and UDC- low for 2-bit times, followed by an idle for 1-bit time. If the idle persists for more than 3 ms, the UDC enters suspend mode and it is placed in low-power mode. The UDC can be awakened from the suspend state by the host by switching the bus to the resume state via normal bus activity, or by signalling a reset. Under normal operating conditions, the host ensures that devices do not enter the suspend state by periodically signalling an end of packet (EOP).

11.8.1.2 Bit Encoding

USB uses nonreturn to zero inverted (NRZI) to encode individual bits. Both the clock and the data are encoded and transmitted within the same signal. Instead of representing data by controlling the state of the signal, transitions are used. A zero is represented by a transition, and a one is represented by no transition (this produces the data). Each time a zero occurs, the receiver logic synchronized the baud clock to the incoming data (this produces the clock). To ensure the receiver is periodically synchronized, any time six consecutive ones are detected in the serial bit stream, a zero is automatically inserted by the transmitter. This procedure is known as “bit stuffing”. The receiver logic, in turn, automatically detects stuffed bits and removes them from the incoming data. Bit stuffing causes a transition on the incoming signal at least once every seven bit-times to guarantee baud clock lock. Bit stuffing is enabled for an entire packet beginning when the start of packet is detected until the end of packet is detected (enabled during the sync field all the way through the CRC field). Figure 11-15 shows the NRZI encoding of the data byte 0b1101 0010.

Figure 11-15. NRZI Bit Encoding Example



11.8.1.3 Field Formats

Individual bits are assembled into groups called fields. Fields are used to construct packets and packets are used to construct frames or transactions. The seven USB field types include: sync, packet identifier, address, endpoint, frame number, data, and CRC fields.

A sync is preceded by the idle state on the USB bus and is always the first field of every packet. The first bit of a sync field signals the start of packet (SOP) to the UDC or host. A sync is 8 bits wide and consists of seven zeros followed by a one (0x80).

The packet identifier (PID) is 1 byte wide and always follows the sync field. The first 4 bits contain an encoded value that represents packet type (token, data, handshake, special), packet format, and type of error detection. The last four bits contain a check field that ensures the PID is transmitted without errors. The check field is generated by performing a ones complement of the PID. The UDC automatically XORs the PID and check field and takes the appropriate action (as prescribed by the USB standard) if the result does not contain all ones, indicating an error has occurred in transmission.

The UDC's three endpoints are accessed using the address and endpoint fields. The address field contains 7 bits and permits 128 unique devices to be placed on the USB. After the SA-1100 is reset, or a reset is signalled via the USB bus, the UDC (and all other 127 possible devices) is assigned the default address of zero. The host is then responsible for assigning unique addresses for each device on the bus. This is performed in the enumeration process one device at a time. Once the host assigns the UDC an address, it responds only to transactions addressed to it. The address field is transmitted in every packet and follows the PID field.

When the UDC detects that a packet is addressed to it, the endpoint field is used to determine which of the UDC's three endpoints are being addressed. The endpoint field is 4 bits. However, only the encodings for endpoints 0 through 2 are allowed. The endpoint field follows the address field. Table 11-11 shows the valid values for the endpoint field.

Table 11-11. Endpoint Field Addressing

Endpoint Field Value	UDC Endpoint Selected
0000	Endpoint 0
0001	Endpoint 1
0010	Endpoint 2
0011	Invalid
01xx	Invalid
10xx	Invalid
11xx	Invalid

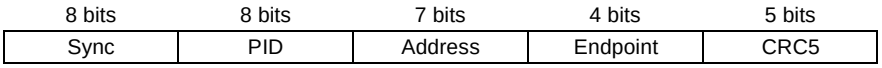
The frame number is an 11-bit field that is incremented by the host each time a frame is transmitted. When it reaches its maximum value of 2047 (0x7FF), it rolls over. It is transmitted in the start of frame (SOF) packet, which is output by the host in 1 ms intervals. The frame number field is used only by device controllers to control isochronous transfers, and therefore, does not affect the UDC. Data fields are used to transmit the bulk data between the host and the UDC. A data field is made up of 0 to 1023 bytes. Each byte is transmitted LSB first.

Cyclic redundancy check fields are used to detect errors introduced during transmission of token and data packets, and is applied to all the fields in the packet except the PID field (recall the PID contains its own 4-bit ones complement check field for error detection). Token packets use a 5-bit CRC (x^5+x^2+1) and data packets use a 16-bit CRC ($x^{16}+x^{15}+x^2+1$). For both CRCs, the checker is reset to all ones at the start of each packet.

11.8.1.4 Packet Formats

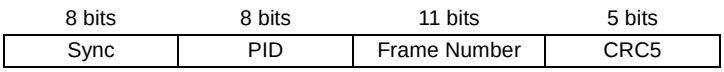
USB supports four packet types: token, data, handshake, and special. A token packet is placed at the beginning of a frame and is used to identify OUT, IN, SOF, and SETUP transactions. OUT and IN frames are used to transfer data, SOF packets are used to time isochronous transactions, and SETUP packets are used for control transfers to configure endpoints. A token packet consists of a sync, a PID, an address, an endpoint, and a CRC5 field (see Figure 11-16). For OUT and SETUP transactions, the address and endpoint fields are used to select which UDC endpoint is to receive the data, and for an IN transaction, which endpoint must transmit data.

Figure 11-16. IN, OUT, and SETUP Token Packet Format



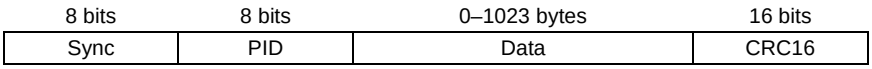
A start of frame (SOP) is a special type of token packet that is issued by the host once every 1 ms. SOF packets consist of a sync, a PID, a frame number (which is incremented after each frame is transmitted), and a CRC5 field, as shown in Figure 11-17. Even though the UDC on the SA-1100 does not make use of the frame number field, the presence of SOF packets every 1ms will prevent the UDC from going into suspend mode.

Figure 11-17. SOF Token Packet Format



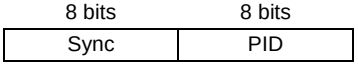
Data packets follow token packets, and are used to transmit data between the host and UDC. There are two types of data packets as specified by the PID: DATA0 and DATA1. These two types are used to provide a mechanism to guarantee data sequence synchronization between the transmitter and receiver across multiple transactions. During the handshake phase, both communicate and agree which data token type to transmit first. For each subsequent packet transmitted, the data packet type is toggled (DATA0, DATA1, DATA0, and so on). A data packet consists of a sync, a PID, from 0 to 1023 bytes of data, and a CRC16 field, as shown in Figure 11-18.

Figure 11-18. Data Packet Format



Handshake packets consist of only a sync and a PID. Handshake packets do not contain a CRC because the PID contains its own check field. They are used to report data transaction status, including whether data was successfully received, flow control, and stall conditions. Only transactions that support flow control can return handshakes. The three types of handshake packets are: ACK, NAK, and STALL. ACK indicates that a data packet was received without bit stuffing, CRC, or PID check errors. NAK indicates that the UDC was unable to accept data from the host or it has no data to transmit. NAK is also used by endpoint 1 to indicate no interrupts are pending. STALL indicates that the UDC is unable to transmit or receive data, and requires host intervention to clear the stall condition. Bit stuffing, CRC, and PID errors are signalled by the receiving unit by omitting a handshake packet. Figure 11-19 shows the format of a handshake packet.

Figure 11-19. Handshake Packet Format



11.8.1.5 Transaction Formats

Packets are assembled into groups to form transactions. Four different transaction formats are used in the USB protocol. Each is specific to a particular endpoint type: bulk, control, interrupt, and isochronous. Note that isochronous and interrupt transactions are not supported by the UDC and are not described in this section. Endpoint 0, by default, is a control endpoint and receives only control transactions; both endpoints 1 and 2 use bulk transactions. Note that all USB bus transactions are initiated by the host controller and that transmission takes place between the host and UDC one direction at a time (half-duplex).

Bulk transactions guarantee error-free transmission of data between the host and UDC by using packet error detection and retry. The three packet types used to construct bulk transactions are: token, data, and handshake. The eight possible types of bulk transactions based on data direction, error, and stall conditions are shown in Figure 11-20. Note that packets sent by the UDC to the host are highlighted in boldface type, and packets sent by the host to the UDC are not.

Figure 11-20. Bulk Transaction Formats

Action	Token Packet	Data Packet	Handshake Packet
Host successfully received data from UDC	IN	DATA0/DATA1	**ACK**
UDC temporarily unable to transmit data	IN	None	**NAK**
UDC endpoint needs host intervention	IN	None	**STALL**
Host detected PID, CRC, or bit stuff error	IN	DATA0/DATA1	None
UDC successfully received data from host	OUT	DATA0/DATA1	**ACK**
UDC temporarily unable to receive data	OUT	DATA0/DATA1	**NAK**
UDC endpoint needs host intervention	OUT	DATA0/DATA1	**STALL**
UDC detected PID, CRC, or bit stuff error	OUT	DATA0/DATA1	none

Packets from UDC to host are **boldface**

Control transactions are used by the host to configure endpoints and query their status. Like bulk transactions, control transactions begin with a setup packet, followed by an optional data packet, then a handshake packet. Note that control transactions, by default, use DATA0 type transfers. Figure 11-21 shows the four possible types of control transactions. Note that packets sent by the UDC to the host are highlighted in boldface type, and packets sent by the host to the UDC are not.

Figure 11-21. Control Transaction Formats

Action	Token Packet	Data Packet	Handshake Packet
UDC successfully received control from host	SETUP	DATA0	**ACK**
UDC temporarily unable to receive data	SETUP	DATA0	**NAK**
UDC endpoint needs host intervention	SETUP	DATA0	**STALL**
UDC detected PID, CRC, or bit stuff error	SETUP	DATA0	None

Packets from UDC to host are **boldface**

Control transfers are assembled by the host by first sending a control transaction to tell the UDC what type of control transfer is taking place (control read or control write), followed by two or more bulk data transactions. The control transaction, by default, uses a DATA0 transfer, and each subsequent bulk data transaction toggles between DATA1 and DATA0 transfers. For a control write to an endpoint, OUT transactions are used. For control reads, IN transactions are used. The transfer direction of the last bulk data transaction is reversed. It is used to report status and functions as a handshake. The last bulk data transaction always uses a DATA1 transfer by default (even if the previous bulk transaction used DATA1). For a control write, the last transaction is an IN from the UDC to the host, and for a control read, the last transaction is an OUT from the host to the UDC.

11.8.1.6 UDC Device Requests

The UDC's control, status, and data registers are used only to control and monitor the transmit and receive FIFOs for endpoints 1 and 2. All other UDC configuration and status reporting is controlled by the host via the USB bus using device requests that are sent as control transactions to endpoint 0. Each setup packet to endpoint 0 is 8 bytes long and specifies:

- Data transfer direction: host to device, device to host
- Data transfer type: standard, class, vendor
- Data recipient: device, interface, endpoint, other
- Number of bytes to transfer
- Index or offset
- Value: used to pass a variable-sized data parameter
- Device request

Table 11-12 shows a summary of all device requests. Users should refer to the *Universal Serial Bus Specification Revision 1.0* for a full description of host device requests.

Table 11-12. Host Device Request Summary

Request	Name
SET_FEATURE	Used to enable a specific feature such as device remote wake-up and endpoint stalls.
CLEAR_FEATURE	Used to clear or disable a specific feature.
SET_CONFIGURATION	Configures the UDC for operation. Used following a reset of the SA-1100 or after a reset has been signalled via the USB bus.
GET_CONFIGURATION	Returns the current UDC configuration to the host.
SET_DESCRIPTOR	Used to set existing descriptors or add new descriptors. Existing descriptors include: device, configuration, string, interface, and endpoint.
GET_DESCRIPTOR	Returns the specified descriptor if it exists.
SET_INTERFACE	Used to select an alternate setting for the UDC's interface.
GET_INTERFACE	Returns the selected alternate setting for the specified interface.
GET_STATUS	Returns the UDC's status including: remote wake-up, self-powered, data direction, endpoint number, and stall status.
SET_ADDRESS	Sets the UDC's 7-bit address value for all future device accesses.
SYNCH_FRAME	Used to set and then report an endpoint's synchronization frame.

11.8.2 UDC Register Definitions

All configuration, request/service, and status reporting is controlled by the USB host controller and is communicated to the UDC via the USB bus. Several registers are available to the programmer to control the interfacing of the UDC to software. A control register is used to enable the UDC and to mask the various interrupt sources that exist within the UDC. A status register is used to indicate the state of the various interrupt sources. The device address register is available, which software writes when parsing a SET_ADDRESS command from the USB host controller. There is a register for each of the OUT and IN endpoints' maximum packet size. All three endpoints (control, OUT, and IN) have a control/status register. Endpoint 0 (control) has an address for the 8 x 8 data FIFO used for both transmitting and receiving data, as well as a write count register used to determine how many bytes the USB host controller has sent to the endpoint 0. Both endpoints 1 and 2 (OUT and IN, respectively) share a data register address that contains an 8-bit field, which addresses the top of the transmit FIFO and bottom of the receive FIFO. When it is read, the receive FIFO is accessed, and when it is written, the transmit FIFO is accessed.

Note: Due to the internal synchronization required by the UDC's configuration registers, it is possible for the processor to write the UDC registers and FIFOs too fast. It is required that all writes to the UDC be complete before another write may take place. In order to guarantee that a write is complete, it is necessary to observe the effect of a write before another write may take place. For example, when clearing a bit, read it back until the bit is actually cleared before going on.

11.8.3 UDC Control Register

The UDC control register (UDCR) contains seven control bits: two to enable or disable the UDC and five to mask the transmit and receive FIFO service requests.

11.8.3.1 UDC Disable (UDD)

The UDC disable (UDD) bit is used to enable and disable the UDC. When UDD=0, the UDC is enabled for serial transmission or reception. When UDC=1, it is disabled and the UDC+ and UDC- pins are tristated.

If UDD is written to one the entire UDC design is reset. If this is done while the UDC is actively transmitting or receiving data, it stops immediately and the remaining bits within the transmit or receive serial shifter are reset. In addition, all entries within the transmit and receive FIFO ar reset.

11.8.3.2 UDC Active (UDA)

This read-only bit can be read to determine if the UDC is currently active. A one indicates that the UDC is currently involved in a transaction.

11.8.3.3 Bit 2 Reserved

Bit 2 is reserved and should always be written to a zero to ensure compatibility with future revisions of this design. This bit also will be set if the UDC detects that the data toggle mechanism did not occur.

11.8.3.4 Endpoint 0 Interrupt Mask (EIM)

The endpoint 0 interrupt mask (EIM) bit is used to mask or enable the endpoint 0 interrupt request. When EIM=1, the interrupt is masked and the EIR bit in the status/interrupt register is not allowed to be set. When EIM=0, the interrupt is enabled, and whenever an interruptible condition occurs in the receiver, the EIR bit is set. Note that programming EIM=1 does not affect the current state of EIR; it only blocks future zero to one transitions of EIR.

11.8.3.5 Receive Interrupt Mask (RIM)

The receive interrupt mask (RIM) bit is used to mask or enable the receive FIFO service request interrupt. When RIM=1, the interrupt is masked and the RIR bit in the status/interrupt register is not allowed to be set. When RIM=0, the interrupt is enabled, and whenever an interruptible condition occurs in the receiver, the RIR bit is set. Note that programming RIM=1 does not affect the current state of RIR; it only blocks future zero to one transitions of RIR.

11.8.3.6 Transmit Interrupt Mask (TIM)

The transmit interrupt mask (TIM) bit is used to mask or enable the transmit endpoint 2 interrupt request. When TIM=1, the interrupt is masked and the TIR bit in the status/interrupt register is not allowed to be set. When TIM=0, the interrupt is enabled, and whenever an interruptible condition occurs in the transmitter, the TIR bit is set. Note that programming TIM=1 does not affect the current state of TIR; it only blocks future zero to one transitions of TIR.

11.8.3.7 Suspend/Resume Interrupt Mask (SRM)

The suspend/resume interrupt mask (SRM) bit is used to mask or enable the suspend/resume interrupt request. When SRM=1, the interrupt is masked, and the SUSIR/RESIR bits in the status/interrupt register are not allowed to be set. When SRM=0, the interrupt is enabled, and whenever a suspend or resume condition occurs, the SUSIR or RESIR bit is set. Note that programming SRM=1 does not affect the current state of SUSIR/RESIR; it only blocks future zero to one transitions of SUSIR/RESIR.

11.8.3.8 Reset Interrupt Mask (REM)

The reset interrupt mask (REM) bit is used to mask or enable the reset interrupt request. When REM=1, the interrupt is masked, and the RSTIR bit in the status/interrupt register is not allowed to be set. When REM=0, the interrupt is enabled, and whenever the USB host controller issues a reset to the UDC, the RSTIR bit is set. Note that programming REM=1 does not affect the current state of RSTIR; it only blocks future zero to one transitions of RSTIR.

The following table shows the location of the UDE, RIM, and TIM bits in UDC control register (UDCR). The state of RIM and TIM are unknown and must be initialized before enabling the UDC. The UDE bit is cleared to zero, disabling the UDC following a reset of the SA-1100. This gives control of the UDC's pins to the PPC unit that configures them as inputs. Writes to reserved bits are ignored and reads return zeros.

Address: 0h 8000 0000 UDCCR Read/Write & Read Only

Bit	7	6	5	4	3	2	1	0
	REM	SRM	TIM	RIM	EIM	Res.	UDA	UDD
Reset	0	1	0	0	0	0	0	1

Bit	Name	Description
0	UDD	UDD disable. 0 – UDD disabled. 1 – UDD enabled, UDC+ and UDC- used for USB serial transmission/reception.
1	UDA	UDC active (read-only). 0 – UDC currently inactive. 1 – UDC currently active.
2	—	Reserved.
3	EIM	Endpoint zero interrupt mask. 0 – Endpoint zero interrupt enabled. 1 – Endpoint zero interrupt disabled.
4	RIM	Receive interrupt mask. 0 – Receive interrupt enabled. 1 – Receive interrupt disabled.
5	TIM	Transmit interrupt mask. 0 – Transmit interrupt enabled. 1 – Transmit interrupt disabled.
6	SRM	Suspend/resume interrupt mask. 0 – Suspend/resume interrupt enabled. 1 – Suspend/resume interrupt disabled.
7	REM	Reset interrupt mask. 0 – Reset interrupt enabled. 1 – Reset interrupt disabled.

11.8.4 UDC Address Register

The UDC address register contains a 7-bit field that holds the device address. After a reset of the UDC core, the value of this register is zero. The CPU writes an address to this register when it receives a SET_ADDRESS from the USB host controller. It extracts the address assigned to the UDC from the SET_ADDRESS command and writes the value into the UDC address register. The new address is not propagated to the rest of the UDC core until the SET_ADDRESS command is completed with an acknowledged handshake from the UDC.

Address: 0h 8000 0004 UDCAR Read/Write

Bit	7	6	5	4	3	2	1	0
	Res	7-bit Function Address						
Reset	0	0	0	0	0	0	0	0

Bit	Name	Description
7	—	Reserved. Always read zero.
6..0	Address	Function address field 7-bit function address. Reset to all zero.

11.8.5 UDC OUT Max Packet Register

The UDC OUT max packet register holds the value of the maximum packet size the UDC core will accept minus one. This is done in order to accommodate maximum packets of 256 bytes, without going to a max packet field of more than 8 bits. In order to accept packets up to 256 bytes, a value of 0xff (255) should be written into the OUT max packet register. At reset the OUT max packet register contains 0x08, and will therefore accept packets of length 9 bits or less.

Address: 0h 8000 0008 UDCOMP Read/Write

Bit	7	6	5	4	3	2	1	0
	Max Packet Size - 1							
Reset	0	0	0	0	1	0	0	0

Bit	Name	Description
7..0	OUT MaxP	OUT max packet size. 8-bit field containing the value of the maximum packet size minus one.

11.8.6 UDC IN Max Packet Register

The UDC IN max packet register holds the value of the number of bytes the UDC core is to transmit minus one. This is done in order to accommodate maximum packets of 256 bytes, without going to a max packet field of more than 8 bits. In order to transmit packets of 256 bytes, a value of 0xff (255) should be written into the IN max packet register. At reset the IN max packet register contains 0x08, and will therefore transmit packets of length 9 bits.

Address: 0h 8000 000C **UDCIMP** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	Max Packet Size - 1							
Reset	0	0	0	0	1	0	0	0

Bit	Name	Description
7..0	IN MaxP	IN max packet size. 8-bit field containing the value of the number of bytes to transmit minus one.

11.8.7 UDC Endpoint 0 Control/Status Register

The UDC endpoint zero control/status register contains 8 bits that are used to operate endpoint zero (control endpoint).

11.8.7.1 OUT Packet Ready (OPR)

The OUT packet ready bit is set by the UDC when it receives a valid token to endpoint zero. When this bit is set, the EIR bit will be set in the UDC status/interrupt register if endpoint zero interrupts are enabled. This bit is cleared by writing a one to the serviced out packet ready bit (6). The UDC is not allowed to enter the data phase of a transaction until this bit is cleared. If there is no data phase, then the CPU should set the data end bit (4) at the same time it clears this bit.

11.8.7.2 IN Packet Ready (IPR)

The IN packet ready bit is set by the CPU after it has written a packet to the endpoint zero FIFO to be transmitted. The UDC will automatically clear this bit when the packet has been successfully transmitted. When this bit is cleared, the EIR bit in the UDC status/interrupt register will be set if endpoint zero interrupts are enabled. The CPU will not be able to clear this bit.

11.8.7.3 Sent Stall (SST)

The sent stall bit is set by the UDC when it must abort the current control transfer by issuing a STALL handshake due to a protocol violation. When this bit is set, the EIR bit in the UDC status/interrupt register will be set if endpoint zero interrupts are enabled. The CPU clears this bit by writing a one to it.

11.8.7.4 Force Stall (FST)

The force stall bit can be set by the UDC to force the UDC to issue a STALL handshake. The UDC issues a STALL handshake for the current setup control transfer and the bit is cleared by the UDC because endpoint zero cannot remain in a stalled condition.

11.8.7.5 Data End (DE)

The data end bit is set by the UDC after it writes the last packet for the current descriptor. Once the current setup transfer has ended, the UDC clears this bit. When this bit is cleared the EIR bit in the UDC status/interrupt register will be set if endpoint zero interrupts are enabled. If there is no data phase, the CPU should set this bit at the same time it clears the OPR bit (0).

11.8.7.6 Setup End (SE)

The setup end bit is set by the UDC when a control transfer ends before the DE bit (4) gets set. When this bit is set the EIR bit in the UDC status/interrupt register will be set if endpoint zero interrupts are enabled. This bit is cleared by writing a one to the serviced setup end bit (7). When the CPU detects this bit being set (if the OPR bit (0) is also set), then it should unload the new setup packet after it clears setup end.

11.8.7.7 Serviced OPR (SO)

The serviced bit will clear the OPR bit (0) when writing a one.

11.8.7.8 Serviced Setup End (SSE)

The serviced setup end bit will clear the SE bit (5) when writing a one.

Address: 0h 8000 0010 **UDCCS0** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	SSE	SO	SE	DE	FST	SST	IPR	OPR
Reset	0	0	0	0	0	0	0	0

Bit	Name	Description
0	OPR	OUT packet ready (read-only). 1 – OUT packet ready.
1	IPR	IN packet ready (read/write 1 to set). 1 – IN packet ready.
2	SST	Sent stall (read/write 1 to clear). 1 – UDC sent stall handshake.
3	FST	Force stall (read/write 1 to set). 1 – Force stall handshake.
4	DE	Data end (read/write 1 to set). 1 – The last byte of the data phase has been written.
5	SE	Setup end (read-only). 1 – Control transfer ended before data end got set.
6	SO	Serviced OPR (write-only). 1 – Clear OPR, bit 0.
7	SSE	Serviced setup end (write-only). 1 – Clear SE, bit 5.

11.8.8 UDC Endpoint 1 Control/Status Register

The UDC endpoint 1 control/status register contains 6 bits that are used to operate endpoint 1 (OUT endpoint).

11.8.8.1 Receive FIFO Service (RFS)

The receive FIFO service bit will be set if the receive FIFO has between 8 and 12 or more bytes (out of 20) in it. Because the FIFOs are asynchronous, the exact threshold cannot be determined, but is guaranteed to be in this range. This signal is also used as a DMA request signal to trigger the DMA unit to service the FIFO.

11.8.8.2 Receive Packet Complete (RPC)

The receive packet complete bit gets set by the UDC when an OUT packet has been received. When this bit is set the RIR bit in the UDC status/interrupt register will be set if receive interrupts are enabled. This bit can be used to validate the other status/error bits in the endpoint 1 control/status register. The RPC bit gets cleared by writing a one to it. The UDC will issue NAK handshakes to all OUT tokens while this bit is set.

11.8.8.3 Receive Packet Error (RPE)

The receive packet error bit will be set if a CRC, bit stuffing, or FIFO overrun error occurs. It is only valid if the RPC bit (1) is set and gets cleared when the RPC bit gets cleared.

11.8.8.4 Sent Stall (SST)

The sent stall bit is set by the UDC when it must abort the current transfer by issuing a STALL handshake due to a protocol violation (the host sends more data than the maximum packet size). The CPU clears this bit by writing a one to it.

11.8.8.5 Force Stall (FST)

The force stall bit can be set by the UDC to force the UDC to issue a STALL handshake to all OUT tokens. STALL handshakes will continue to be sent until the CPU clears this bit. The sent stall bit (3) will be set when the STALL state is actually entered (this may be delayed if the UDC is active when the FST bit is set), and the STALL state will not be exited until both the FST and SST bits are cleared.

11.8.8.6 Receive FIFO Not Empty (RNE)

The receive FIFO not empty bit indicates that there is unread data in the receive FIFO. This bit must be polled when the RPC bit is set to determine if there is any data in the FIFO that DMA did not read. The receive FIFO must continue to be read until this bit clears or data will be lost.

11.8.8.7 Bits 7..6 Reserved

Bits 7..6 are reserved for future use.

Address: 0h 8000 0014 **UDCCS1** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	Res.		RNE	FST	SST	RPE	RPC	RFS
Reset	0	0	0	0	0	0	0	0

Bit	Name	Description
0	RFS	Receive FIFO service (read-only). 0 – Receive FIFO has less than 12 bytes. 1 – Receive FIFO has 12 bytes or more.
1	RPC	Receive packet complete (read/write 1 to clear). 0 – Error/status bits invalid. 1 – Receive packet has been received and error/status bits are valid.
2	RPE	Receive packet error (read-only). 0 – Receive packet has no errors. 1 – Receive packet has errors; valid only when RPC is set.
3	SST	Sent stall (read/write 1 to clear). 1 – STALL handshake was sent; valid only when RPC is set.
4	FST	Force stall (read/write). 1 – Issue STALL handshakes to OUT tokens.
5	RNE	Receive FIFO not empty (read-only). 0 – Receive FIFO empty. 1 – Receive FIFO not empty.
7..6	—	Reserved. Always reads zero.

11.8.9 UDC Endpoint 2 Control/Status Register

The UDC endpoint 2 control status register contains 6 bits that are used to operate endpoint 2 (IN endpoint).

11.8.9.1 Transmit FIFO Service (TFS)

The transmit FIFO service bit will be active if there are 8 or less (out of 16) bytes remaining in the transmit FIFO. This bit will be used as a DMA request to trigger the DMA unit to service the transmit FIFO.

11.8.9.2 Transmit Packet Complete (TPC)

The transmit packet complete bit will be set by the UDC when an entire packet has been sent to the host. When this bit is set, the TIR bit in the UDC status/interrupt register will be set if transmit interrupts are enabled. This bit can be used to validate the other status/error bits in the endpoint 2 control/status register. The TPC bit gets cleared by writing a one to it. The UDC will issue NAK handshakes to all IN tokens while this bit is set.

11.8.9.3 Transmit Packet Error (TPE)

The transmit packet error bit acts as a status bit and will be valid while TPC is set. The TPE bit being set will indicate that the host did not issue an ACK handshake to the current packet. The TPE bit will be cleared when the TPC bit is cleared.

11.8.9.4 Transmit Underrun (TUR)

The transmit underrun bit will be set if the transmit FIFO experiences an underrun. This bit will be valid when the TPC bit is set. When the UDC experiences an underrun, the packet is shortened and the CRC is corrupted to ensure that the host discards the packet. The TUR bit will be cleared when the TPC bit is cleared.

11.8.9.5 Sent STALL (SST)

The sent stall bit indicates that a STALL handshake was issued to the host. The CPU writes a one to this bit to clear it. When this bit is cleared the transmit FIFO is flushed.

11.8.9.6 Force STALL (FST)

The CPU can set the force stall bit to force the UDC to issue a STALL handshake to all IN tokens. STALL handshakes will continue to be sent until the CPU clears this bit. The sent stall bit (4) will be set when the STALL state is actually entered (this may be delayed if the UDC is active when the FST bit is set), and the STALL state will not be exited until both the FST and SST bits are cleared.

11.8.9.7 Bits 7..6 Reserved

Bits 7..6 are reserved for future use.

Address: 0h 8000 0018 **UDCCS2** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	Res.		FST	SST	TUR	TPE	TPC	TFS
Reset	0	0	0	0	0	0	0	0

Bit	Name	Description
0	TFS	Transmit FIFO service (read-only). 0 – Transmit FIFO has more than 8 bytes. 1 – Transmit FIFO has 8 bytes or less.
1	TPC	Transmit packet complete (read/write 1 to clear). 0 – Error/status bits invalid. 1 – Transmit packet has been sent and error/status bits are valid.
2	TPE	Transmit packet error (read-only). 0 – Transmit packet was received with no errors. 1 – Transmit packet has errors and the host did not issue ACK. Valid only when RPC is set.
3	TUR	Transmit FIFO underrun. 1 – Transmit FIFO experienced an underrun. Valid only when TPC is set.
4	SST	Sent STALL (read/write 1 to clear). 1 – STALL handshake was sent. Valid only when TPC is set.
5	FST	Force STALL (read/write). 1 – Issue STALL handshakes to IN tokens.
7..6	—	Reserved. Always reads zero.

11.8.10 UDC Endpoint 0 Data Register

The UDC endpoint 0 data register is actually an 8-bit x 8-entry bidirectional FIFO. When the host transmits data to the UDC endpoint 0, the CPU reads the UDC endpoint 0 register to access the data. When the UDC is sending data to the host, the CPU writes the data to be sent into the UDC endpoint 0 register. Although the same FIFO can be read and written by the CPU during various points in a control sequence, the CPU may not read and write the FIFO at the same time. The direction that the FIFO is flowing is controlled by the UDC. Normally, the UDC will be in an idle state, waiting for the host to send commands. When this happens, the UDC fills the FIFO with the command from the host and the CPU reads the command from the FIFO once it has arrived. The UDC will do a partial decode of the command to determine if the CPU is going to be filling the FIFO with data to send to the host. If so, the direction is turned around to accept data from the CPU and have the UDC transmit the data. If the command is such that no data will be required from the UDC, then this will not happen. The only time the CPU may write the endpoint 0 FIFO is when a valid command from the host has been received which requires a transmission in response, that is, a GET_DESCRIPTOR command.

Address: 0h 8000 001C **UDCD0** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	Bottom of Endpoint 0 FIFO							
Reset	0	0	0	0	0	0	0	0

Read Access

Bit	7	6	5	4	3	2	1	0
	Top of Endpoint 0 FIFO							
Reset	0	0	0	0	0	0	0	0

Write Access

Bit	Name	Description
7..0	DATA	Top/bottom of endpoint 0 FIFO data. Read – Bottom of endpoint 0 FIFO data. Write – Top of endpoint 0 FIFO data.

11.8.11 UDC Endpoint 0 Write Count Register

The UDC endpoint 0 write count register can be read when a packet has been received by the endpoint 0 to determine how many bytes to read out of the UDC endpoint 0 data register. When data is present in the FIFO, this 4-bit field should read between 1 and 8.

Address: 0h 8000 0020 **UDCWC** **Read-Only**

Bit	7	6	5	4	3	2	1	0
	Reserved				Write Count			
Reset	0	0	0	0	0	0	0	0

Bit	Name	Description
3..0	WC	Endpoint 0 write count (read-only). 4-bit field representing the number of bytes in the endpoint 0 FIFO.
7..4	—	Reserved. Always reads zero.

11.8.12 UDC Data Register

The UDC data register (UDDR) is an 8-bit register corresponding to both the top and bottom entries of the transmit and receive FIFOs, respectively. Data is placed by the UDC's receive logic into the top of the receive FIFO. The data is transferred down the FIFO to the lowest location that is empty. When UDDR is read, the bottom entry of the 8-bit receive FIFO is accessed. After the read, the bottom FIFO entry is invalidated, which causes all data in the FIFO to automatically transfer down one location.

When UDDR is written, the topmost FIFO entry of the 8-bit transmit FIFO is accessed. After a write, the data is automatically transferred down the FIFO to the lowest location that is empty. The UDC's transmit logic takes 8-bit values from the bottom of the transmit FIFO one at a time, places the data into a serial shifter, and transmits the value out onto the UDC pins. Each time a value is taken from the bottom entry, the location is invalidated, which causes all data in the FIFO to automatically transfer down one location.

The following table shows the location of the top/bottom of the transmit/receive FIFOs in the UDC data register (UDDR). Note that both FIFOs are cleared when the SA-1100 is reset and when UDE is written to zero. After either of these actions takes place, the user may prime the transmit FIFO by writing up to sixteen 8-bit values to UDDR before enabling the UDC.

Address: 0h 8000 0008 **UDDR** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	Bottom of receive FIFO							
Reset	0	0	0	0	0	0	0	0

Read Access

Bit	7	6	5	4	3	2	1	0
	Top of transmit FIFO							
Reset	0	0	0	0	0	0	0	0

Write Access

Bit	Name	Description
7..0	DATA	Top/bottom of transmit/receive FIFO data. Read – Bottom of receive FIFO data. Write – Top of transmit FIFO data.

11.8.13 UDC Status/Interrupt Register

The UDC status/interrupt register (UDCSR) contains bits that are used to generate the UDC's interrupt request. Each bit in the UDC status/interrupt register is logically ORed together to produce one interrupt request. When the ISR for the UDC is executed, it must read the UDC status/interrupt register to determine why the interrupt occurred.

Every bit in the UDCSR is controlled by a mask bit in the UDC control register. The mask bits, when set, will prevent a status bit in the UDCSR from being set. If the mask bit for a particular status bit is cleared and an interruptible condition occurs, the status bit will be set. In order to clear status bits, the CPU must write a one into the position that it wishes to clear. The interrupt request for the UDC will remain active as long as the value of the UDCSR is non-zero.

11.8.13.1 Endpoint 0 Interrupt Request (EIR)

The endpoint 0 interrupt request will be set if the EIM bit in the UDC control register is cleared, and in the UDC endpoint 0 control/status register, the OUT packet ready bit gets set, the IN packet ready bit gets cleared, the data end bit gets cleared, the setup end bit gets set, or the sent STALL bit gets set. The EIR bit is cleared by writing a one to it.

11.8.13.2 Receive Interrupt Request (RIR)

The receive interrupt request bit gets set if the RIM bit in the UDC control register is cleared and the Receive Packet Complete bit in the UDC endpoint 1 control/status register gets set. The RIR bit is cleared by writing a one to it.

11.8.13.3 Transmit Interrupt Request (TIR)

The transmit interrupt request bit gets set if the TIM bit in the UDC control register is cleared and the Transmit Packet Complete bit in the UDC endpoint 2 control/status register gets set. The RIR bit is cleared by writing a one to it.

11.8.13.4 Suspend Interrupt Request (SUSIR)

The suspend interrupt request bit will be set if the SRM bit in the UDC control register is cleared and the USB bus remains idle for more than 3 ms. The SUSIR bit gets cleared by writing a one to it.

11.8.13.5 Resume Interrupt Request (RESIR)

The resume interrupt request bit will be set if the SRM bit in the UDC control register is cleared, the UDC is currently in the suspended state, and the USB bus is driven with resume signalling.

11.8.13.6 Reset Interrupt Request (RSTIR)

The reset interrupt request register will be set if the REM bit in the UDC control register is cleared and the host issues a reset. When the host issues a reset, the entire UDC is reset. The RSTIR bit retains its state so software can determine that the design was reset.

Address: 0h 8000 0030 **UDCSR** **Read/Write (Clear)**

Bit	7	6	5	4	3	2	1	0
	Res.		RSTIR	RESIR	SUSIR	TIR	RIR	EIR
Reset	0	0	0	0	0	0	0	0

Bit	Name	Description
0	EIR	Endpoint 0 interrupt request (read/write clear). 1 – Endpoint 0 needs service.
1	RIR	Receive interrupt request (read/write clear). 1 – Receive endpoint (1) needs service.
2	TIR	Transmit interrupt request (read/write clear). 1 – Transmit endpoint (2) needs service.
3	SUSIR	Suspend interrupt request (read/write clear). 1 – UDC received suspend signalling from the host.
4	RESIR	Resume interrupt request (read/write clear). 1 – UDC received resume signalling from the host.
5	RSTIR	Reset interrupt request (read/write clear). 1 – UDC was reset by the host.
7..6	—	Reserved. Always reads zero.

11.8.14 UDC Register Locations

Table 11-13 shows the registers associated with the UDC and the physical addresses used to access them.

Table 11-13. UDC Control, Data, and Status Register Locations

Address	Name	Description
0h8000 0000	UDCCR	UDC control register
0h8000 0004	UDCAR	UDC address register
0h8000 0008	UDCOMP	UDC OUT max packet register
0h8000 000C	UDCIMP	UDC IN max packet register
0h8000 0010	UDCCS0	UDC endpoint 0 control/status register
0h8000 0014	UDCCS1	UDC endpoint 1 (OUT) control/status register
0h8000 0018	UDCCS2	UDC endpoint 2 (IN) control/status register
0h8000 001c	UDCD0	UDC endpoint 0 data register
0h8000 0020	UDCWC	UDC endpoint 0 write count register
0h8000 0024	—	Reserved
0h8000 0028	UDCDR	UDC transmit/receive data register (FIFOs)
0h8000 002c	—	Reserved
0h8000 0030	UDCSR	UDC status/interrupt register

11.9 Serial Port 1 – SDLC/UART

Serial port 1 is a combination synchronous data link controller (SDLC) and universal asynchronous receiver/transmitter (UART) serial controller. The user can configure it to perform one of the two functions, but operation of both modes using serial port 1's pins cannot occur simultaneously (SDLC transmit and UART receive). However, the peripheral pin control (PPC) unit can be configured to take control of two GPIO pins and use them for UART transmission, while serial port 1's pins are used for SDLC operation. See the Section 11.13, "Peripheral Pin Controller (PPC)" on page 11-184 for a description of how the PPC is configured to allow use of both the SDLC and UART.

For both protocols, serial port 1 can operate at baud rates from 56.24 bps to 230.4 Kbps. Both also contain an 11-bit wide by 12-entry deep receive FIFO and an 8-bit wide by 8-entry deep transmit FIFO to buffer incoming and outgoing data, respectively. The FIFOs can be filled or emptied either by the DMA or the CPU, with service requests being signalled when the transmit FIFO is half-empty and the receive FIFO is one- to two-thirds full.

Used as an SDLC controller, serial port 1 supports much of the functionality found in commercial serial communications controllers, such as the 85C30. Frames contain an 8-bit address, an optional control field, a data field of any size that is a multiple of 8 bits, and a 16-bit CRC-CCITT. The start and stop flags and CRC generation and checking are handled automatically. Data can be selectively saved in the receive FIFO by programming an address with which to compare against all incoming frames. Interrupts are signalled when CRC checks performed on received data indicate an error, when a receiver abort occurs, when the transmit or receive FIFO needs to be filled or emptied, when the transmit FIFO underruns during an active frame and is aborted, when the receive FIFO overruns and data is lost, and when the last byte of data within a frame is contained within the bottom four entries of the receive FIFO.

Used as a UART, serial port 1 is identical to serial port 3. It supports most of the functionality of the 16C550 protocol including 7 and 8 bits of data (odd, even, or no parity), one start bit, either one or two stop bits, and transmits a continuous break signal. An interrupt is generated when a framing, parity, or receiver overrun error is present within the bottom four entries of the receive FIFO, when the transmit FIFO is half-empty or the receive FIFO is one- to two-thirds full, when a begin and end of break is detected on the receiver, and when the receive FIFO is partially full and the receiver is idle for three or more frame periods. Because programming and operation of serial port 1 as a UART is identical to serial port 3, see the Section 11.9, "Serial Port 1 – SDLC/UART" on page 11-78 for a complete description of using serial port 1 in UART mode.

The external pins dedicated to this interface are TXD1 and RXD1. If serial transmission is not required and both the SDLC and UART are disabled, control of these pins is given to the peripheral pin control (PPC) unit for use as general- purpose input/output pins (noninterruptible). See the section 11.13 on page 184.

Modem control signals (RTS, CTS, DTR, and DSR) are not provided in this block but can be implemented using the general-purpose I/O port (GPIO) pins described in the Chapter 9, "System Control Module".

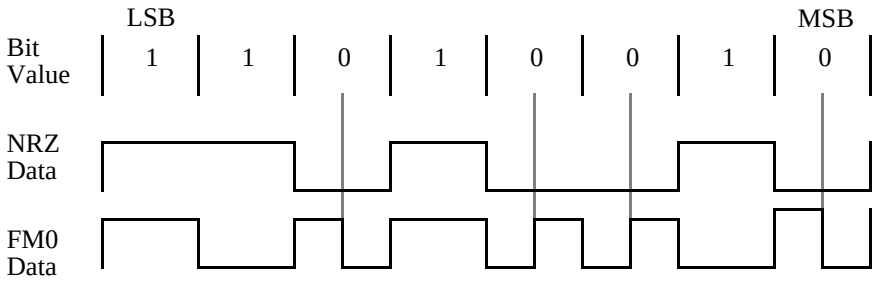
11.9.1 SDLC Operation

Following reset, both the SDLC and UART are disabled, which causes the peripheral pin controller (PPC) to assume control of the port's pins. Reset causes the PPC to configure all of the peripheral pins as inputs, including serial port 1's transmit (**TXD1**) and receive (**RXD1**) pins. Reset also causes the SDLC's transmit and receive FIFOs to be flushed (all entries invalidated). Before enabling the SDLC, the user must first clear any writable or "sticky" status bits that are set by writing a one to each bit. Next, the desired mode of operation is programmed in the control registers. At this point, the user can "prime" the transmit FIFO by writing up to eight values, or the FIFO can remain empty and either programmed I/O or the DMA can be used to service it after the SDLC is enabled. Once the SDLC is enabled, transmission and reception of data can begin on the transmit (**TXD1**) and receive (**RXD1**) pins.

11.9.1.1 Bit Encoding

SDLC uses frequency modulation zero (FM0) to encode individual bits. Both the clock and the data are encoded and transmitted on the same line. Instead of representing data by controlling the state of the line, its frequency is used. The line transitions at a frequency that represents the serial stream's bit rate (this produces the clock). Individual bits are separated by each transition. A zero is encoded by placing an extra transition at the middle of its bit period. A one is represented by no added transitions within its bit period (this produces the data). Note that nonreturn to zero (NRZ) bit encoding can also be programmed in the SDLC. In NRZ encoding, a one is represented when the line transitions, and a zero when the line does not transition. Figure 11-22 shows both the NRZ and FM0 encoding of the data byte 8b 0100 1011. Note that the byte's LSB is transmitted first.

Figure 11-22. FM0/NRZ Bit Encoding Example (0100 1011)



11.9.1.2 Frame Format

SDLC uses a flag (reserved bit pattern) to denote the beginning of a frame of information and to synchronize frame transmission. The flag contains eight bits that start and end with a zero, and contains six sequential ones in the middle (01111110). This sequence of six ones is unique because all data between the start and stop flags is prohibited from having more than five consecutive ones. Data that violates this rule is altered before transmission by automatically inserting a zero after five consecutive ones are detected in the transmitted bit stream. This technique is commonly referred to as “bit stuffing” and is transparent to the user. The information field within an SDLC frame is placed between two flags and consists of an 8-bit address, an optional 8-bit control field, a data field containing any multiple of 8 bits, and a 16-bit cyclic redundancy check (CRC-CCITT). The user can also program the SDLC to insert an optional second start flag. Note that each byte within the address, control, and data fields is transmitted and received LSB first, ending with the byte’s MSB. However, the CRC is transmitted and received MSB first. Figure 11-23 shows the SDLC frame format.

Figure 11-23. SDLC Frame Format

8 Bits (optional)	8 Bits	8 Bits	8 Bits (optional)	Any Multiple of 8 Bits	16 Bits	8 Bits
Start Flag 0111 1110	Start Flag 0111 1110	Address	Control	Data	CRC-CCITT	Stop Flag 0111 1110

11.9.1.3 Address Field

The 8-bit address field is used by a transmitter to target a select group of receivers when multiple stations are connected to the same set of serial lines. The address allows up to 255 stations to be uniquely addressed (00000000 to 11111110). The global address (11111111) is used to broadcast messages to all stations. Serial port 1 contains an 8-bit register that is used to program a unique address for broadcast recognition. It also contains a control bit to enable or disable the address match function. Note that the address of received frames is stored in the receive FIFO along with normal data; it is transmitted and received starting with its LSB and ending with its MSB.

11.9.1.4 Control Field

The SDLC control field is typically 8 bits, but can be any length. Serial port 1 does not provide any hardware decode support for the control byte; it treats all bytes between the address and the CRC as data. Note that the control field is transmitted and received starting with its LSB and ending with its MSB.

11.9.1.5 Data Field

The data field can be any length that is a multiple of 8 bits, including zero. The user determines the data field length according to the application requirements and transmission characteristics of the target system. Usually a length is selected that maximizes the amount of data that can be transmitted per frame to allow the CRC checker to consistently detect all errors during transmission. Note that serial port 1 does not support residue coding found in common SCCs; all data fields must be a multiple of 8 bits. If a data field that is not a multiple of 8 bits is received, an abort is signalled and the end of frame tag is set within the receive FIFO. Also note that each byte within the data field is transmitted and received starting with its LSB and ending with its MSB.

11.9.1.6 CRC Field

SDLC uses the established CCITT cyclic redundancy check (CRC) to detect bit errors that occur during transmission. A 16-bit CRC-CCITT is computed using the address, control, and data fields, and is included in each frame. A separate CRC generator is implemented in both the transmit and receive logic. The transmitter calculates a CRC while data is actively transmitted, and places the 16-bit value at the end of each frame before the flag is transmitted. The receiver calculates a CRC for each received data frame, and compares the calculated CRC to the expected CRC value contained within the end of each received frame. If the calculated value does not match the expected value, an interrupt is signalled. The CRC computation logic is preset to all ones before reception or transmission of each frame. Note that, unlike all other fields within the frame, the CRC is transmitted and received starting with its MSB and ending with its LSB. The CRC logic uses the following four-term polynomial in the implementation of its linear feedback shift register.

$$CRC(x) = (X^{16} + X^{12} + X^{5} + 1)$$

11.9.1.7 Baud Rate Generation

The baud or bit rate is derived by dividing down the 3.6864-MHz clock generated by the on-chip PLL. The clock is first divided by a programmable number between 1 and 4096, and then by a fixed value of 16. The receive baud clock is synchronized with the data steam each time a transition is detected on the receive data line at a bit's boundary. For FM0 encoding, zeros and ones are decoded within the incoming data stream by detecting whether a transition occurs between the boundaries of a bit time. If the receive line transitions, a zero is decoded; otherwise, a one is decoded. The baud synchronizer differentiates a transition of the receive line at the bit boundary from a transition caused by a zero by first establishing the bit boundary during reception of the string of ones within the flag (01111110). A counter is then used to cause the synchronizer to ignore transitions that occur during mid-bit. This is accomplished by using the clock produced before the fixed divide by 16 takes place. This clock is used to increment a counter that is reset at the boundary of each bit. Transitions that take place at any time before the counter reaches the value 12 (3/4 of a total bit time) are ignored. This function effectively masks a transition, which occurs during reception of a zero, excluding it from the bit synchronization process. When NRZ encoding is used, each bit of received data is sampled at its midpoint by using the clock that is generated before the fixed divide by 16 takes place. A sample rate counter is used that is reset at the boundary of each bit and is incremented using this clock. When it reaches a value of 8 (halfway through the bit period), the receive data pin is sampled.

11.9.1.8 Receive Operation

Once the SDLC receiver is enabled, it enters hunt mode, searching the incoming data stream for the flag (01111110). The flag serves to achieve bit synchronization, denotes the beginning of a frame, and delineates the boundaries of individual bytes of data. The end of the flag denotes the beginning of the address byte. Once the flag is found, the receiver is synchronized to incoming data and hunt mode is exited.

After each bit is decoded, a serial shifter is used to receive the incoming data a byte at a time. Once the flag is recognized, each subsequent byte of data is decoded and placed within a 2-byte temporary FIFO. A temporary FIFO is used to prevent the CRC from being placed within the receive FIFO. When the temporary FIFO is filled, data values are pushed out one by one to the receive FIFO. The first byte of a frame is the address. If receiver address matching is enabled, the received address is compared to the address programmed in the address match value field in a control register. If the two values are equal or if the incoming address contains all ones, all subsequent data bytes including the address byte are stored in the receive FIFO. If the values do not match, the receive logic does not store any data in the receive FIFO, ignores the remainder of the frame, and begins to search for the stop flag. The second byte of the frame can contain an optional control field, which must be decoded in software (no hardware support within the SDLC). Use of a control byte is determined by the user.

When the receive FIFO is one- to two-thirds full, an interrupt and/or DMA request is signalled. If the data is not removed soon enough, and the FIFO is completely filled, an overrun error is generated when the receive logic attempts to place additional data into the full FIFO. Once the FIFO is full, all subsequent data bytes received are lost while all FIFO contents remain intact.

Frames can contain any amount of data in multiples of 8 bits. Although the SDLC protocol does not limit frame size, in practice they tend to be implemented in numbers ranging from hundreds to thousands of bytes.

The receive logic continuously searches for the stop flag at the end of the frame. Once it is recognized, the last byte that was placed within the receive FIFO is flagged as the last byte of the frame, and the two bytes remaining within the temporary FIFO are removed and used as the 16-bit CRC value for the frame. Instead of placing this in the receive FIFO, the receive logic compares it to the CRC-CCITT value, which is continuously calculated using the incoming data stream. If they do not match, the last byte that was placed within the receive FIFO is also flagged with a CRC error. The CRC value is not placed in the receive FIFO.

The SDLC protocol permits back-to-back frames to be received. When this occurs, the flag at the end of the first frame also serves as the flag to denote the beginning of the next frame (only one flag separates the two). Most commercial SCCs continuously transmit flags between frames when they do not occur back-to-back. To support both of these cases, the receive logic allows one or more flags to separate frames. When the use of two start flags is programmed by the user, two flags always separate back-to-back frames that are transmitted.

Most commercial SCCs can generate an abort (7 to 13 ones) when their transmit FIFO underruns. The receive logic contains a counter that increments each time a one is decoded before entering the serial shifter and is reset any time a zero is decoded. When seven or more ones are detected, a receiver abort occurs. Note that data is moved from the serial shifter to the temporary FIFO a byte at a time, and seven consecutive ones may bridge two bytes. For this reason, after an abort is detected, the remaining data in the serial shifter is discarded along with the most recent byte of data placed in the temporary FIFO. After this data is discarded, the oldest byte of data in the temporary FIFO is placed in the receive FIFO, the EOF tag is set within the top entry of the FIFO (next to the byte transferred from the temporary FIFO), the receiver abort interrupt is signalled, and the receiver logic enters hunt mode until it recognizes the next flag.

If the user disables the receiver during operation, reception of the current data byte is stopped immediately, the serial shifter and receive FIFO are cleared, control of the **RXD1** pin is given to the peripheral pin control (PPC) unit, and all clocks used by the receive logic are automatically shut off to conserve power. However, the transmitter continues to function as normal.

11.9.1.9 Transmit Operation

The SDLC transmit logic can operate at the same time as the receive logic (full-duplex). The user may either "prime" the transmit FIFO by filling it with data or allow service requests to cause the CPU or DMA to fill the FIFO once the SDLC transmitter is enabled. Once enabled, the transmit logic issues a service request if its FIFO is empty. Flags are transmitted continuously until valid data resides within the FIFO. Once a byte of data resides at the bottom of the transmit FIFO, it is transferred to the serial shifter. It is encoded and shifted out onto the TXD1 pin clocked by the programmed baud rate clock. Note that the flag and CRC value are automatically transmitted and need not be placed in the transmit FIFO.

When the transmit FIFO is emptied halfway, an interrupt and/or DMA service request is signalled. If new data is not supplied soon enough, the FIFO is completely emptied and the transmit logic attempts to take additional data from the empty FIFO. The user can program one of two actions: an underrun to signal the normal completion of a frame or an unexpected termination of a frame in progress.

When normal frame completion is selected and an underrun occurs, the transmit logic transmits the 16-bit CRC value calculated during the transmission of all data within the frame (including the address and control bytes), followed by a flag to denote the end of the frame. The transmitter then continuously transmits flags until data is once again available within the FIFO. Once data is available, the transmitter begins transmission of the next frame.

When unexpected frame termination is selected and an underrun occurs, the transmit logic outputs an abort and interrupts the CPU. An abort continues to be transmitted until data is once again available in the transmit FIFO. The SDLC then transmits a flag and starts the new frame. The off-chip receiver can choose to ignore the abort and continue to receive data, or to signal serial port 1 to retry transmission of the aborted frame.

If the user disables the transmitter during operation, transmission of the current data byte is stopped immediately, the serial shifter and transmit FIFO are cleared, control of the TXD1 pin is given to the peripheral pin control (PPC) unit, and all clocks used by the transmit logic are automatically shut off to conserve power. However, the receiver continues to function as normal.

11.9.1.10 Simultaneous Use of the UART and SDLC

Serial port 1 contains a control bit to select which serial protocol to use: SDLC or UART. Note that the two protocols cannot be combined at the same time (SDLC transmit and UART receive). However, since the SDLC and UART are fully independent blocks, a mode is supported that allows the user to enable the SDLC using serial port 1's pins (TXD1 and RXD1) while the UART is enabled using two GPIO pins (GPIO<14> for transmit and GPIO<15> for receive operation). This mode is enabled by setting the UART pin reassignment (UPR) control bit within the peripheral pin controller (PPC). See the Section 11.13, "Peripheral Pin Controller (PPC)" on page 11-184. Note that when this mode is enabled, serial port 1's control bit, which selects SDLC versus UART operation, is ignored and serial port 1 defaults to SDLC mode.

11.9.1.11 Transmit and Receive FIFOs

To reduce chip size and power consumption, the SDLC's FIFOs use self-timed logic (they are not clocked). Because of process and environmental variations, the depth at which a service request is triggered to empty the receive FIFO is variable. This variation spans a maximum of four FIFO entries; the receive FIFO service request can be made at four different FIFO depths. To compensate for this variability and guarantee that at least four valid entries of data exist within the FIFO before generating a service request, an extra four entries have been added to the receive FIFO (four entries more than the transmit FIFO). The transmit FIFO is 8 entries deep and the receive FIFO is 12 entries deep. The point at which the receive FIFO service request is triggered spans the middle third of the 12-entry FIFO. The service request is signalled at a depth from one-third full to two-thirds full or when the FIFO contains five, six, seven, or eight entries of data.

This service request variation applies only to an empty FIFO that is filled (receive FIFO). It does not apply to a full FIFO that is emptied (transmit FIFO). The transmit FIFO is guaranteed to signal a service request when it has four or more empty entries and negate the request when the FIFO contains five or more entries that are filled.

If the DMA is used to service either one or both of the SDLC's FIFOs, the burst size must be set to 4 words, even though more than four entries of data may exist within the receive FIFO. If programmed I/O is used to service the FIFOs, a maximum of 4 words may be added to the transmit FIFO without checking if more space is available. Likewise, a maximum of 4 words may be removed from the receive FIFO without checking if more data is available. After this point, the user must poll a set of status bits that indicate if any data remains in the receive FIFO or if space is available in the transmit FIFO before emptying or filling the FIFOs any further.

11.9.1.12 CPU and DMA Register Access Sizes

Bit positioning, byte ordering, and addressing of the SDLC is described in terms of little endian ordering. All SDLC registers are 8 bits wide and are located in the least significant byte of individual words. The ARM peripheral bus does not support byte or half-word operations. All reads and writes of the SDLC by the CPU should be wordwide. Two separate dedicated DMA requests exist for both the transmit and the receive FIFOs. If the DMA controller is used to service the transmit and/or receive FIFOs, the user must ensure that the DMA is properly configured to perform byte-wide accesses, using 4 bytes per burst (half the size of the FIFOs). Note that a separate set of registers also exist to configure UART operation.
See the Section 11.9, "Serial Port 1 – SDLC/UART" on page 11-78 for a full description of programming and the operation of serial port 1 as a UART.

11.9.2 SDLC Register Definitions

There are eight registers within serial port 1: five control registers, one data register, and two status registers. The control registers are used to select UART or SDLC mode, baud rate, number of start flags, bit modulation mode, and address match value. They are used to select whether an abort or end of frame occurs when the transmit FIFO underruns, whether the sample clock is an input or output, and which edge of the sample clock is used to sample receive data and drive transmit data. Also they are used to enable or disable the FIFO interrupt service request, sample clock input/output operation, aborts after frames, receive operation, transmit operation, receive address matching, and loopback mode. See the Section 11.9, "Serial Port 1 – SDLC/UART" on page 11-78 for a full description of UART programming and operation.

The data register addresses the top location of the transmit FIFO and bottom location of the receive FIFO. When it is read, the receive FIFO is accessed, and when it is written, the transmit FIFO is accessed.

The status registers contain bits that signal CRC, overrun, underrun, and receiver abort errors, and the transmit FIFO service request, receive FIFO service request, and end-of-frame conditions. Each of these hardware-detected events signals an interrupt request to the interrupt controller. The status registers also contains flags for transmitter busy, receiver synchronized, receive FIFO not empty, transmit FIFO not full, and receive transition detect (no interrupt generated).

11.9.3 SDLC Control Register 0

SDLC control register 0 (SDCR0) contains 8 bit fields that control various functions within the SDLC.

11.9.3.1 SDLC/UART Select (SUS)

The SDLC/UART select (SUS) bit is used to select whether serial port 1 is used for SDLC or UART operation. When SUS=0, SDLC operation is selected. The receiver and transmitter logic is then enabled individually by programming the transmitter and receiver enable bits (TXE, RXE). When SUS=0 and TXE=0, control of the transmit pin (TXD1) is given to the PPC unit; when SUS=0 and RXE=0, control of the receive pin (RXD1) is given to the PPC unit. When SUS=1, UART operation is selected and the state of all remaining SDLC register bits is ignored (remaining unchanged) and control of the TXD1 and RXD1 pins is given to the UART. See the Section 11.9, “Serial Port 1 – SDLC/UART” on page 11-78 for a description of the programming and operation of serial port 1 as a UART. SUS, TXE, and RXE are the only bits within the control register that are reset placing serial port 1 into SDLC mode while disabling the transmitter and receiver.

The user also has the ability to take control of two GPIO pins and use them for UART serial transmission while the SDLC makes use of serial port 1’s transmit and receive pins to allow both units to be used at the same time. The peripheral pin control (PPC) unit can be programmed to connect the UART’s transmit and receive lines to GPIO pins 14 and 15. When the UART pin reassignment (UPR) bit is set in the PPC pin assignment register (PPAR), the UART transmits using the GPIO<14> pin and receives using the GPIO<15> pin. The SUS bit is ignored in this case and serial port 1 operation defaults to SDLC mode. Note that the user must set bits 14 and 15 in the GPIO alternate function register (GAFR), and set bit 14 and clear bit 15 in the GPIO pin direction register (GPDR). See the “Peripheral Pin Controller (PPC)” on page 11-184 for a description of how to program the PPC and the Section 9.1, “General-Purpose I/O” on page 9-1 for a description of how to program the GPIO unit for this mode of operation.

11.9.3.2 Single/Double Flag Select (SDF)

The single/double flag select (SDF) bit is used to select whether one or two flags (01111110) are transmitted at the start of each frame. When SDF=0, the transmit logic uses one flag. When SDF=1, the transmit logic uses two flags. Note that SDF does not affect the number of flags that are transmitted at the end each frame (one flag is always used). Normally, when back-to-back transmissions are made, only one flag is inserted between the two frames (one flag serves as both the frame’s start and end flag). However, when SDF=1, two flags are inserted between each frame. SDF does not affect SDLC receive operation.

11.9.3.3 Loopback Mode (LBM)

The loopback mode (LBM) bit is used to enable and disable the ability of the SDLC transmit and receive logic to communicate. When LBM=0, the SDLC operates normally. The transmit and receive data paths are independent and communicate via their respective pins. When LBM=1, the output of the transmit serial shifter is directly connected to the input of the receive serial shifter internally, and control of the TXD1 and RXD1 pins are given to the peripheral pin control (PPC) unit.

11.9.3.4 Bit Modulation Select (BMS)

The bit modulation select (BMS) bit selects whether the SDLC uses NRZ or FM0 bit encoding for both transmit and receive data. When BMS=0, FM0 encoding is selected and when BMS=1, NRZ encoding is selected. In frequency modulation zero (FM0) encoding, a transition occurs on every bit boundary. Zeros are represented by an additional transition in the middle of the bit period, and ones are represented by the lack of an additional transition in the middle of the bit period. In nonreturn to zero (NRZ) encoding, a one is represented when the pin is high, and a zero when the pin is low. Note that bit-stuffing/bit-extraction (the insertion/deletion of a zero after five ones are encountered) is not affected by BMS. Also note that NRZ encoding must be selected (BMS=1) when sample clock operation is enabled (SCE=1).

11.9.3.5 Sample Clock Enable (SCE)

The sample clock enable (SCE) bit is used to enable or disable driving or receiving a clock using GPIO pin 16 for synchronous transmission/reception of data. When SCE=0, the on-chip 3.6864-MHz PLL, the SDLC's programmable baud rate generator, and the receive logic's digital PLL are used. When SCE=1, the sample clock direction (SCD) bit is decoded to determine the direction of the clock used on GPIO pin 16.

11.9.3.6 Sample Clock Direction (SCD)

When the sample clock function is enabled (SCE=1), the sample clock direction (SCD) bit is used to select whether the sample clock is an input from or an output to GPIO pin 16.

When SCD=0, the sample clock is input using GPIO pin 16 and is used to synchronously drive both the transmit and receive logic. For the receive logic, the RCE bit is decoded to select which edge of the input clock is used to latch each bit of the incoming frame. Note that the clock is not embedded within the data stream, and the digital PLL is shut down to conserve power. For the transmit logic, the TCE bit is decoded to select which edge of the input clock is used to drive each bit of the outgoing frame. The on-chip clock used to drive the programmable baud rate generator is shut down to conserve power. Note that input clock frequency to GPIO<16> cannot exceed 3.6864 MHz.

When SCD=1, the sample clock, which is generated within the SDLC unit (the clock that is output after dividing the 3.6864-MHz reference by the programmable BCD field, but before the fixed divide by 16), is output to GPIO pin 16, and again the RCE and TCE bits are decoded to determine which edge of this clock output is used to sample receive data and drive transmit data. Because the baud clock that is generated before the fixed divide by 16 is used to synchronously drive the SDLC, the effective baud rate is 16 times greater, allowing the SDLC to operate at speeds ranging from 899.78 bps to 3.6864 Mbps.

When the sample clock function is enabled (SCE=1), the user must program the SDLC bit modulation select (BMS) control bit to select NRZ encoding (BMS=1). Unpredictable results occur when FM0 encoding is selected during sample clock operation. Note that the SDLC frame format is not affected during sample clock operation, only the sampling and driving of individual data bits. Bit stuff (insertion of a zero after five consecutive ones) still occurs during NRZ encoding.

11.9.3.7 Receive Clock Edge Select (RCE)

When sample clock operation is enabled (SCE=1), the receive clock edge select (RCE) bit is used to select which edge of the clock input from or output to GPIO pin 16 to use (rising or falling) to synchronously sample data from the receive pin. When RCE=0, each bit received is sampled on the rising edge of the sample clock; when RCE=1, bits are sampled on the clock's falling edge. Note that the internal baud rate generator and receive logic's digital PLL are not used in this mode.

11.9.3.8 Transmit Clock Edge Select (TCE)

When sample clock operation is enabled (SCE=1), the transmit clock edge select (TCE) bit is used to select which edge of the clock input from or output to GPIO pin 16 to use (rising or falling) to synchronously drive data onto the transmit pin. When TCE=0, each bit transmitted is driven on the rising edge of the sample clock; when TCE=1, bits are driven on the clock's falling edge. Note that the internal baud rate generator is not used in this mode.

The following table shows the location of all bit fields located in SDLC control register 0 (SDCR0). The SDLC must be disabled (SUS=RXE=TXE=0) when changing the state of any bit within this register. The reset state of all control bits except SUS is unknown (indicated by question marks) and must be initialized before enabling the SDLC.

Address: 0h 8002 0060 **SDCR0** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	TCE	RCE	SCD	SCE	BMS	LBM	SDF	SUS
Reset	?	?	?	?	?	?	?	0

Bit	Name	Description
0	SUS	SDLC/UART select. 0 – SDLC mode selected. 1 – UART mode selected. **Note:** For SUS=0, if TXE=0, TXD1 control is given to PPC unit; if RXE=0, RXD1 control is given to PPC unit. If UPR is set in the PPC unit, SUS is ignored, the UART uses GPIO<14> to transmit and GPIO<15> to receive data, and serial port 1 defaults to SDLC mode. The user must also program the GAFR and GPDR registers appropriately in the GPIO unit.
1	SDF	Single/double flag select. 0 – One flag generated at start of each transmit frame. 1 – Two flags generated at start of each transmit frame. **Note:** SDF does not affect receive operation.
2	LBM	Loopback mode. 0 – Normal serial port operation enabled. 1 – Output of transmit serial shifter is connected to input of receive serial shifter internally and control of TXD1 and RXD1 pins is given to the PPC unit.
3	BMS	Bit modulation select. 0 – FM0 bit encoding/decoding selected. 1 – NRZ bit encoding/decoding selected. **Note:** BMS must be programmed to select NRZ (BMS=1) encoding when sample clock operation is enabled (SCE=1).

4	SCE	Sample clock enable. 0 – On-chip baud rate generator and digital PLL used to transmit and receive SDLC data. 1 – A clock is input or output via GPIO pin 16 and is used to synchronously sample receive data and drive transmit data. **Note:** BMS must be programmed to select NRZ encoding when sample clock operation is enabled (BMS=1).
5	SCD	Sample clock direction. 0 – If sample clock enabled, it is input using GPIO pin 16. 1 – If sample clock enabled, the sample clock generated by the programmable baud rate generator but before the fixed divide by 16 is output using GPIO pin 16. **Note:** For both directions, the sample clock is used to synchronously sample receive data and drive transmit data on the edges selected using RCE and TCE. A maximum of 3.6864-MHz clock allowed.
6	RCE	Receive clock edge select. 0 – Rising edge of clock input/output on GPIO pin 16 used to latch data from the receive pin. 1 – Falling edge of clock input/output on GPIO pin 16 used to latch data from the receive pin.
7	TCE	Transmit clock edge select. 0 – Rising edge of clock input/output on GPIO pin 16 used to drive data onto the transmit pin. 1 – Falling edge of clock input/output on GPIO pin 16 used to drive data onto the transmit pin.

11.9.4 SDLC Control Register 1

SDLC control register 1 (SDCR1) contains eight bit fields that control various functions within the SDLC.

11.9.4.1 Abort After Frame (AAF)

The abort after frame (AAF) bit controls whether or not the SDLC transmits an abort at the end of each frame transmitted, and also controls the state of GPIO pin 17. When the AAF bit is set, each time the SDLC completes transmission of the flag at the end of a frame, the transmit logic signals an abort by transmitting 12 sequential ones on the transmit pin (TXD1). Additionally, any time the transmitter is idle (not sending a frame or the abort at the end of the frame), the SDLC forces GPIO pin 17 high. Likewise, when the SDLC is actively transmitting a frame (including the start and stop flags, and the abort at the end of the frame), it forces GPIO pin 17 low. If the transmit FIFO is emptied at the end of a frame, the abort is signalled followed by the continuous transmission of flags. If there is data present within the FIFO (indicating a new frame is available), the abort is followed by the programmed number of start flags, then data transmission begins again. For this case, GPIO<17> is not asserted because the two frames occur back-to-back (no idle time between the two frames). Note that the user must configure GPIO<17> as an output by setting the pin direction bit for pin 17 within GPDR. When AAF=1, the state of GPIO<17> is controlled solely by serial port 1. Writing to the pin set (GPSR) or pin clear (GPCR) registers for pin 17 has no effect. See Chapter 9, “System Control Module” for a description of GPIO programming.

11.9.4.2 Transmit Enable (TXE)

The transmit enable (TXE) bit is used to enable and disable SDLC transmit operation. When TXE=0, the transmit logic is disabled and its clocks are turned off to conserve power. When TXE=1, the SDLC transmitter logic is enabled for serial transmission. It is required that the user first program all other control bits before setting TXE. If the TXE bit is cleared to zero while the SDLC is actively transmitting data, transmission is stopped immediately, all data within the transmit FIFO and serial output shifter is cleared, and control of the TXD1 pin is given to the peripheral pin control (PPC) unit. Note that SUS, TXE, and RXE are the only control bits within the SDLC that are initialized when a hardware reset occurs. Clearing TXE to zero ensures the SDLC transmitter is disabled, giving control of the transmit pin to the PPC unit, which configures TXD1 as an input following a reset of the SA-1100. Note that TXE is ignored when SUS=1 (enables UART operation).

11.9.4.3 Receive Enable (RXE)

The receive enable (RXE) bit is used to enable or disable SDLC receive operation. When RXE=0, the receive logic is disabled and its clocks are turned off to conserve power. When RXE=1, the SDLC receiver logic is enabled for serial reception. It is required that the user first program all other control bits before setting RXE. If the RXE bit is cleared to zero while the SDLC is actively receiving data, reception is stopped immediately, all data within the receive FIFO and serial input shifter is cleared, and control of the RXD1 pin is given to the peripheral pin control (PPC) unit. Note that SUS, TXE, and RXE are the only control bits within the SDLC that are initialized when a hardware reset occurs. Clearing RXE to zero ensures the SDLC receiver is disabled, giving control of the receive pin to the PPC unit, which configures RXD1 as an input following a reset of the SA-1100. Note that RXE is ignored when SUS=1 (enables UART operation).

11.9.4.4 Receive FIFO Interrupt Enable (RIE)

The receive FIFO interrupt enable (RIE) bit is used to mask or enable the receive FIFO service request interrupt. When RIE=0, the interrupt is masked and the state of the receive FIFO service request (RFS) bit within SDLC status register 0 is ignored by the interrupt controller. When RIE=1, the interrupt is enabled and whenever RFS is set (one), an interrupt request is made to the interrupt controller. Note that programming RIE=0 does not affect the current state of RFS or the receive FIFO logic’s ability to set and clear RFS; it only blocks the generation of the interrupt request. Also note that RIE does not affect generation of the receive FIFO DMA request, which is asserted whenever RFS=1.

11.9.4.5 Transmit FIFO Interrupt Enable (TIE)

The transmit FIFO interrupt enable (TIE) bit is used to mask or enable the transmit FIFO service request interrupt. When TIE=0, the interrupt is masked and the state of the transmit FIFO service request (TFS) bit within SDLC status register 0 is ignored by the interrupt controller. When TIE=1, the interrupt is enabled, and whenever TFS is set (one), an interrupt request is made to the interrupt controller. Note that programming TIE=0 does not affect the current state of TFS or the transmit FIFO logic’s ability to set and clear TFS; it only blocks the generation of the interrupt request. Also note that TIE does not affect generation of the transmit FIFO DMA request, which is asserted whenever TFS=1.

11.9.4.6 Address Match Enable (AME)

The address match enable (AME) bit is used to enable or disable the receive logic from comparing the address programmed in the address match value (AMV) bit field to the address of all incoming frames. When AME=1, data is stored in the receive FIFO for only those frames that have addresses that match AMV, and for any frame that contains an address that contains all ones (11111111), denoting a global address. For frames in which the address does not match, the data and CRC are ignored and the receiver begins to search for the next flag. When AME=0, address values are not compared and the data in every frame is stored in the receive FIFO.

11.9.4.7 Transmit FIFO Underrun Select (TUS)

The transmit FIFO underrun select (TUS) bit is used to select what action to take as a result of a transmit FIFO underrun and to mask or enable the transmit FIFO underrun interrupt.

When TUS=0, transmit FIFO underruns are used to signal the transmit logic that the end of the frame has been reached. When the transmit FIFO experiences an underrun, the CRC value, which is calculated continuously on outgoing data, is loaded to the serial shifter and transmitted, followed by a flag. Also when TUS=0, the transmit FIFO interrupt is masked and the state of the transmit FIFO underrun (TUR) status bit is ignored by the interrupt controller.

When TUS=1, transmit FIFO underruns are used to signal the transmit logic that the end of the frame has not yet been reached and that the rate in which data is supplied to the transmit FIFO is not sufficient. When the transmit FIFO experiences an underrun, ones are continuously output by the transmitter to signal an abort condition until data is once again available within the transmit FIFO, and the CRC value is discarded. Additionally, when TUS=1, the transmit FIFO underrun interrupt is enabled, and whenever TUR is set (one), an interrupt request is made to the interrupt controller. To change the state of this bit during operation, the user should fill the transmit FIFO to ensure TUS is not written at the same time the transmit FIFO underruns. Note that programming TUS=0 does not affect the current state of TUR or the transmit FIFO logic's ability to set and clear TUR; it only blocks the generation of the interrupt request.

TUS is useful for ensuring that frames are not prematurely ended due to an unexpected transmit FIFO underrun. At the start of a frame, the user can configure TUS=1 so that any underrun signals an abort to the off-chip receiver. Just before the end of the frame, the user can then configure TUS=0 (the last time the transmit FIFO is filled, for example), allowing the remaining data to be output by the transmit logic. The FIFO then underruns, causing the CRC and end flag to be transmitted.

11.9.4.8 Receiver Abort Interrupt Enable(RAE)

The receiver abort interrupt enable (RAE) bit is used to mask or enable whether or not an abort sequence, which is detected by the receive logic, generates an interrupt to the CPU. When RAE=0, the interrupt is masked and the state of the receiver abort status (RAS) bit is ignored by the interrupt controller. When RAE=1, the interrupt is enabled and whenever RAS is set (one), an interrupt request is made to the interrupt controller. Note that programming RAE=0 does not affect the current state of RAS or the receive logic's ability to set and clear RAS as the result of an abort detect; it only blocks the generation of the interrupt request.

The following table shows the location of the bits within SDLC control register 1. RXE and TXE are the only control bits in this register that are reset to a known state to ensure the SDLC is disabled following a reset of the SA-1100. The reset state of all other control bits is unknown (indicated by question marks) and must be initialized before enabling the SDLC. Note that SDCR1 may be written while the SDLC is enabled to allow various modes to be changed during active operation.

Address: 0h 8002 0064 **SDCR1** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	RAE	TUS	AME	TIE	RIE	RXE	TXE	AAF
Reset	?	?	?	?	?	0	0	?

Bit	Name	Description
0	AAF	Abort after frame. 0 – Aborts not signalled following transmission of a frame. GPIO<17> controlled by system unit. 1 – Abort is signalled after the end flag of a frame by transmitting 12 ones. GPIO<17> pin forced high during idle; forced low during transmission of a frame or the abort. **Note:** The user must configure GPIO<17> as an output within GPDR in the system control module.
1	TXE	Transmit enable. 0 – SDLC transmit logic disabled. Control of the TXD1 pin is given to the PPC unit if SUS=0. 1 – SDLC transmit logic enabled if SUS=0.
2	RXE	Receive enable. 0 – SDLC receive logic disabled. Control of the RXD1 pin is given to the PPC unit if SUS =0. 1 – SDLC receive logic enabled if SUS=0.
3	RIE	Receive FIFO interrupt enable. 0 – Receive FIFO one- to two-thirds full or more condition does not generate an interrupt (RFS bit ignored). 1 – Receive FIFO one- to two-thirds full or more condition generates an interrupt (state of RFS sent to interrupt controller).
4	TIE	Transmit FIFO interrupt enable. 0 – Transmit FIFO half-full or less condition does not generate an interrupt (TFS bit ignored). 1 – Transmit FIFO half-full or less condition generates an interrupt (state of TFS sent to interrupt controller).
5	AME	Address match enable. 0 – Disable receiver address match function. Stores data from all incoming frames in receive FIFO. 1 – Enable receiver address match function. Do not FIFO data unless address recognized or incoming address contains all ones (0hFF).
6	TUS	Transmit FIFO underrun select. 0 – Transmit FIFO underrun. Causes CRC and a flag to be transmitted, and masks interrupt generation (TUR ignored). 1 – Transmit FIFO underrun. Causes an abort to be transmitted, and generates an interrupt (state of TUR sent to interrupt controller).
7	RAE	Receiver abort interrupt enable. 0 – Abort detected by the receiver. Does not generate an interrupt (RAS bit ignored). 1 – Abort detected by the receiver. Generates an interrupt (state of RAS sent to interrupt controller).

11.9.5 SDLC Control Register 2

SDLC control register 2 (SDCR2) contains the 8-bit address match value field that is used by the SDLC to selectively receive frames.

11.9.5.1 Address Match Value (AMV)

The 8-bit address match value (AMV) field is programmed with an address value that is used to selectively store only the data within receive frames that have the same address value. The address match enable (AME) bit must be set to enable this function. For incoming frames, which have the same address value as the AMV field, the frame's address, control, and data are stored in the receive FIFO. For those that do not, the remainder of the frame is ignored, and the receive logic looks for the next start flag in the incoming data stream. One special address exists that is always matched by the address match logic regardless of the value programmed in AMV. When address matching is enabled, whenever a frame is received with an address containing all ones (11111111), the value programmed in AMV is ignored and the frame data is automatically stored in the receive FIFO. The address value is contained within the first byte of data in a frame following the flag. AMV can be written at any time, and is used for comparison for the next frame that occurs following its update.

The following table shows the address match value field within SDLC control register 2. The reset state of AMV is unknown (indicated by question marks) and must be initialized before enabling the SDLC. Note that SDCR2 may be written while the SDLC is enabled to allow the address match value to be changed during active receive operation.

Address: 0h 8002 0068 **SDCR2** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	AMV							
Reset	?	?	?	?	?	?	?	?

Bit	Name	Description
7..0	AMV	Address match value. The 8-bit value used by receiver logic to compare to address of incoming frames. If address matches, store frame address, control, and data in receive FIFO; if address does not match, ignore frame and search for next flag. **Note:** An address of 0hFF (all ones) in the incoming frame automatically generates a match (AMV is ignored).

11.9.6 SDLC Control Registers 3 and 4

SDLC control register 3 (SDCR3) contains the upper 4 bits and SDLC control register 4 (SDCR4) the lower 8 bits of the baud rate divisor field.

11.9.6.1 Baud Rate Divisor (BRD)

The 12-bit baud rate divisor (BRD) field is used to select the baud or bit rate of the SDLC. A total of 4096 different baud rates can be selected, ranging from a minimum of 56.24 bps to a maximum of 230.4 Kbps. The baud rate generator uses the 3.6864-MHz clock generated by the on-chip PLL and first divides it by the programmable baud rate using BRD. The resultant clock (called the sample clock) is then divided by 16 to generate the bit clock. The receive baud clock is synchronized with the data steam each time a transition is detected on the receive data line at a bit's boundary. The resultant baud rate given a specific BRD value, or required BRD value given a desired baud rate, can be calculated using the following two respective equations, where BRD is the decimal equivalent of the unsigned binary value programmed within the bit field:

$$BaudRate = \frac{3.6864 \times 10^6}{16X(BRD+1)}$$

$$BRD = \frac{3.6864 \times 10^6}{16X BaudRate} - 1$$

The following tables show the bit locations corresponding to the baud rate divisor field that is split between two 8-bit registers. The upper 4 bits of BRD reside within SDCR3 and the lower 8 bits reside within SDCR4. The SDLC must be disabled (SUS=RXE=TXE=0) whenever these registers are written. Note that writes to reserved bits are ignored and reads return zeros; question marks indicate that the values are unknown at reset.

Address: 0h 8002 006C **SDCR3** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	Reserved				BRD<11:8>			
Reset	0	0	0	0	?	?	?	?

Bit	Name	Description
3..0	BRD<11:8>	Baud rate divisor. Encoded value (from 0 to 4096). Used to generate the baud rate of the SDLC. Baud Rate = $3.6864 \times 10^6/(16 \times (BRD+1))$, where BRD is a decimal value.
7..4	—	Reserved.

Address: 0h 8002 0070 **SDCR4** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	BRD<7:0>							
Reset	?	?	?	?	?	?	?	?

Bit	Name	Description
7..0	BRD<7:0>	Baud rate divisor. Encoded value (from 0 to 4096). Used to generate the baud rate of the SDLC. Baud Rate = $3.6864 \times 10^6/(16 \times (BRD+1))$, where BRD is a decimal value.

11.9.7 SDLC Data Register

The SDLC data register (SDDR) is an 8-bit register corresponding to both the top and bottom entries of the transmit and receive FIFOs, respectively.

When SDDR is read, the lower 8 bits of the bottom entry of the 11-bit receive FIFO is accessed. As data enters the top of the receive FIFO, bits 8..10 are used as tags to indicate various conditions that occur during reception of each piece of data. The tag bits are transferred down the FIFO along with the data byte that encountered the condition. When data reaches the bottom, bit 8 of the bottom FIFO entry is automatically transferred to the end of frame (EOF) flag, bit 9 to the CRC error (CRE) flag, and bit 10 to the receiver overrun (ROR) flag, all within SDLC status register 1. The user can read these flags to determine if the value at the bottom of the FIFO represents the last byte within the packet and/or encountered an error during reception. After checking the flags, the FIFO value can then be read, which causes the data in the next location of the receive FIFO to automatically transfer down to the bottom entry and its EOF/CRE/ROR bits to be transferred to the status register.

The end/error in FIFO (EIF) status bit is set within status register 0 whenever one or more of the tag bits (8..10) are set within any of the bottom four entries of the receive FIFO and is cleared when no error bits are set in the bottom four entries of the FIFO. When EIF is set, an interrupt is generated and receive FIFO DMA requests are disabled so that the user can manually empty FIFO, always checking the end of frame, CRC error, and overrun error flags in status register 1 *first* before removing each data value from the FIFO. After each entry is removed, the user should check the EIF bit to see if any errors remain, and repeat the procedure until all errors are flushed from the FIFO. Once EIF is cleared, servicing of the receive FIFO by the DMA controller is automatically reenabled.

When SDDR is written, the topmost entry of the 8-bit transmit FIFO is accessed. After a write, data is automatically transferred down to the lowest location within the transmit FIFO, which does not already contain valid data. Data is removed from the bottom of the FIFO one piece at a time by the transmit logic, is loaded into the transmit serial shifter, and is then serially shifted out onto the TXD1 pin at the programmed baud rate.

The following table shows the bit locations corresponding to the data field and end-of-frame bit as well as the cyclic redundancy check and receiver overrun error bits within the SDLC data register. Note that both FIFOs are cleared when the SA-1100 is reset, the transmit FIFO is cleared when writing TXE=0, and the receive FIFO is cleared when writing RXE=0.

Address: 0h 8002 0078 **SDDR** **Read/Write**

Bit	10	9	8	7	6	5	4	3	2	1	0
	ROR	CRE	EOF	Bottom of Receive FIFO Data							
Reset	0	0	0	0	0	0	0	0	0	0	0

Read Access

Note: ROR, CRE, EOF are not read, but rather are transferred to corresponding status bits in SDSR1 each time a new data value is transferred to SDDR.

Bit	7	6	5	4	3	2	1	0
	Top of Transmit FIFO Data							
Reset	0	0	0	0	0	0	0	0

Write Access

Bit	Name	Description
7..0	DATA	Top/bottom of transmit/receive FIFO data. Read – Bottom of receive FIFO. Write – Top of transmit FIFO.
8	EOF	End of frame. 0 – The last byte of the frame has not been encountered. 1– The data value at the bottom of the receive FIFO represents the last byte of the frame. **Note:** Each time an 11-bit value reaches the bottom of the receive FIFO, bit 8 from the last FIFO entry is transferred to the EOF bit in SDSR1.
9	CRE	CRC error. 0 – CRC not encountered yet, or the CRC value calculated on the incoming data matched the received CRC value. 1 – The CRC value calculated on the incoming data did not match the received CRC value. **Note:** Each time an 11-bit value reaches the bottom of the receive FIFO, bit 9 from the last FIFO entry is transferred to the CRE bit in SDSR1.
10	ROR	Receiver overrun. 0 – No receiver overrun has been detected. 1 – Receive logic attempted to place data into receive FIFO while it was full; one or more data values *after* the data value at the bottom of the receive FIFO were lost. **Note:** Each time an 11-bit value reaches the bottom of the receive FIFO, bit 10 from the last FIFO entry is transferred to the ROR bit in SDSR1.

11.9.8 SDLC Status Register 0

SDLC status register 0 (SDSR0) contains bits that signal the transmit FIFO service request, receive FIFO service request, receiver abort, transmit FIFO underrun, and the end/error in receive FIFO condition. Each of these hardware-detected events signal an interrupt request to the interrupt controller.

A bit that can cause an interrupt signals the interrupt request as long as the bit is set. Once the bit is cleared, the interrupt is cleared. Read/write bits are called status bits; read-only bits are called flags. Status bits are referred to as “sticky” (once set by hardware, must be cleared by software). Writing a one to a sticky status bit clears it; writing a zero has no effect. Read-only flags are set and cleared by hardware; writes have no effect. Additionally, some bits that cause interrupts have corresponding enable/mask bits in the control registers and are indicated in the following section headings. Note that the user has the ability to mask all SDLC interrupts by clearing bit 14 within the interrupt controller mask register (ICMR). See the Section 9.2, “Interrupt Controller” on page 9-11.

11.9.8.1 End/Error in FIFO Status (EIF) (read-only, nonmaskable interrupt)

The end/error in FIFO flag (EIF) is a read-only bit that is set when any tag bits (8 through 10) are set within the bottom four entries of the receive FIFO and is cleared when no error bits are set within the bottom four entries of the FIFO. When EIF is set, an interrupt is signalled and DMA requests to empty the receive FIFO are disabled until EIF is cleared. To discover which FIFO entry contains the end of frame or an error condition, the user should check the state of the EOF, CRE, and ROR bits and read the corresponding value from the SDDR. This procedure should be repeated until EIF is cleared because set tag bits that are present within *any* of the four lowest entries in the receive FIFO can set EIF. Once all set tags bits are cleared from the bottom half of the receive FIFO, EIF is automatically cleared, which in turn, clears the interrupt and reenables the receive FIFO DMA request.

11.9.8.2 Transmit Underrun Status (TUR) (read/write, maskable interrupt)

The transmit underrun status bit (TUR) is set when the transmit logic attempts to fetch data from the transmit FIFO after it has been completely emptied. When an underrun occurs, the transmitter takes one of two actions. When the transmit underrun select bit is clear (TUS=0), the transmitter ends the frame by shifting out the CRC that is calculated continuously on outgoing data, followed by a flag. When TUS=1, the transmitter is forced to transmit an abort and continues to transmit ones until valid data is again available within the FIFO. Once data resides within the bottom entry of the transmit FIFO, a new data frame is initiated by transmitting a start flag followed by the transmission of data from the FIFO. When the TUR bit is set, an interrupt request is made unless it is masked. When TUS=0, the interrupt is masked; when TUS=1 it is enabled. Note that underruns are not generated when the SDLC transmitter is first enabled and is in the idle state (continuously transmits flags).

11.9.8.3 Receiver Abort Status (RAB) (read/write, maskable interrupt)

The receiver abort status bit (RAB) is set for three different cases:

- when an abort is detected during receipt of an incoming frame
- if the receive carrier is lost during active operation
- if the stop flag is not received on a byte boundary.

An abort is signalled when seven or more consecutive ones are detected on the RXD1 pin. An abort is also signalled if the receive pin is held high or low for more than six bit periods, which indicates a loss of carrier. It is also generated when the end flag is received and it is not on a byte boundary,

which indicates that the address, control, and data fields did not add up to an even multiple of 8 bits. When an abort is received, the current data byte within the serial shifter is discarded, the least recent byte (the oldest of the two bytes) of data in the temporary FIFO is moved to the receive FIFO (the other byte is discarded), and the EOF tag is set in the FIFO entry that corresponds to the last piece of data that was received before the frame was aborted. The receiver then enters hunt mode, searching for a flag. When the RAB bit is set, an interrupt request is made unless the receiver abort enable (RAE) bit is cleared.

11.9.8.4 Transmit FIFO Service Request Flag (TFS) (read-only, maskable interrupt)

The transmit FIFO service request flag (TFS) is a read-only bit that is set when the transmit FIFO is nearly empty and requires service to prevent an underrun. TFS is set whenever the transmit FIFO has four or fewer entries of valid data (half-full or less), and is cleared when it has five or more entries of valid data. When the TFS bit is set, an interrupt request is made unless the transmit FIFO interrupt request enable (TIE) bit is cleared. The state of TFS is also sent to the DMA controller, and can be used to signal a DMA service request. Note that TIM has no effect on the generation of the DMA service request. After the DMA or CPU fills the FIFO such that five or more locations are filled within the transmit FIFO, the TFS flag (and the service request and/or interrupt) is automatically cleared.

11.9.8.5 Receive FIFO Service Request Flag (RFS) (read-only, maskable interrupt)

The receive FIFO service request flag (RFS) is a read-only bit that is set when the receive FIFO is nearly filled and requires service to prevent an overrun. The amount of data that causes RFS to be set is nondeterministic. However, the range in which RFS will be set is guaranteed. RFS is set at some point when the receive FIFO is one- to two-thirds full (or more). The UART's FIFOs are self-timed to reduce cost and save power. As a result, the depth at which the receive FIFO service request is generated is variable. This is the reason the receive FIFO is twelve entries deep instead of eight like the transmit FIFO. At which entry in the FIFO the request is actually triggered is dependent on IC process, operating temperature, and so on. The receive FIFO is designed to signal the RFS bit to be set when it contains eight entries of valid data. However, because of the variability of the self-timed logic, RFS may also be set when seven, six, or five entries of valid data are present within the FIFO. Likewise, under normal circumstances, RFS is cleared when the receive FIFO has seven remaining entries of valid data. However, again due to variations, RFS may be cleared when six or five entries of data remain.

When the RFS bit is set, a DMA service request is made. An interrupt request is also made unless the receive FIFO interrupt request enable (RIE) bit is cleared. Even though more than four entries of data may exist within the receive FIFO, the user must configure the DMA burst size to four words. If programmed I/O is used to service the receive FIFO, a maximum of 4 words may be removed without checking if data is valid. After this point, the receive FIFO not empty (RNE) flag must be polled before each read to see if more data remains. After the DMA or CPU empties the FIFO such that five or more empty locations are available within the receive FIFO, the RFS flag (as well as the DMA and interrupt request) is automatically cleared.

The following table shows the bit locations corresponding to the status and flag bits within SDLC status register 0. Note that the reset state of all writable status bits is unknown (indicated by question marks) and must be cleared (by writing a one to them) before enabling the SDLC. Also note that writes to reserved bits are ignored and reads return zeros.

Address: 0h 8002 0080 **SDSR0** **Read/Write & Read-Only**

Bit	7	6	5	4	3	2	1	0
	Reserved			RFS	TFS	RAB	TUR	EIF
Reset	0	0	0	0	0	?	?	?

Bit	Name	Description
0	EIF	Error in FIFO (read-only). 0 – Bits 8..10 are not set within any of the four bottom entries of the receive FIFO; receive FIFO DMA service requests are enabled. 1 – One or more tag bits (8..10) are set within one or more of the bottom four entries of the receive FIFO; request interrupt, disable receive FIFO DMA service requests.
1	TUR	Transmit FIFO underrun. 0 –Transmit FIFO has not experienced an underrun. 1 – Transmit logic attempted to fetch data from transmit FIFO while it was empty; interrupt request signalled if not masked (if TUS=1).
2	RAB	Receiver abort. 0 – No abort has been detected for the incoming frame. 1 – Abort detected during receipt of incoming frame, seven or more ones detected on receive pin, EOF bit set in receive FIFO next to last piece of “good” data received before the abort, interrupt requested if it is enabled (if RAE=1).
3	TFS	Transmit FIFO service request (read-only). 0 – Transmit FIFO is more than half-full (five or more entries filled) or transmitter disabled. 1 – Transmit FIFO is half-full or less (four or fewer entries filled) and transmitter operation is enabled. DMA service request signalled, interrupt request signalled if it is enabled (if TIE=1).
4	RFS	Receive FIFO service request (read-only). 0 – Receive FIFO contains seven or fewer entries of data or receiver disabled. 1 – Receive FIFO is one- to two-thirds full (contains 5, 6, 7, or 8 entries of data) or more, receiver operation is enabled, DMA service request signalled, and interrupt request signalled if it is enabled (if RIE=1).
7..5	—	Reserved.

11.9.9 SDLC Status Register 1

SDLC status register 1 (SDSR1) contains flags and status bits that indicate when the receiver is synchronized, the transmitter is active, that the transmit FIFO is not full, that the receive FIFO is not empty, a transition has been detected on the receive line, and when an end of frame, CRC error, or underrun error has occurred. All bits within SDSR1 are noninterruptible.

11.9.9.1 Receiver Synchronized Flag (RSY) (read-only, noninterruptible)

The receiver synchronized (RSY) flag is a read-only bit that is set when the receiver is synchronized with the incoming data stream and is cleared when the receiver logic is in hunt mode (looking for a flag to achieve bit and frame synchronization) or the receiver is disabled (RXE=0). This bit does not request an interrupt.

11.9.9.2 Transmitter Busy Flag (TBY) (read-only, noninterruptible)

The transmitter busy (TBY) flag is a read-only bit that is set when the transmitter is actively transmitting a frame (address, control, data, CRC, start, or stop flag) or an abort, and is cleared when the transmitter is idle (transmitting flags that are not part of a frame) or the transmitter is disabled (TXE=0). This bit does not request an interrupt.

11.9.9.3 Receive FIFO Not Empty Flag (RNE) (read-only, noninterruptible)

The receive FIFO not empty flag (RNE) is a read-only bit that is set whenever the receive FIFO contains one or more bytes of valid data and is cleared when it no longer contains any valid data. This bit can be polled when using programmed I/O to remove remaining bytes of data from the receive FIFO because DMA service and CPU interrupt requests are made only when 8, 7, 6, or 5 bytes reside within the FIFO. Data remains after each service request as well as at the end of a frame. This bit does not request an interrupt.

11.9.9.4 Transmit FIFO Not Full Flag (TNF) (read-only, noninterruptible)

The transmit FIFO not full flag (TNF) is a read-only bit that is set whenever the transmit FIFO contains one or more entries that do not contain valid data and is cleared when the FIFO is completely full. This bit can be polled when using programmed I/O to fill the transmit FIFO over its halfway mark. This bit does not request an interrupt.

11.9.9.5 Receive Transition Detect Status (RTD) (read/write, noninterruptible)

The receive transition detect (RTD) status bit is set whenever the receiver is enabled (RXE=1) and a transition is detected on the RXD1 pin (either rising or falling). This bit does not request an interrupt.

11.9.9.6 End of Frame Flag (EOF) (read-only, noninterruptible)

The end of frame flag (EOF) is set when the last byte of data within a frame (including aborted frames) resides within the bottom entry of the receive FIFO.

The receive FIFO contains three tag bits (8, 9, and 10) that are not directly readable. The 8th bit is set at the top of the FIFO whenever the last byte within a frame is moved from the receive serial shifter to the top of the receive FIFO. This tag travels along with the last data value as it moves down the FIFO. Each time a data value is transferred to the bottom of the FIFO (caused by a read of the previous value), the state of the tag bit is moved from the FIFO to the EOF bit in the status

register. After the error in FIFO (EIF) status bit is set, the user should always read SDSR1 first to check EOF before reading the data value from SDDR because EOF corresponds to the current data byte at the bottom of the receive FIFO and is updated each time data is removed from the FIFO.

11.9.9.7 CRC Error Status (CRE) (read-only, noninterruptible)

The CRC error flag (CRE) is set when the CRC value calculated by the receive logic does not match the CRC value contained within the incoming serial data stream.

The receive FIFO contains 3 tag bits (8, 9, and 10) that are not directly readable. Whenever a CRC error is detected, the 9th bit is set within the top entry of the receive FIFO, corresponding to the last byte of data within the frame. This tag travels along with the last piece of data from the frame as it moves down the FIFO. Each time a data value is transferred to the bottom of the FIFO (caused by a read of the previous value), the state of the tag bit is moved from the FIFO to the CRE bit in the status register, indicating whether or not the frame has encountered a CRC error. After the error in the FIFO (EIF) status bit is set, the user should always read SDSR1 first to check CRE before reading the data value from SDDR because CRE corresponds to the current data byte at the bottom of the receive FIFO and is updated each time data is removed from the FIFO.

11.9.9.8 Receiver Overrun Status (ROR) (read-only, noninterruptible)

The receiver overrun flag (ROR) is set when the receive logic attempts to place data into the receive FIFO after it has been completely filled.

The receive FIFO contains 3 tag bits (8, 9, and 10) that are not directly readable. The 10th bit is set within the top entry of the receive FIFO whenever an overrun occurs. This tag travels along with the last “good” data value before the overflow occurred as it moves down the FIFO. Each time a data value is transferred to the bottom of the FIFO (caused by a read of the previous value), the state of the tag bit is moved from the FIFO to the ROR bit in the status register, indicating that the next value in the FIFO is the last “good” piece of data before the overflow occurred. After the error in the FIFO (EIF) status bit is set, the user should always read SDSR1 first to check CRE before reading the data value from SDDR because CRE corresponds to the current data byte at the bottom of the receive FIFO and is updated each time data is removed from the FIFO.

The following table shows the location of the flag and status bits within SDLC status register 1. The bits within this register do not produce interrupt requests. Note that the reset value of RTD is unknown (indicated by question marks) and must be cleared if set following a reset of the SA-1100. The remainder of SDSR1 is read-only (writes are ignored).

Address: 0h 8002 0084 **SDSR1** **Read/Write & Read-Only**

Bit	7	6	5	4	3	2	1	0
	ROR	CRE	EOF	RTD	TNF	RNE	TBY	RSY
Reset	0	0	0	?	1	0	0	0

Bit	Name	Description
0	RSY	Receiver synchronized flag (read-only). 0 – Receiver is in hunt mode or is disabled. 1 – Receiver logic is synchronized with the incoming data (no interrupt generated).
1	TBY	Transmitter busy flag (read-only). 0 – Transmitter is idle (continuous flags) or disabled. 1– Transmit logic is currently transmitting a frame (address, control, data, CRC, or start/stop flag) or an abort (no interrupt generated).
2	RNE	Receive FIFO not empty (read-only). 0 – Receive FIFO is empty. 1 – Receive FIFO is not empty (no interrupt generated).
3	TNF	Transmit FIFO not full (read-only). 0 – Transmit FIFO is full. 1 – Transmit FIFO is not full (no interrupt generated).
4	RTD	Receive transition detect. 0 – No transition detected on RXD1 pin since the last time software cleared this bit. 1 – Rising and/or falling edge detected on RXD1 pin (no interrupt generated).
5	EOF	End of frame (read-only). 0 – Current frame has not completed. 1 – The value at the bottom of the receive FIFO is the last byte of data within the frame.
6	CRE	CRC error (read-only). 0 – No CRC check errors encountered in the receipt of data. 1 – CRC calculated on the incoming data does not match CRC value contained within the received frame.
7	ROR	Receive FIFO overrun (read-only). 0 – Receive FIFO has not experienced an overrun. 1 – Receive logic attempted to place data into receive FIFO while it was full; the next data value in the FIFO is the last piece of “good” data before the FIFO was overrun.

11.9.10 UART Register Locations

Table 11-14 shows the registers associated with the UART and the physical addresses used to access them. See the Section 11.9, “Serial Port 1 – SDLC/UART” on page 11-78 for a description of the programming and operation of the UART (serial port 1’s UART is identical to serial port 3’s UART).

Table 11-14. UART Control, Data, and Status Register Locations

Address	Name	Description
0h 8001 0000	UTCR0	UART control register 0
0h 8001 0004	UTCR1	UART control register 1
0h 8001 0008	UTCR2	UART control register 2
0h 8001 000C	UTCR3	UART control register 3
0h 8001 0010	—	Reserved
0h 8001 0014	UTDR	UART data register
0h 8001 0018	—	Reserved
0h 8001 001C	UTSR0	UART status register 0
0h 8001 0020	UTSR1	UART status register 1
0h 8001 0024 – 0h 8001 005C	—	Reserved

11.9.11 SDLC Register Locations

Table 11-15 shows the registers associated with the SDLC and the physical addresses used to access them.

Table 11-15. SDLC Control, Data, and Status Register Locations

Address	Name	Description
0h 8002 0060	SDCR0	SDLC control register 0
0h 8002 0064	SDCR1	SDLC control register 1
0h 8002 0068	SDCR2	SDLC control register 2
0h 8002 006C	SDCR3	SDLC control register 3
0h 8002 0070	SDCR4	SDLC control register 4
0h 8002 0074	—	Reserved
0h 8002 0078	SDDR	SDLC data register
0h 8002 007C	—	Reserved
0h 8002 0080	SDSR0	SDLC status register 0
0h 8002 0084	SDSR1	SDLC status register 1
0h 8002 0088 – 0h 8002 FFFF	—	Reserved

11.10 Serial Port 2 – Infrared Communications Port (ICP)

The infrared communications port (ICP) operates at half-duplex and provides direct connection to commercially available Infrared Data Association (IrDA) compliant LED transceivers. The ICP supports both the original IrDA standard with speeds up to 115.2 Kbps as well as the newer 4-Mbps standard. Both standards use different bit encoding techniques and serial packet formats. Low-speed IrDA transmission uses the Hewlett-Packard Serial Infrared standard (HP-SIR) for bit encoding and a universal asynchronous receiver-transmitter (UART) as the serial engine; high-speed uses four-position pulse modulation (4PPM) and a specialized serial packet protocol developed expressly for IrDA transmission. To support these two standards, the ICP contains two separate blocks, each comprised of a bit encoder/decoder and serial-to-parallel data engine. The engine within the ICP that implements the special 4-Mbps protocol is called the high-speed serial to parallel (HSSP) receiver-transmitter. Only one of the two standards can be enabled at a time (the user cannot enable low-speed transmit and high-speed receive at the same time). To support a variety of IrDA transceivers, both the transmit and receive data pins can be individually configured to communicate either using normal or inverted data. Additionally, if IrDA transmission is not needed, the ICP's UART can be enabled while disabling the HP-SIR bit encoder for use as a general-purpose serial port.

Note: Programming and operation of serial port 2's UART is identical to serial port 3. See Section 11.11, "Serial Port 3 - UART" on page 11-128 for a complete description of using the ICP for low-speed IrDA operation.

The external pins dedicated to the ICP are TXD2 and RXD2. If serial transmission is not required and the ICP is disabled, control of these pins is given to the peripheral pin control (PPC) unit for use as general-purpose input/output pins (noninterruptible). See Section 11.13, "Peripheral Pin Controller (PPC)" on page 11-184.

11.10.1 Low-Speed ICP Operation

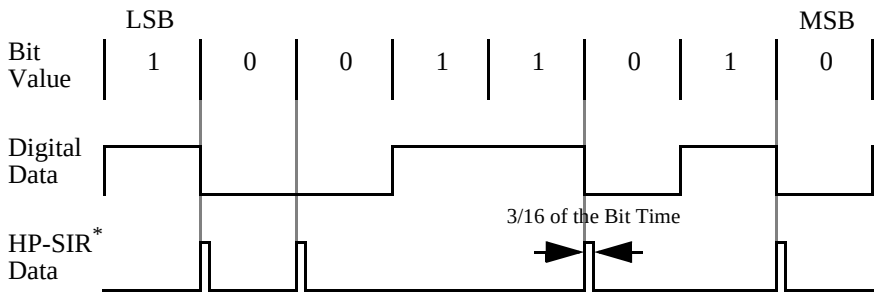
Following reset, both the UART and HSSP are disabled, which causes the peripheral pin controller (PPC) to assume control of the port's pins. Reset causes the PPC to configure all of the peripheral pins as inputs, including serial port 2's transmit (TXD2) and receive (RXD2) pins. Reset also causes the UART's transmit and receive FIFOs to be flushed (all entries invalidated). Before enabling the ICP for low-speed operation, the user must first clear any writable or "sticky" status bits, which are set by writing a one to each bit. Next, the desired mode of operation is programmed in the control registers. At this point the user may "prime" the UART's transmit FIFO by writing up to eight values, or the FIFO can remain empty and either programmed I/O or the DMA can be used to service it after the ICP is enabled. Once the ICP is enabled, transmission/reception of data can begin on the transmit (TXD2) and receive (RXD2) pins.

For low-speed operation, all serial data that is transferred between the TXD2/RXD2 pins and the ICP's UART is modulated/demodulated according to the HP-SIR IrDA standard. The IrDA standard also specifies the frame format that must be used by the UART.

11.10.1.1 HP-SIR* Modulation

Hewlett-Packard Serial Infrared* (SIR) modulation is used for low-speed transmission up to 115.2 Kbps. Logic zero is represented by a pulse of light that is either 3/16 of the bit time wide, or 1.6 µs wide (1.6 µs is 3/16 of the bit time for the highest bit rate of 115.2 Kbps). The rising edge of the pulse corresponds to the start of the zero bit time. Logic one is represented by the absence of light pulses. Figure 11-24 shows an example of HP-SIR modulation of the byte, 8'b01011001. Note that the byte is transmitted starting with the LSB first.

Figure 11-24. HP-SIR Modulation Example



11.10.1.2 UART Frame Format

For transmission rates up to 115.2 Kbps, the ICP's UART is used. The user must program it to produce a frame that produces 8 bits of data, one stop bit, and no parity, as shown in Figure 11-25. Note that PE=1, SBS=1, DSS=0, SCE=1, BRK=1, RXE=0, TXE=0, and BRD=0x000 are illegal programming modes for IrDA operation and will produce unpredictable results. See Section 11.11, "Serial Port 3 - UART" on page 11-128 for a complete description of how to program and operate the ICP's UART.

Figure 11-25. UART Frame Format for IrDA Transmission (<= 115.2 Kbps)

Start Bit	Data<7>	Data<6>	Data<5>	Data<4>	Data<3>	Data<2>	Data<1>	Data<0>	Stop Bit

UTCR0-2 Programming:

PE=0	DSS = 1	TCE = don't care	RXE = 1	RIE = 0 or 1
OES = don't care	SCE = 0	BRD = 0x001 to 0xFFF	TXE = 1	TIE = 0 or 1
SBS = 0	RCE = don't care		BRK = 0	

11.10.2 High-Speed ICP Operation

Before enabling the ICP for high-speed operation, the user must first clear any writable or "sticky" status bits that are set by writing a one to each bit. Next, the desired mode of operation is programmed in the control registers. At this point the user can "prime" the HSSP's transmit FIFO by writing up to 16 values, or the FIFO can remain empty and either programmed I/O or the DMA can be used to service it after the HSSP is enabled. Once the HSSP is enabled, transmission/reception of data can begin on the transmit (TXD2) and receive (RXD2) pins.

For high-speed operation, all serial data, which is transferred between the TXD2/RXD2 pins and the ICP's HSSP, is modulated/demodulated according to the 4PPM IrDA standard. Additionally, the HSSP uses a frame format that is very similar to the SDLC's. For high-speed transmission, both the modulation technique and the HSSP's frame format are discussed in the following sections.

11.10.2.1 4PPM Modulation

Four-position pulse modulation (4PPM) is used for the high-speed transmission rate of 4.0 Mbps. Two data bits are encoded at a time by placing a single 125 ns light pulse within one of four timeslots. The four timeslots are collectively termed a "chip." Bytes are encoded one at a time. They are divided into four individual nibbles (2-bit pairings) and the least significant nibble is transmitted first. Figure 11-26 shows the 4PPM encoding for the four possible 2-bit combinations and Figure 11-27 shows an example of 4PPM modulation of the byte 8'b10110001 that is constructed using four chips. Note that bits within each nibble are not reordered, but nibble 0 (least significant) is transmitted first, ending with nibble 3 (most significant).

Figure 11-26. 4PPM Modulation Encodings

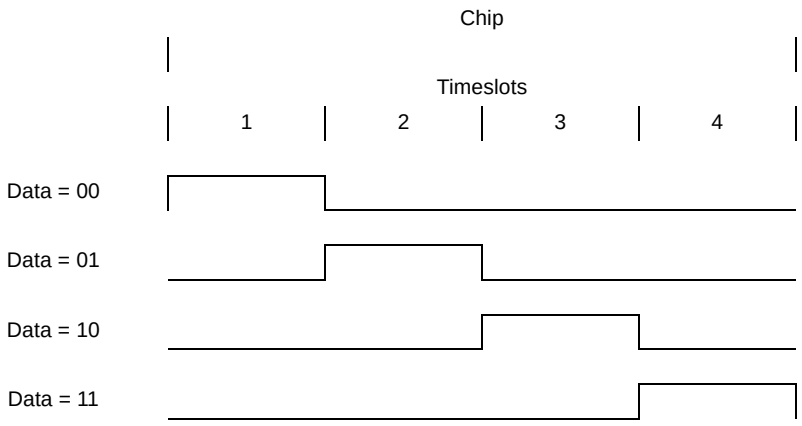
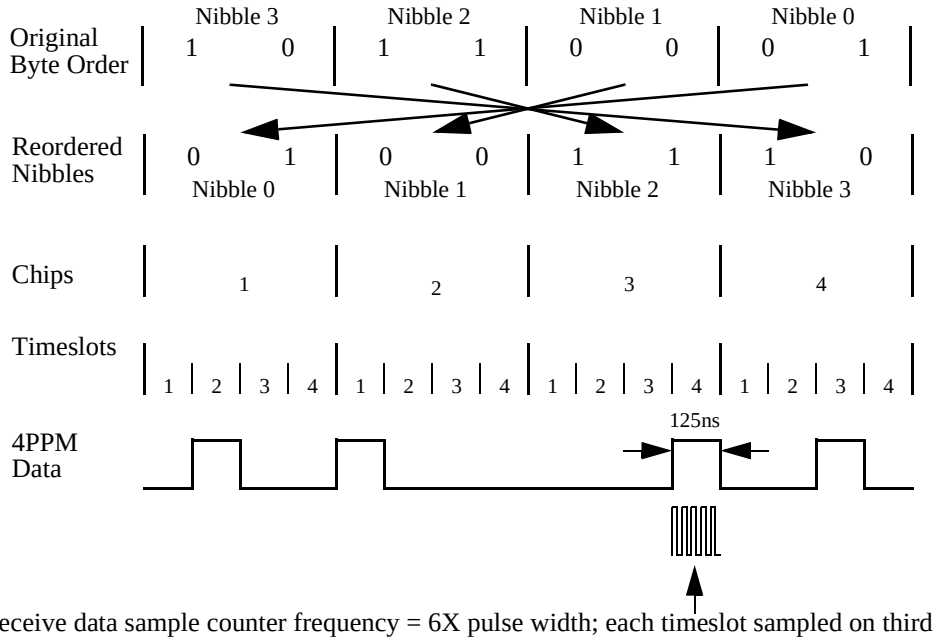


Figure 11-27. 4PPM Modulation Example



Receive data sample counter frequency = 6X pulse width; each timeslot sampled on third clock.

11.10.2.2 HSSP Frame Format

When the 4-Mbps transmission rate is used, the high-speed serial/parallel (HSSP) interface within the ICP is used along with the 4PPM bit encoding. The high-speed frame format shown in Figure 11-28 is similar to serial port 1's SDLC format with several minor modifications: the start/stop flags and CRC are twice as long, and instead of one start flag, a preamble and start flag of differing lengths are used.

Figure 11-28. High-Speed Serial Frame Format for IrDA Transmission (4.0 Mbps)

64 chips	8 chips	4 chips (8 bits)	4 chips (8 bits)	8180 chips max (2045 bytes)	16 chips (32 bits)	8 chips
Preamble	Start Flag	Address	Control (optional)	Data	CRC-32	Stop Flag

Start Flag	|0000|1100|0000|1100|0110|0000|0110|0000|	
	|0000|1100|0000|1100|0000|0110|0000|0110|	Stop Flag
Preamble	|1000|0000|1010|1000|... repeated 16 times	

The preamble, start, and stop flags are a mixture of chips that contain either 0, 1, or 2 pulses within the four timeslots. Chips with 0 and 2 pulses are used to construct flags because they represent invalid data bit pairings (one pulse required per chip to represent one of four bit pairs). The preamble contains 16 repeated transmissions of the four chips: 1000 0000 1010 1000; the start flag contains one transmission of eight chips: 0000 1100 0000 1100 0110 0000 0110 0000; and the stop flag contains one transmission of eight chips: 0000 1100 0000 1100 0000 0110 0000 0110. The address, control, data, and CRC-32 use the standard 4PPM chip encoding to represent 2 bits per chip.

11.10.2.3 Address Field

The 8-bit address field is used by a transmitter to target a select group of receivers when multiple stations are connected to the same set of serial lines. The address allows up to 255 stations to be uniquely addressed (00000000 to 11111110). The global address (11111111) is used to broadcast messages to all stations. Serial port 1 contains an 8-bit register, which is used to program a unique address for broadcast recognition, as well as a control bit to enable/disable the address match function. Note that the address of received frames is stored in the receive FIFO along with normal data and that it is transmitted and received starting with its LSB and ending with its MSB.

11.10.2.4 Control Field

The IPC control field is 8 bits and is optional (as defined by the user). Serial port 2 does not provide any hardware decode support for the control byte, but instead treats all bytes between the address and the CRC as data. Note that the control field is transmitted and received starting with its LSB and ending with its MSB.

11.10.2.5 Data Field

The data field can be any length that is a multiple of 8 bits from 0 to 2045 bytes. The user determines the data field length according to the application requirements and transmission characteristics of the target system. Usually a length is selected that maximizes the amount of data that can be transmitted per frame while allowing the CRC checker to be able to consistently detect all errors during transmission. Note that serial port 2 does not contain any hardware that restricts the maximum amount of data transmitted or received. It is up to the user to maintain these limits. If a data field that is not a multiple of 8 bits is received, an abort is signalled. Also note that each byte within the data field is transmitted and received starting with its LSB and ending with its MSB.

11.10.2.6 CRC Field

The HSSP uses the established 32-bit cyclic redundancy check (CRC-32) to detect bit errors that occur during transmission. A 32-bit CRC is computed using the address, control, and data fields, and is included in each frame. A separate CRC generator is implemented in both the transmit and receive logic. The transmitter calculates a CRC, and while data is actively transmitted, places the inverse of the resultant 32-bit value at the end of each frame before the flag is transmitted. In a similar manner, the receiver also calculates a CRC for each received data frame and compares the calculated CRC to the expected CRC value contained within the end of each received frame. If the calculated value does not match the expected value, an interrupt is signalled. The CRC computation logic is preset to all ones before reception or transmission of each frame and the result is inverted before it is used for comparison or transmission. Note that unlike the address, control, and data fields, the 32-bit inverted CRC value is transmitted and received from least significant byte to most significant, and within each byte the least significant nibble or chip is encoded or decoded first. The cyclic redundancy checker uses the 32-term polynomial:

$$CRC(x) = (x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1)$$

11.10.2.7 Baud Rate Generation

The baud rate is derived by dividing down a fixed 48-MHz clock generated by one of the two on-chip PLLs by six. The 8-MHz baud (or timeslot) clock for the receive logic is synchronized with the 4PPM data stream each time a transition is detected on the receive data line using a digital PLL. To encode a 4-Mbps data stream, the required “chip” frequency is 2.0 MHz, with four timeslots per chip at a frequency of 8.0 MHz. Receive data is sampled halfway through each time-slot period by counting three out of the six 48-MHz clock periods that make up each timeslot (see Figure 11-27). The chips are synchronized during preamble reception. The repeating pattern (four chips repeated 16 times) is used to identify the first timeslot or beginning of a chip and resets the 2-bit time-slot counter logic, such that the 4PPM data is properly decoded.

11.10.2.8 Receive Operation

The IrDA standard specifies that all transmission occurs at half-duplex. This restriction forces the user to enable one direction at a given time: either the transmit or receive logic, but not both. However, the HSSP’s hardware does not impose such a restriction.The user may enable both the transmitter and receiver at the same time. Although forbidden by the IrDA standard, this feature is particularly useful when using the ICP’s loopback mode, which internally connects the output of the transmit serial shifter to the input of the receive serial shifter.

After the ICP is enabled for 4-Mbps transmission, the receiver logic begins by selecting an arbitrary chip boundary, receives four incoming 4PPM chips from the RXD2 pin using a serial shifter, and latches and decodes the chips one at a time. If the chips do not decode to the correct preamble, the time-slot counter’s clock is forced to skip one 8-MHz period, effectively delaying the time-slot count by one. This process is repeated until the preamble is recognized, signifying that the time-slot counter is synchronized. The preamble can be repeated as few as 16 times or may be continuously repeated to indicate an idle receive line.

At any time after the transmission of 16 preambles, the start flag can be received. The start flag is eight chips long. If any portion of the start flag does not match the standard encoding, the receive logic signals a framing error and the receive logic once again begins to look for the frame preamble.

Once the correct start flag is recognized, each subsequent grouping of four chips is decoded into a data byte and placed within a 5-byte temporary FIFO, which is used to prevent the CRC from being placed within the receive FIFO. When the temporary FIFO is filled, data values are pushed out one by one to the receive FIFO. The first data byte of a frame is the address. If receiver address matching is enabled, the received address is compared to the address programmed in the address match value field in one of the control registers. If the two values are equal or if the incoming address contains all ones, all subsequent data bytes, including the address byte, are stored in the receive FIFO. If the values do not match, the receiver logic does not store any data in the receive FIFO, ignores the remainder of the frame, and begins to search for the next preamble. The second data byte of the frame can contain an optional control field as defined by the user and must be decoded in software (no hardware support within the HSSP).

Frames can contain any amount of data in multiples of 8 bits up to a maximum of 2047 bytes (including the address and control bytes). The HSSP does not limit frame size; it is the responsibility of the user to check that the size of each incoming frame does not exceed the IrDA protocol’s maximum allowed frame size.

When the receive FIFO is one- to two-thirds full, an interrupt or DMA transfer is signalled. If the data is not removed soon enough and the FIFO is completely filled, an overrun error is signalled when the receive logic attempts to place additional data into the full FIFO. Once the FIFO is full, all subsequent data bytes received are lost while all FIFO contents remain intact.

If any two sequential chips within the data field do not contain pulses (are 0000), the frame is aborted, the least recent or oldest byte within the temporary FIFO is moved to the receive FIFO (the remaining four FIFO entries are discarded), the end-of-frame (EOF) tag is set within the same FIFO entry where the last “good” byte of data resides, and the receiver logic begins to search for the preamble. An abort also occurs if any data chip containing 0011, 1010, 0101, or 1001 occurs (invalid chips that do not occur in the stop flag).

The receive logic continuously searches for the 8-chip stop flag. Once it is recognized, the last byte that was placed within the receive FIFO is flagged as the last byte of the frame and the data in the temporary FIFO is removed and used as the 32-bit CRC value for the frame. Instead of placing this in the receive FIFO, the receive logic compares it to the CRC-32 value, which is continuously calculated using the incoming data stream. If they do not match, the last byte that was placed within the receive FIFO is also tagged with a CRC error. The CRC value is not placed in the receive FIFO.

If the user disables the HSSP’s receiver during operation, reception of the current data byte is stopped immediately, the serial shifter and receive FIFO are cleared, control of the RXD2 pin is given to the peripheral pin control (PPC) unit, and all clocks used by the receive logic are automatically shut off to conserve power. The user should ensure that the polarity of the RXD2 input is reprogrammed properly if this pin is to be used as a GPIO input.

11.10.2.9 Transmit Operation

Before enabling the HSSP for transmission, the user may either “prime” the transmit FIFO by filling it with data or allow service requests to cause the CPU or DMA to fill the FIFO once the HSSP is enabled. Once enabled, the transmit logic issues a service request if its FIFO is empty. For each frame output, a minimum of 16 preambles are transmitted. If data is not available after the sixteenth preamble, additional preambles are output until a byte of valid data resides within the bottom of the transmit FIFO. The preambles are then followed by the start flag and then the data from the transmit FIFO. Four chips (8 bits) are encoded at a time and then loaded into a serial shift register. The contents are shifted out onto the TXD2 pin clocked by the 8-MHz baud clock. Note that the preamble, start and stop flags, and CRC value are automatically transmitted and need not be placed in the transmit FIFO.

When the transmit FIFO is emptied halfway, an interrupt and/or DMA service request is signalled. If new data is not supplied soon enough, the FIFO is completely emptied, and the transmit logic attempts to take additional data from the empty FIFO (one of two actions can be taken as programmed by the user). An underrun can either signal the normal completion of a frame or an unexpected termination of a frame in progress.

When normal frame completion is selected and an underrun occurs, the transmit logic transmits the 32-bit CRC value calculated during the transmission of all data within the frame (including the address and control bytes), followed by the stop flag to denote the end of the frame. The transmitter then continuously transmits preambles until data is once again available within the FIFO. Once data is available, the transmitter begins transmission of the next frame.

When unexpected frame termination is selected and an underrun occurs, the transmit logic outputs an abort and interrupts the CPU. An abort continues to be transmitted until data is once again available in the transmit FIFO. The HSSP then transmits 16 preambles, a start flag, and starts the new frame. The off-chip receiver can choose to ignore the abort and continue to receive data or signal the HSSP to retry transmission of the aborted frame.

At the end of each frame transmitted, the HSSP outputs a pulse called the serial infrared interaction pulse (SIP). A SIP is required at least every 500 ms to keep slower speed devices (115.2 Kbps and slower) from colliding with the higher speed transmission. The SIP simulates a start bit that causes all low-speed devices to stay off the bus for at least another 500 ms. Transmission of the SIP pulse causes the **TXD2** pin to be forced high for a duration of 1.625 µs and low for 7.375 µs (total SIP period = 9.0 µs). After the 9.0 µs elapses, the preamble is then transmitted continuously to indicate to the off-chip receiver that the HSSP's transmitter is in the idle state. The preamble continues to be transmitted until new data is available within the transmit FIFO, or the HSSP's transmitter is disabled. Note that it is the responsibility of the user to ensure that a frame completes once every 500 ms such that a SIP pulse is produced, keeping all low-speed devices from interrupting transmission. Because most IrDA compatible devices produce a SIP after each frame transmitted, the user only needs to ensure that a frame is either transmitted or received by the ICP every 500 ms. Note that frame length does not represent a significant portion of the 500 ms timeframe in which a SIP must be produced. At 4.0 Mbps, the longest frame allowed is 16,568 bits, which takes just over 4 ms to transmit. Also note that the HSSP issues a SIP when the transmitter is first enabled to ensure all low-speed devices are silenced before transmitting its first frame.

If the user disables the HSSP's transmitter during operation, transmission of the current data byte is stopped immediately, the serial shifter and transmit FIFO are cleared, control of the TXD2 pin is given to the peripheral pin control (PPC) unit, and all clocks used by the transmit logic are automatically shut off to conserve power. The user should ensure that the polarity of the TXD2 output is reprogrammed properly if this pin is to be used as a GPIO output.

11.10.2.10 Transmit and Receive FIFOs

To reduce chip size and power consumption, the HSSP's FIFOs use self-timed logic (they are not clocked). Because of process and environmental variations, the depth at which a service request is triggered to empty the receive FIFO is variable. This variation spans a maximum of four FIFO entries; the receive FIFO service request can be made at four different FIFO depths.To compensate for this variability and guarantee that at least eight valid entries of data exist within the FIFO before generating a service request, an extra four entries have been added to the receive FIFO (four entries more than the transmit FIFO). The transmit FIFO is 16 entries deep and the receive FIFO is 20 entries deep. The point at which the receive FIFO service request is triggered spans one fifth (four entries) of the 20-entry FIFO. The service request is signalled at a depth from two-fifths full to three-fifths full (when the FIFO contains nine, ten, eleven, or twelve entries of data).

This service request variation applies only to an empty FIFO that is filled (receive FIFO). It does not apply to a full FIFO that is emptied (transmit FIFO). The transmit FIFO is guaranteed to signal a service request when it has eight or more empty entries and negate the request when the FIFO contains nine or more entries that are filled.

If the DMA is used to service either one or both of the HSSP's FIFOs, the burst size must be set to eight words, even though more than eight entries of data may exist within the receive FIFO. If programmed I/O is used to service the FIFOs, a maximum of 8 words may be added to the transmit FIFO without checking if more space is available. Likewise, a maximum of 8 words may be removed from the receive FIFO without checking if more data is available. After this point, the user must poll a set of status bits that indicate if any data remains in the receive FIFO or if space is available in the transmit FIFO before emptying or filling the FIFOs any further.

11.10.2.11 CPU and DMA Register Access Sizes

Bit positioning, byte ordering, and addressing of the SDLC is described in terms of little endian ordering. All ICP (HSSP and UART) registers are 8 bits wide and are located in the least significant byte of individual words. The ARM peripheral bus does not support byte or half-word

operations. All reads and writes of the ICP by the CPU should be wordwide. Two separate, dedicated DMA requests exist for both the transmit and the receive FIFOs. If the DMA controller is used to service the transmit and/or receive FIFOs, the user must ensure the DMA is properly configured to perform bytewide accesses, using 8 bytes per burst for the HSSP and 4 bytes per burst for the UART. See later sections in this chapter for summaries of the ICP's UART registers and HSSP registers.

11.10.3 UART Register Definition

The ICP's UART is the same as serial port 3's UART except that one additional register exists to control HP-SIR modulation for low-speed operation. See Section 11.11, "Serial Port 3 - UART" on page 11-128 for a description of the programming and operation of all other features of the ICP's UART. Note that the user must ensure that the UART is programmed to yield the frame format shown in Figure 11-25.

11.10.4 UART Control Register 4

UART control register 4 (UTCR4) contains two different bit fields that control various functions for 115.2-Kbps (low-speed) IrDA transmission.

11.10.4.1 HP-SIR Enable (HSE)

The HP-SIR enable (HSE) bit controls whether the HP-SIR bit modulation logic is enabled or disabled. When HSE=0, HP-SIR modulation is disabled, and if UART operation is enabled (ITR=0), it is used for normal serial transmission (NRZ encoding only) rather than IrDA communication. When HSE=1, HP-SIR modulation is enabled for low-speed IrDA communication; zeros are represented by pulses that are 3/16 of the programmed bit width, while ones are represented by no pulses.

11.10.4.2 Low-Power Mode (LPM)

The low-power mode (LPM) bit controls whether the HP-SIR bit modulation logic represents zeros using a pulse that is 3/16 of the chosen bit width or a fixed 1.6 µs pulse width. When LPM=0, zeros are encoded as a pulse, which is 3/16 of the bit width programmed within the UART's baud rate divisor (BRD) bit field. When LPM=1, the UART's programmed bit length is ignored and zeros are represented by pulses that are 1.6 µs in duration. Programming LPM=1 minimizes the time that the off-chip LED transceiver is turned on to the minimum pulse width specified by the IrDA low-speed standard, which in turn, minimizes power consumption.

The following table shows the location of the bits within UART control register 4; question marks indicate that the values are unknown at reset. Both bits are reset to zero. Note that the UART must be disabled (RXE=TXE=0) when changing the state of either of these two bits. Also note that writes to reserved bits are ignored and reads return zeros.

Address: 0h 8003 0010 **UTCR4** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	Reserved						LPM	HSE
Reset	0	0	0	0	0	0	?	?

Bit	Name	Description
0	HSE	HP-SIR enable. 0 – HP-SIR modulation disabled; ICP functions as normal UART if ITR=0. 1 – HP-SIR modulation enabled; ICP functions as low-speed IrDA port if ITR=0.
1	LPM	Low-power mode. 0 – Each zero encoded as a pulse that is 3/16 of the programmed bit time if ITR=0. 1 – Each zero encoded as a pulse that is 1.6 µs wide if ITR=0.
7..2	—	Reserved.

11.10.5 HSSP Register Definitions

There are six registers within the HSSP: three control registers, one data register, and two status registers. The control registers are used to select IrDA transmission rate, address match value, whether an abort or end of frame occurs when the transmit FIFO underruns, and true or complemented transmit and receive data; to enable or disable transmit and receive operation, the FIFO interrupt service requests, receive address matching, and loopback mode.

The data register addresses the top location of the transmit FIFO and bottom location of the receive FIFO. When it is read, the receive FIFO is accessed, and when it is written, the transmit FIFO is accessed.

The status registers contain bits that signal CRC, overrun, underrun, framing, and receiver abort errors as well as the transmit FIFO service request, receive FIFO service request, and end-of-frame conditions. Each of these hardware-detected events signals an interrupt request to the interrupt controller. The status registers also contain flags for transmitter busy, receiver synchronized, receive FIFO not empty, and transmit FIFO not full (no interrupt generated).

11.10.6 HSSP Control Register 0

The HSSP control register 0 (HSCR0) contains eight different bit fields that control various functions for 4 Mbps IrDA transmission.

11.10.6.1 IrDA Transmission Rate (ITR)

The IrDA transmission rate (ITR) bit is used to select the transmission speed of the ICP. ITR selects the correct type of IrDA bit modulation to use (HP-SIR or 4PPM), and enables the correct serial-to-parallel engine (UART or HSSP). When ITR=0, the HP-SIR modulator is enabled along with serial port 2's UART. When ITR=1, the 4PPM modulator is enabled as well as the HSSP. Note that ITR is the only control bit that affects both the UART and HSSP. Once one of the two speeds is selected, all further programming is controlled by the individual units (UART or HSSP).

11.10.6.2 Loopback Mode (LBM)

The loopback mode (LBM) bit is used to enable and disable the ability of the HSSP's transmit and receive logic to communicate. When LBM=0, the HSSP operates normally. The transmit and receive data paths are independent and communicate via their respective pins. When LBM=1, the output of the transmit serial shifter is directly connected to the input of the receive serial shifter internally, and control of the TXD2 and RXD2 pins is given to the peripheral pin control (PPC) unit. Note that even though the IrDA standard permits only half-duplex operation, the HSSP does not restrict the user from transmitting and receiving data at the same time; both are fully independent units. This function is essential when using the HSSP in loopback mode.

11.10.6.3 Transmit FIFO Underrun Select (TUS)

The transmit FIFO underrun select (TUS) bit is used both to select what action to take as a result of a transmit FIFO underrun as well as mask or enable the transmit FIFO underrun interrupt.

When TUS=0, transmit FIFO underruns are used to signal the transmit logic that the end of the frame has been reached. When the transmit FIFO experiences an underrun, the CRC value, which is calculated continuously on outgoing data, is loaded to the serial shifter and transmitted, followed by the stop flag and SIP pulse. Also when TUS=0, the transmit FIFO interrupt is masked and the state of the transmit FIFO underrun (TUR) status bit is ignored by the interrupt controller.

When TUS=1, transmit FIFO underruns are used to signal the transmit logic that the end of the frame has not yet been reached. When the transmit FIFO experiences an underrun, the CRC value, which is calculated continuously on outgoing data, is loaded to the serial shifter and transmitted, followed by the stop flag and SIP pulse. Additionally, when TUS=0, the transmit FIFO underrun interrupt is masked, causing the state of the transmit FIFO underrun (TUR) status bit to be ignored by the interrupt controller. Note that programming TUS=0 does not affect the current state of TUR or the transmit FIFO logic's ability to set and clear TUR; it only blocks the generation of the interrupt request.

When TUS=1, transmit FIFO underruns are used to signal the transmit logic that the end of the frame has not yet been reached and that the rate in which data is supplied to the transmit FIFO is not sufficient. When the transmit FIFO experiences an underrun, two sequential chips, each containing zeros (0000), are output by the transmitter to signal an abort condition; next a SIP pulse is output, followed by a minimum of 16 preambles. Preambles continue to be output until data is once again available within the transmit FIFO. Additionally, when TUS=1, the transmit FIFO underrun interrupt is enabled, and whenever TUR is set (one), an interrupt request is made to the interrupt controller. To change the state of TUS during operation, the user should fill the transmit FIFO to ensure TUS is not written at the same time that the transmit FIFO underruns.

TUS is useful for ensuring that frames are not prematurely ended due to an unexpected transmit FIFO underrun. At the start of a frame, the user can configure TUS=1 such that any underrun signals an abort to the off-chip receiver. Just before the end of the frame, the user can then configure TUS=0, allowing the remaining data to be output by the transmit logic. The FIFO then underruns, causing the CRC, stop flag, and SIP to be transmitted.

11.10.6.4 Transmit Enable (TXE)

The transmit enable (TXE) bit is used to enable and disable HSSP transmit operation. When TXE=0, the transmit logic is disabled and its clocks are turned off to conserve power. When TXE=1, the HSSP transmitter logic is enabled for IrDA transmission. It is required that the user first program all other control bits before setting TXE. If the TXE bit is cleared to zero while the HSSP is actively transmitting data, transmission is stopped immediately, all data within the transmit FIFO and serial output shifter is cleared, and control of the TXD2 pin is given to the peripheral pin control (PPC) unit. When the transmitter is turned on (TXE=0→1), a SIP pulse is transmitted before transmission of data. A SIP pulse is used to prevent slower devices (115.2 Kbps) from attempting to take control of infrared transmission. See the previous sections for further timing details of the SIP pulse.

TXE and RXE are the only control bits within the HSSP that are initialized when a hardware reset occurs. Clearing TXE to zero ensures the HSSP transmitter is disabled, giving control of the transmit pin to the PPC unit that configures TXD1 as an input following a reset of the SA-1100. Note that TXE is ignored when ITR=0 (enables UART operation). Also note that even though the IrDA standard permits only half-duplex operation, the HSSP does not restrict the user from

transmitting and receiving data at the same time; both are fully independent units. This function is particularly useful when using the HSSP in loopback mode. See the Section 11.10.6.2, “Loopback Mode (LBM)” on page 11-112.

11.10.6.5 Receive Enable (RXE)

The receive enable (RXE) bit is used to enable or disable HSSP receive operation. When RXE=0, the receive logic is disabled and its clocks are turned off to conserve power. When RXE=1, the HSSP receiver logic is enabled for IrDA reception. It is required that the user first program all other control bits before setting RXE. If the RXE bit is cleared to zero while the HSSP is actively receiving data, reception is stopped immediately, all data within the receive FIFO and serial input shifter is cleared, and control of the RXD2 pin is given to the peripheral pin control (PPC) unit. Note that TXE and RXE are the only control bits within the HSSP that are initialized when a hardware reset occurs. Clearing RXE to zero ensures the HSSP receiver is disabled, giving control of the receive pin to the PPC unit, which configures RXD2 as an input following a reset of the SA-1100. Note that RXE is ignored when ITR=0, which enables UART operation. Also note that even though the IrDA standard permits only half-duplex operation, the HSSP does not restrict the user from transmitting and receiving data at the same time; both are fully independent units. This function is particularly useful when using the HSSP in loopback mode. See the Section 11.10.6.2, “Loopback Mode (LBM)” on page 11-112.

11.10.6.6 Receive FIFO Interrupt Enable (RIE)

The receive FIFO interrupt mask (RIE) bit is used to mask or enable the receive FIFO service request interrupt. When RIE=0, the interrupt is masked, and the state of the receive FIFO service request (RFS) bit within HSSP status register 0 is ignored by the interrupt controller. When RIE=1, the interrupt is enabled, and whenever RFS is set (one), an interrupt request is made to the interrupt controller. Note that programming RIE=0 does not affect the current state of RFS or the receive FIFO logic’s ability to set and clear RFS; it only blocks the generation of the interrupt request. Also note that RIE does not affect generation of the receive FIFO DMA request , which is asserted whenever RFS=1.

11.10.6.7 Transmit FIFO Interrupt Enable (TIE)

The transmit FIFO interrupt mask (TIE) bit is used to mask or enable the transmit FIFO service request interrupt. When TIE=0, the interrupt is masked and the state of the transmit FIFO service request (TFS) bit within HSSP status register 0 is ignored by the interrupt controller. When TIE=1, the interrupt is enabled, and whenever TFS is set (one), an interrupt request is made to the interrupt controller. Note that programming TIE=0 does not affect the current state of TFS or the transmit FIFO logic’s ability to set and clear TFS; it only blocks the generation of the interrupt request. Also note that TIE does not affect generation of the transmit FIFO DMA request, which is asserted whenever TFS=1.

11.10.6.8 Address Match Enable (AME)

The address match enable (AME) bit is used to enable or disable the receive logic from comparing the address programmed in the address match value (AMV) bit field to the address of all incoming frames. When AME=1, data is stored in the receive FIFO only for those frames that have addresses that match AMV and for any frame that contains an address containing all ones (11111111), denoting a global address. For frames in which the address does not match, the data and CRC are ignored and the receiver resumes hunting for a preamble. When AME=0, address values are not compared and the data in every frame is stored in the receive FIFO.

The following table shows the location of the bits within HSSP control register 0. RXE and TXE are the only control bits that are reset to a known state to ensure the HSSP is disabled following a reset of the SA-1100. The reset state of all other control bits is unknown (indicated by question marks) and must be initialized before enabling the HSSP. Note that the HSSP must be disabled (RXE=TXE=0) when changing the state of bits 0 and 1, and bits 2 through 7 may be written while the HSSP is enabled to allow various modes to be changed during active operation.

Address: 0h 8004 0060 **HSCR0** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	AME	TIM	RIM	RXE	TXE	TUS	LBM	ITR
Reset	?	?	?	0	0	?	?	?

Bit	Name	Description
0	ITR	IrDA transmission rate. 0 – 115.2 Kbps (selects HP-SIR modulation, enables the ICP's UART engine). 1 – 4.0 Mbps (selects 4PPM modulation, enables the ICP's HSSP engine).
1	LBM	Loopback mode. 0 – Normal serial port operation enabled. 1 – Output of HSSP's transmit serial shifter is connected to input of receive serial shifter internally. Control of TXD2 and RXD2 pins is given to the PPC unit if ITR=1.
2	TUS	Transmit FIFO underrun select. 0 – Transmit FIFO underrun causes CRC, stop flag, and SIP to be transmitted, and masks transmit underrun interrupt generation (TUR ignored). 1 –Transmit FIFO underrun causes an abort to be transmitted, and generates an interrupt (state of TUR sent to interrupt controller).
3	TXE	Transmit enable. 0 – HSSP transmit logic disabled; control of the TXD2 pin is given to the PPC unit if ITR=1. 1 – HSSP transmit logic enabled if ITR=1. **Note:** A SIP is transmitted immediately after the transmitter is enabled (TXE = 0 $\rightarrow$ 1).
4	RXE	Receive enable. 0 – HSSP receive logic disabled; control of the RXD2 pin is given to the PPC unit if ITR =1. 1 – HSSP receive logic enabled if ITR=1.
5	RIE	Receive FIFO interrupt enable. 0 – Receive FIFO two- or three-fifths full or more condition does not generate an interrupt (RFS bit ignored). 1 – Receive FIFO two- or three-fifths full or more condition generates an interrupt (state of RFS sent to interrupt controller).
6	TIE	Transmit FIFO interrupt enable. 0 – Transmit FIFO half-full or less condition does not generate an interrupt (TFS bit ignored). 1 – Transmit FIFO half-full or less condition generates an interrupt (state of TFS sent to interrupt controller).
7	AME	Address match enable. 0 – Disable receiver address match function, store data from all incoming frames in receive FIFO. 1 – Enable receiver address match function; do not FIFO data unless address recognized or incoming address contains all ones (0hFF).

11.10.7 HSSP Control Register 1

HSSP control register 1 (HSCR1) contains the 8-bit address match value field that is used by the HSSP to selectively receive frames.

11.10.7.1 Address Match Value (AMV)

The 8-bit address match value (AMV) field is programmed with an address value that is used to selectively store only the data within receive frames that have the same address value. The address match enable (AME) bit must be set to enable this function. For incoming frames, which have the same address value as the AMV field, the frame's address, control, and data are stored in the receive FIFO. For those that do not, the remainder of the frame is ignored and the receive logic switches to hunt mode, looking for the preamble in the incoming data stream. One special address exists, which is always matched by the address match logic regardless of the value programmed in AMV. When address matching is enabled, whenever a frame is received with an address containing all ones (11111111), the value programmed in AMV is ignored and the frame data is automatically stored in the receive FIFO. The address value is contained within the first byte of data in a frame following the flag. AMV can be written at any time and is used for comparison with the next frame, which occurs following its update.

The following table shows the address match value field within HSSP control register 1. The reset state of AMV is unknown (indicated by question marks) and must be initialized before enabling the HSSP. Note that HSCR1 may be written while the HSSP is enabled to allow the address match value to be changed during active receive operation.

Address: 0h 8004 0064 **HSCR1** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	AMV							
Reset	?	?	?	?	?	?	?	?

Bit	Name	Description
7..0	AMV	Address match value. The 8-bit value used by receiver logic to compare to address of incoming frames. If AME=1 and AVM matches the address of the incoming frame, store the frame address, control, and data in receive FIFO; if address does not match, ignore the frame and search for the next preamble. **Note:** An address of 0hFF (all ones) in the incoming frame automatically generates a match (AMV is ignored).

11.10.8 HSSP Control Register 2

The HSSP control register 2 (HSCR2) contains two bit-fields that control the polarity of the transmit and receive data pins. Note that unlike the rest of the HSSP's registers, its bits are located in byte 2 of the addressed word (bits 23..16). Word reads or writes should be used to access this register. Also note that this register resides within the PPC's address space.

11.10.8.1 Transmit Pin Polarity Select (TXP)

The transmit pin polarity select (TXP) bit is used to select whether data output to the ICP's transmit pin (TXD2) is true or complemented. When TXP=0, data output from the UART (low-speed mode), HSSP (high-speed mode), or PPC (GPIO output mode) is inverted first before being output to the TXD2 pin. When TXP=1, data output from either the UART, HSSP, or PPC to the TXD2 pin is true or noninverted. TXP is initialized to 1 following reset such that output pin data defaults to true data.

Note that TXP affects the TXD2 pin during all modes of operation including HSSP, UART, and PCC. The user should ensure that this bit is properly programmed when using serial port 2 for high- or low-speed IrDA, normal UART, or GPIO operation. Note that for GPIO mode, the user needs to configure TXP only when the pin is to be used as an output (PPDR<14>=1). When used as a GPIO input, TXP has no effect on the state of TXD2. See the Peripheral Pin Controller chapter.

Additionally, the user must ensure that the PPC sleep state direction bit for TXD2 is inverted from its normal value, if TXP=0 indicating inverted data. Thus if the user wishes to make TXD2 an output in sleep mode, but TXP=0 indicating the output is inverted, the PPC should be programmed such that PSDR<14>=1. Likewise, if TXP=0 and the user wishes to make TXD2 an input in sleep mode, the PPC should be programmed such that PSDR<14>=0. If TXP=1 indicating true data, PSDR should be programmed normally.

11.10.8.2 Receive Pin Polarity Select (RXP)

The receive pin polarity select (RXP) bit is used to select whether data input to the ICP's receive pin (RXD2) is viewed by the ICP as true or complemented. When RXP=0, data input from the RXD2 pin is first inverted before being sent to either the UART (low-speed mode), HSSP (high-speed mode), or PPC (GPIO input mode). When RXP=1, data input from the RXD2 pin is treated as true data and is not inverted before being sent to either the UART, HSSP, or PPC. RXP is initialized to 1 following reset such that input pin data defaults to true data.

Note that RXP affects the RXD2 pin during all modes of operation including HSSP, UART, and PCC. The user should ensure that this bit is properly programmed when using serial port 2 for high- or low-speed IrDA, normal UART, or GPIO operation. Note that for GPIO mode, the user needs to configure RXP only when the pin is to be used as an input (PPDR<15>=0). When used as a GPIO output, RXP has no effect on the state of RXD2.

Also note that, unlike the TXP bit, RXP has no effect on the PPC sleep state direction bit for RXD2. PSDR<15> should be programmed normally.

The following table shows the location of the bits within HSSP control register 2. Both bits are set to one to ensure serial port 2's pins default to normal "true" data operation following a reset of the SA-1100. Note that the HSSP and UART must be disabled (RXE=TXE=0) when changing the state of these bits. Also note that reads of reserved bits return zero and writes have no effect.

Address: 0h 9006 0028 **HSCR2** **Read/Write**

Bit	23	22	21	20	19	18	17	16
	Reserved				RXP	TXP	Reserved	
Reset	0	0	0	0	1	1	0	0

Bit	Name	Description
17..16	—	Reserved.
18	TXP	Transmit pin polarity select. 0 – Data output from the HSSP, UART, or PPC is first inverted before being output to TXD2. 1 – Data output from the HSSP, UART, or PPC to TXD2 is true or non-inverted data.
19	RXP	Receive pin polarity select. 0 – Data input from RXD2 is first inverted before being used by the HSSP, UART, or PPC. 1 – Data input from RXD2 to the HSSP, UART, or PPC is true or non-inverted data.
23..20	—	Reserved.

11.10.9 HSSP Data Register

The HSSP data register (HSDR) is an 8-bit register corresponding to both the top and bottom entry of the transmit and receive FIFOs, respectively.

When HSDR is read, the lower 8 bits of the bottom entry of the 11-bit receive FIFO is accessed. As data enters the top of the receive FIFO, bits 8 – 10 are used as tags to indicate various conditions that occur during reception of each piece of data. The tag bits are transferred down the FIFO along with the data byte that encountered the condition. When data reaches the bottom, bit 8 of the bottom FIFO entry is automatically transferred to the end-of-frame (EOF) flag, bit 9 to the CRC error (CRE) flag, and bit 10 to the receiver overrun (ROR) flag, all within HSSP status register 1. The user can read these flags to determine if the value at the bottom of the FIFO represents the last byte within the frame or if an error was encountered during reception. After checking the flags, the FIFO value can then be read, which causes the data in the next location of the receive FIFO to automatically transfer down to the bottom entry and its EOF/CRE/ROR bits to be transferred to the status register.

The end/error in FIFO (EIF) flag is set within status register 0 whenever one or more of the tag bits (8 – 10) are set within any of the bottom eight entries of the receive FIFO and is cleared when no error bits are set in the bottom eight entries of the FIFO. When EIF is set, an interrupt is generated and receive FIFO DMA requests are disabled so that the user can manually empty the FIFO, always checking the end-of-frame, CRC error, and overrun error flags in status register 1 first before removing each data value from the FIFO. After each entry is removed, the user should check the EIF bit to see if any set end or error tag remains, and repeat the procedure until all set tags are flushed from the bottom eight entries of the FIFO. Once EIF is cleared, servicing of the receive FIFO by the DMA controller is automatically reenabled.

When HSDR is written, the topmost entry of the 8-bit transmit FIFO is accessed. After a write, data is automatically transferred down to the lowest location within the transmit FIFO, which does not already contain valid data. Data is removed from the bottom of the FIFO one piece at a time by the transmit logic, encoded using the 4PPM modulation technique, loaded into the transmit serial shifter, then serially shifted out onto the TXD2 pin.

The following table shows the bit locations corresponding to the data field, end-of-frame bit as well as the cyclic redundancy check and receiver overrun error bits within the HSSP data register. Note that both FIFOs are cleared when the SA-1100 is reset, the transmit FIFO is cleared when TXE=0, and the receive FIFO is cleared when RXE=0.

Address: 0h 8004 006C **HSDR** **Read/Write**

Bit	10	9	8	7	6	5	4	3	2	1	0
	ROR	CRE	EOF	Bottom of receive FIFO data							
Reset	0	0	0	0	0	0	0	0	0	0	0

Read Access

(**Note:** ROR, CRE, EOF are not read, but rather transferred to corresponding status bits in the HSSP status register 1(HSSR1) each time a new data value is transferred to HSDR).

Bit	7	6	5	4	3	2	1	0
	Top of transmit FIFO data							
Reset	0	0	0	0	0	0	0	0

Write Access

Bit	Name	Description
7..0	DATA	Top/bottom of transmit/receive FIFO data. Read – Bottom of receive FIFO. Write –Top of transmit FIFO.
8	EOF	End of frame. 0 – The last byte of the frame has not been encountered. 1 – The data value at the bottom of the receive FIFO represents the last byte of the frame. **Note:** Each time an 11-bit value reaches the bottom of the receive FIFO, bit 8 from the last FIFO entry is transferred to the EOF bit in HSSR1.
9	CRE	CRC error. 0 – CRC not encountered yet, or the CRC value calculated on the incoming data matched the received CRC value. 1 – The CRC value calculated on the incoming data did not match the received CRC value. **Note:** Each time an 11-bit value reaches the bottom of the receive FIFO, bit 9 from the last FIFO entry is transferred to the CRE bit in HSSR1.
10	ROR	Receiver overrun. 0 – No receiver overrun has been detected. 1 – Receive logic attempted to place data into receive FIFO while it was full; one or more data values *after* the data value at the bottom of the receive FIFO were lost. **Note:** Each time an 11-bit value reaches the bottom of the receive FIFO, bit 10 from the last FIFO entry is transferred to the ROR bit in HSSR1.

11.10.10 HSSP Status Register 0

HSSP status register 0 (HSSR0) contains bits that signal the transmit FIFO service request, receive FIFO service request, receiver abort, transmit FIFO underrun, framing error, and the end/error in receive FIFO conditions. Each of these hardware-detected events signal an interrupt request to the interrupt controller.

A bit that can cause an interrupt signals the interrupt request as long as the bit is set. Once the bit is cleared, the interrupt is cleared. Read/write bits are called status bits; read-only bits are called flags. Status bits are referred to as “sticky” (once set by hardware, must be cleared by software). Writing a one to a sticky status bit clears it; writing a zero has no effect. Read-only flags are set and cleared by hardware; writes have no effect. Additionally, some bits that cause interrupts have corresponding mask bits in the control registers and are indicated in the following sections. Note that the user has the ability to mask all HSSP interrupts by clearing bit 16 within the interrupt controller mask register (ICMR).

11.10.10.1 End/Error in FIFO Status (EIF) (read-only, nonmaskable interrupt)

The end/error in FIFO flag (EIF) is a read-only bit that is set when any tag bits (8 through 10) are set within the bottom eight entries of the receive FIFO and is cleared when no tag bits are set within the bottom eight entries of the FIFO. When EIF is set, an interrupt is signalled and DMA requests to empty the receive FIFO are disabled until EIF is cleared. To discover which FIFO entry contains the end-of-frame or an error condition, the user should check the state of the EOF, CRE, and ROR bits (described in the following sections), then read the corresponding value from the HSDR. This procedure should be repeated until EIF is cleared because set flag bits that are present within *any* of the eight lowest entries in the receive FIFO can set EIF. Once all tags are cleared from the bottom eight entries of the receive FIFO, EIF is automatically cleared, which in turn, clears the interrupt and reenables receive FIFO DMA requests.

11.10.10.2 Transmit Underrun Status (TUR) (read/write, maskable interrupt)

The transmit underrun status bit (TUR) is set when the transmit logic attempts to fetch data from the transmit FIFO after it has been completely emptied. When an underrun occurs, the transmitter takes one of two actions. When the transmit underrun select bit is clear (TUS=0), the transmitter ends the frame by shifting out the CRC that is calculated continuously on outgoing data, followed by a stop flag and SIP pulse. When TUS=1, the transmitter is forced to transmit an abort and continues to transmit chips containing all zeros (0000) until valid data is again available within the FIFO. Once data resides within the bottom entry of the transmit FIFO, a new data frame is initiated by transmitting 16 preambles and a start flag followed by the transmission of data from the FIFO. When the TUR bit is set, an interrupt request is made unless it is masked. When TUS=0, the interrupt is masked; when TUS=1, it is enabled. Note that underruns are not generated when the HSSP transmitter is first enabled and is in the idle state (continuously transmits flags).

11.10.10.3 Receiver Abort Status (RAB) (read/write, nonmaskable interrupt)

The receiver abort status bit (RAB) is set when an abort is detected during receipt of an incoming frame. An abort is signalled when two or more chips that do not contain any pulses (0000) or chips containing 0011, 1001, 1010, or 0101(invalid chips not contained within the stop flag) are detected after a valid start flag has been detected but before a complete stop flag has been received (an incorrect chip in the stop flag generates an abort as well). When an abort is received, the EOF tag is set in the FIFO entry that corresponds to the last piece of data received before the frame was aborted. The receiver then enters hunt mode, searching for the preamble.

11.10.10.4 Transmit FIFO Service Request Flag (TFS) (read-only, maskable interrupt)

The transmit FIFO service request flag (TFS) is a read-only bit that is set when the transmit FIFO is nearly empty and requires service to prevent an underrun. TFS is set any time the transmit FIFO has eight or fewer entries of valid data (half-full or less), and is cleared when it has nine or more entries of valid data. When the TFS bit is set, an interrupt request is made unless the transmit FIFO interrupt request mask (TIE) bit is cleared. The state of TFS is also sent to the DMA controller, and can be used to signal a DMA service request. Note that TIE has no effect on the generation of the DMA service request. After the DMA or CPU fills the FIFO, such that eight or more locations are filled within the transmit FIFO, the TFS flag (and the service request and/or interrupt) is automatically cleared.

11.10.10.5 Receive FIFO Service Request Flag (RFS) (read-only, maskable interrupt)

The receive FIFO service request flag (RFS) is a read-only bit that is set when the receive FIFO is nearly filled and requires service to prevent an overrun. The amount of data that causes RFS to be set is nondeterministic. However, the range in which RFS will be set is guaranteed. RFS is set at some point when the receive FIFO is two- to three-fifths full (or more). The HSSP's FIFOs are self-timed to reduce cost and save power. As a result, the depth at which the receive FIFO service request is generated is variable. This is the reason the receive FIFO is 20 entries deep instead of 16 like the transmit FIFO. At which entry in the FIFO the request is actually triggered is dependent on IC process, operating temperature, and so on. The receive FIFO is designed to signal the RFS bit to be set when it contains 12 entries of valid data. However, because of the variability of the self-timed logic, RFS may also be set when 11, 10, or 9 entries of valid data are present within the FIFO. Likewise, under normal circumstances, RFS is cleared when the receive FIFO has 11 remaining entries of valid data. However, again due to variations, RFS may be cleared when 10 or 9 entries of data remain.

When the RFS bit is set, a DMA service request is made. An interrupt request is also made unless the receive FIFO interrupt request mask (RIE) bit is cleared. Even though more than eight entries of data may exist within the receive FIFO, the user must configure the DMA burst size to eight words. If programmed I/O is used to service the receive FIFO, a maximum of eight words may be removed without checking if data is valid. After this point, the receive FIFO not empty (RNE) flag must be polled before each read to see if more data remains. After the DMA or CPU empties the FIFO such that nine or more empty locations are available within the receive FIFO, the RFS flag (as well as the DMA and interrupt request) is automatically cleared.

11.10.10.6 Framing Error Status (FRE) (read/write, nonmaskable interrupt)

The framing error status (FRE) bit is set when a frame alignment error is detected by the receive logic. A frame alignment error is detected on received data when a preamble is followed by something other than another preamble or a start flag.

The following table shows the bit locations corresponding to the status and flag bits within HSSP status register 0. Note that the reset state of all writable status bits is unknown (indicated by question marks) and must be cleared (by writing a one to them) before enabling the HSSP. Also note that writes to reserved bits are ignored and reads return zeros.

Address: 0h 8004 0074 **HSSR0** **Read/Write & Read-Only**

Bit	7	6	5	4	3	2	1	0
	Reserved		FRE	RFS	TFS	RAB	TUR	EIF
Reset	0	0	?	0	0	?	?	?

Bit	Name	Description
0	EIF	End/error in FIFO (read-only). 0 – Bits 8–10 are not set within any of the eight bottom entries of the receive FIFO. Receive FIFO DMA service requests are enabled. 1 – One or more tag bits (8 – 10) are set within one or more of the bottom eight entries of the receive FIFO. Request interrupt, disable receive FIFO DMA service requests.
1	TUR	Transmit FIFO underrun. 0 – Transmit FIFO has not experienced an underrun. 1 – Transmit logic attempted to fetch data from transmit FIFO while it was empty; interrupt request signalled if not masked (if TUS=1).
2	RAB	Receiver abort. 0 – No abort has been detected for the incoming frame. 1– Abort detected during receipt of incoming frame. Two or more chips containing no pulses (0000) detected on receive pin. EOF bit set in receive FIFO next to last piece of “good” data received before the abort, interrupt requested.
3	TFS	Transmit FIFO service request (read-only). 0 – Transmit FIFO is more than half-full (nine or more entries filled) or transmitter disabled. 1 – Transmit FIFO is half-full or less (eight or fewer entries filled) and transmitter operation is enabled. DMA service request signalled; interrupt request signalled if not masked (if TIE=1).
4	RFS	Receive FIFO service request (read-only). 0 – Receive FIFO contains 11 or fewer entries of data or receiver disabled. 1 – Receive FIFO is two- to three-fifths full (contains 9, 10, 11, or 12 entries of data) or more, and receiver operation is enabled. DMA service request signalled; interrupt request signalled if not masked (if RIE=1).
5	FRE	Framing error. 0 – No framing errors encountered in the receipt of this data. 1 – Framing error occurred; preamble followed by something other than another preamble or start flag, request interrupt.
7..6	—	Reserved.

11.10.11 HSSP Status Register 1

HSSP status register 1 (HSSR1) contains flags that indicate when the receiver is synchronized, the transmitter is active, the transmit FIFO is not full, the receive FIFO is not empty, and when an end-of-frame, CRC error, or underrun error has occurred. All bits within HSSR1 are read-only and noninterruptible.

11.10.11.1 Receiver Synchronized Flag (RSY) (read-only, noninterruptible)

The receiver synchronized (RSY) flag is a read-only bit that is set when the receiver is synchronized with the incoming data stream, and is cleared when the receive logic is in hunt mode (looking for the preamble to achieve byte and frame synchronization), or the receiver is disabled (RXE=0). This bit does not request an interrupt.

11.10.11.2 Transmitter Busy Flag (TBY) (read-only, noninterruptible)

The transmitter busy (TBY) flag is a read-only bit that is set when the transmitter is actively transmitting a frame (address, control, data, CRC, start or stop flag), and is cleared when the transmitter is idle (transmitting preambles) or the transmitter is disabled (TXE=0). This bit does not request an interrupt.

11.10.11.3 Receive FIFO Not Empty Flag (RNE) (read-only, noninterruptible)

The receive FIFO not empty flag (RNE) is a read-only bit that is set whenever the receive FIFO contains one or more bytes of valid data and is cleared when it no longer contains any valid data. This bit can be polled when using programmed I/O to remove remaining bytes of data from the receive FIFO because DMA service and CPU interrupt requests are made only when 12, 11, 10, or 9 bytes reside within the FIFO. Data will remain after each service request as well as at the end of a frame. This bit does not request an interrupt.

11.10.11.4 Transmit FIFO Not Full Flag (TNF) (read-only, noninterruptible)

The transmit FIFO not full flag (TNF) is a read-only bit that is set whenever the transmit FIFO contains one or more entries that do not contain valid data and is cleared when the FIFO is completely full. This bit can be polled when using programmed I/O to fill the transmit FIFO over its halfway mark. This bit does not request an interrupt.

11.10.11.5 End-of-Frame Flag (EOF) (read-only, noninterruptible)

The end-of-frame flag (EOF) is set when the last byte of data within a frame (including aborted frames) resides within the bottom entry of the receive FIFO.

The receive FIFO contains three tag bits (8, 9, and 10) that are not directly readable. The 8th bit is set at the top of the FIFO whenever the last byte within a frame is moved from the receive serial shifter to the top of the receive FIFO. This tag travels along with the last data value as it moves down the FIFO. Each time a data value is transferred to the bottom of the FIFO (caused by a read of the previous value), the state of the tag bit is moved from the FIFO to the EOF bit in the status register. Whenever EOF is set within the bottom eight entries of the receive FIFO, EIF is set within HSSR0, an interrupt is signalled, and the receive FIFO DMA request is disabled. After the end/error in FIFO (EIF) status bit is set, the user should always read HSSR1 first to check EOF before reading the data value from HSDR because EOF corresponds to the current data byte at the bottom of the receive FIFO and is updated each time data is removed from the FIFO.

11.10.11.6 CRC Error Status (CRE) (read-only, noninterruptible)

The CRC error flag (CRE) is set when the CRC value calculated by the receive logic does not match the CRC value contained within the incoming serial data stream.

The receive FIFO contains three tag bits (8, 9, and 10) that are not directly readable. Whenever a CRC error is detected, the 9th bit is set within the top entry of the receive FIFO corresponding to the last byte of data within the frame. This tag travels along with the last piece of data from the frame as it moves down the FIFO. Each time a data value is transferred to the bottom of the FIFO (caused by a read of the previous value), the state of the tag bit is moved from the FIFO to the CRE bit in the status register, indicating whether or not the frame has encountered a CRC error. Whenever CRE is set within the bottom half of the receive FIFO, EIF is set within HSSR0, an interrupt is signalled, and the receive FIFO DMA request is disabled. After the end/error in FIFO (EIF) status bit is set, the user should always read HSSR1 first to check CRE before reading the data value from HSDR because CRE corresponds to the current data byte at the bottom of the receive FIFO and is updated each time data is removed from the FIFO.

11.10.11.7 Receiver Overrun Status (ROR) (read-only, noninterruptible)

The receiver overrun flag (ROR) is set when the receive logic attempts to place data into the receive FIFO after it has been completely filled.

The receive FIFO contains three tag bits (8, 9, and 10) that are not directly readable. The 10th bit is set within the top entry of the receive FIFO whenever an overrun occurs. This tag travels along with the last “good” data value before the overflow occurred as it moves down the FIFO. Each time a data value is transferred to the bottom of the FIFO (caused by a read of the previous value), the state of the tag bit is moved from the FIFO to the ROR bit in the status register, indicating that the next value in the FIFO is the last “good” piece of data before the overflow occurred. Whenever ROR is set within the bottom eight entries of the receive FIFO, EIF is set within HSSR0, an interrupt is signalled, and the receive FIFO DMA request is disabled. After the end/error in FIFO (EIF) status bit is set, the user should always read HSSR1 first to check ROR before reading the data value from HSDR because ROR corresponds to the current data byte at the bottom of the receive FIFO and is updated each time data is removed from the FIFO.

The following table shows the location of the flags within HSSP status register 1. The bits within this register are read-only and do not produce interrupt requests. Note that writes to bit 7 are ignored and reads return zero.

Address: 0h 8004 0078 **HSSR1** **Read-Only**

Bit	7	6	5	4	3	2	1	0
	Res.	ROR	CRE	EOF	TNF	RNE	TBY	RSY
Reset	0	0	0	0	1	0	0	0

Bit	Name	Description
0	RSY	Receiver synchronized flag (read-only). 0 – Receiver is in hunt more or is disabled. 1 – Receiver logic is synchronized with the incoming data (no interrupt generated).
1	TBY	Transmitter busy flag (read-only). 0 – Transmitter is idle (continuous preambles) or disabled. 1 – Transmit logic is currently transmitting a frame (address, control, data, CRC, or start/stop flag); no interrupt generated.
2	RNE	Receive FIFO not empty (read-only). 0 – Receive FIFO is empty. 1 – Receive FIFO is not empty (no interrupt generated).
3	TNF	Transmit FIFO not full (read-only). 0 – Transmit FIFO is full. 1 – Transmit FIFO is not full (no interrupt generated).
4	EOF	End of frame (read-only). 0 – Current frame has not completed. 1 – The value at the bottom of the receive FIFO is the last byte of data within the frame.
5	CRE	CRC error (read-only). 0 – No CRC check errors encountered in the receipt of data. 1 – CRC calculated on the incoming data. Does not match CRC value contained within the received frame.
6	ROR	Receive FIFO overrun (read-only). 0 – Receive FIFO has not experienced an overrun. 1 – Receive logic attempted to place data into receive FIFO while it was full; the next data value in the FIFO is the last piece of “good” data before the FIFO was overrun.
7	—	Reserved.

11.10.12 UART Register Locations

Table 11-16 shows the registers associated with the UART block and the physical addresses used to access them.

Table 11-16. UART Control, Data, and Status Register Locations

Address	Name	Description
0h 8003 0000	UTCR0	UART control register 0
0h 8003 0004	UTCR1	UART control register 1
0h 8003 0008	UTCR2	UART control register 2
0h 8003 000C	UTCR3	UART control register 3
0h 8003 0010	UTCR4	UART control register 4
0h 8003 0014	UTDR	UART data register
0h 8003 0018	—	Reserved
0h 8003 001C	UTSR0	UART status register 0
0h 8003 0020	UTSR1	UART status register 1
0h 8003 0024 – 0h 8003 005C	—	Reserved

11.10.13 HSSP Register Locations

Table 11-17 shows the registers associated with the HSSP block and the physical addresses used to access them.

Table 11-17. HSSP Control, Data, and Status Register Locations

Address	Name	Description
0h 8004 0060	HSCR0	HSSP control register 0
0h 8004 0064	HSCR1	HSSP control register 1
0h 8004 0068	—	Reserved
0h 8004 006C	HSDR	HSSP data register
0h 8004 0070	—	Reserved
0h 8004 0074	HSSR0	HSSP status register 0
0h 8004 0078	HSSR1	HSSP status register 1
0h 8004 007C - 0h 8004 FFFF	—	Reserved

Note: HSCR2 resides within the same address space as the PPC.

0h 9006 0028	HSCR2	HSSP Control register 2

11.11 Serial Port 3 - UART

Serial port 3 is a general-purpose, full-duplex, universal asynchronous receiver/transmitter (UART) that supports much of the functionality of the 16550 protocol. It can operate at baud rates from 56.24 bps to 230.4 Kbps. It supports 7 or 8 bits of data (odd, even, or no parity), one start bit, either one or two stop bits, and can transmit a continuous break signal. An external clock can also be input using GPIO pin 20 to synchronously sample and drive data on either edge of the clock as programmed by the user. The external pins dedicated to this interface are TXD3 and RXD3. If use of the UART is not required, these pins can be used by the peripheral pin controller (PPC) to perform general- purpose input/output (noninterruptible).

An 8-entry x 8-bit FIFO is used to buffer outgoing data, and a 12-entry x 11-bit FIFO is used to buffer incoming data (3 bits per entry are used to store framing, parity, and receive FIFO overrun error flags for each character received). The FIFOs are filled or emptied using the DMA or the CPU. An interrupt is generated when a framing, parity, or receiver overrun error is present within the bottom four entries of the receive FIFO, when the transmit FIFO is half-empty or the receive FIFO is one- to two-thirds full, when a begin and end of break is detected on the receiver, and when the receive FIFO is partially full and the receiver is idle for three or more frame periods.

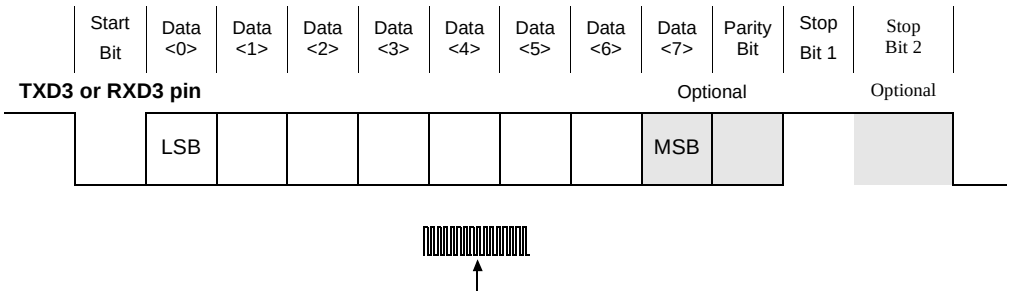
Modem control signals (RTS, CTS, DTR, and DSR) are not implemented in this block, but can be implemented using the general-purpose I/O port (GPIO) pins. See Chapter 9, “System Control Module”.

11.11.1 UART Operation

Following hardware reset, the UART is disabled, which causes the peripheral pin controller (PPC) to assume control of the UART’s pins. Reset causes the PPC to configure all of the peripheral pins as inputs, including the UART’s transmit (TXD3) and receive (RXD3) pins. Reset also causes the UART’s transmit and receive FIFOs to be flushed (all entries invalidated). Before enabling the UART, the user must first clear any writable or “sticky” status bits that are set by writing a one to each bit. Next, the desired mode of operation is programmed in the control registers. At this point, the user may “prime” the transmit FIFO by writing up to eight values, or the FIFO can remain empty and the transmit FIFO DMA or interrupt request may be used to trigger its service when the transmitter is enabled. When the UART is enabled, transmission and reception of data can begin on the transmit (TXD3) and receive (RXD3) pins.

Figure 11-29 shows the format of a single UART data frame.

Figure 11-29. Example UART Data Frame

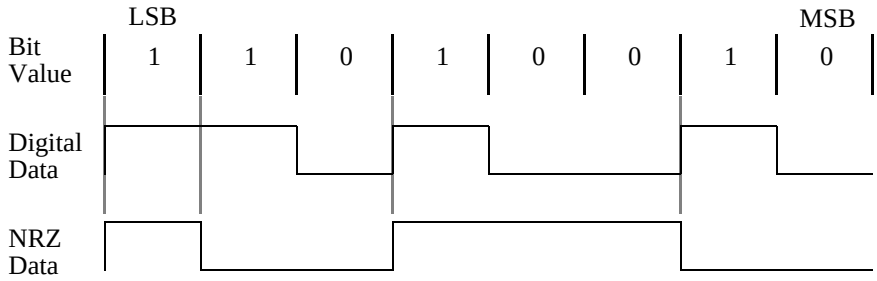


Receive data sample counter frequency = 16x bit frequency, each bit sampled on eighth clock.

11.11.1.1 Frame Format

NRZ encoding is used by the UART to represent individual bit values. A one is represented by a line transition and a zero is represented by no line transition. Figure 11-30 shows the NRZ encoding of the data byte 8b 0100 1011. Note that the byte's LSB is transmitted first.

Figure 11-30. NRZ Bit Encoding Example – (0100 1011)



Each data frame is between 9 bits and 12 bits long depending on the size of data programmed, if parity is enabled and if a second stop bit is enabled. The frame begins with a start bit that is represented by a high to low transition. Next, either 7 bits or 8 bits of data are transmitted, beginning with the least significant bit. An optional parity bit follows, which is set if even parity is enabled and an odd number of ones exist within the data byte, or if odd parity is enabled and the data byte contains an even number of ones. The data frame ends with either one or two stop bits as programmed by the user, which is represented by one or two successive bit periods of a logic one. Note that the receiver only tests for one stop bit per frame.

11.11.1.2 Baud Rate Generation

The baud or bit rate is derived by dividing down the 3.6864-MHz clock generated by the on-chip PLL. The clock is first divided by a programmable number between 1 and 4097, and then by a fixed value of 16. The receive baud clock is synchronized with the data stream using a digital PLL each time the start bit is detected on the receive data line. Receive data is then sampled halfway through each bit period by counting 8 of the 16 clocks, which are produced before the fixed divide by 16 takes place. (See Figure 11-29.)

11.11.1.3 Receive Operation

The UART receives incoming data by using a serial shifter. It latches the frame, strips it of its start, parity, and stop bits, and then places the data within receive FIFO. If parity is enabled, the number of data bits, which is one, is counted as data and is extracted from each frame. Parity is then checked by comparing this value to the stripped parity bit. Either odd or even parity is checked as specified by the programmer. If a parity error is detected, the parity error bit is set in the FIFO entry corresponding to the data value that caused the error. Additionally, if a logic zero is detected by the receive logic where a stop bit was expected, the framing error bit is set in the FIFO entry corresponding to the errant data. When the FIFO fills between one- to two-thirds full, an interrupt or DMA request is signalled. If the FIFO is completely filled and the receive logic attempts to place additional data within the FIFO, the overrun bit is set next to the last byte of data received within the FIFO. Any data received while the FIFO is completely full is discarded.

The parity, framing, and overrun error bits are transferred down the receive FIFO along with the data that caused the error. Whenever any of the four bottom FIFO entries contain one or more error bits that are set, an interrupt is generated and receive FIFO DMA requests are disabled until the error is flushed from the FIFO and the status bit that signalled the interrupt is cleared. At this point, the user should use programmed I/O to check the error bits and remove data one piece at a time until the four FIFO entries are flushed. Each time a data value is transferred to the bottom of the FIFO, the state of the parity, framing, and overrun bits within the last FIFO entry are automatically transferred to their respective flag bits in the status register. When any of these three flags are set in the UART status register, it indicates that the next data value available within the FIFO contains an error. The user must first check the state of these three flags to see if the next value within the FIFO contains an error, then read the FIFO value. After four values have been removed from the FIFO and the errors are identified, the DMA is automatically reenabled once the error in FIFO bits are removed from the FIFO.

If the receive FIFO contains valid data and three frame periods elapse without the reception of data on RXD3, the receiver idle interrupt is generated. Also, if the receive logic detects a null character (all zeros, including the parity bit) followed by a framing error (stop bit is zero as well), the receive logic generates a beginning of break detect, which interrupts the CPU. Because breaks can be signalled for long periods of time, after the break is negated and the receive pin transitions high, the receive logic generates an end of break detect, which again interrupts the CPU.

11.11.1.4 Transmit Operation

The UART transmit logic operates at the same time as the receive logic (full-duplex). Data is taken from the transmit FIFO; start, parity, and stop bits are added to generate a frame; and the value is loaded into a serial shift register. The contents are shifted out onto the TXD3 pin, clocked by the programmed baud clock. When the transmit FIFO is emptied more than halfway, an interrupt or DMA request is signalled. If the transmit FIFO is completely emptied, the transmit line remains high (one) after the last data value is transmitted to indicate the transmitter is idle. The TXD3 pin remains high until additional data is written to the transmit FIFO.

11.11.1.5 Transmit and Receive FIFOs

To reduce chip size and power consumption, the UART's FIFOs use self-timed logic (they are not clocked). Because of process and environmental variations, the depth at which a service request is triggered to empty the receive FIFO is variable. This variation spans a maximum of four FIFO entries; the receive FIFO service request can be made at four different FIFO depths.To compensate for this variability and guarantee that at least four valid entries of data exist within the FIFO before generating a service request, an extra four entries have been added to the receive FIFO (four entries more than the transmit FIFO). The transmit FIFO is 8 entries deep and the receive FIFO is 12 entries deep. The point at which the receive FIFO service request is triggered spans the middle third of the 12-entry FIFO. The service request is signalled at a depth from one-third full to two-thirds full (when the FIFO contains five, six, seven, or eight entries of data).

This service request variation applies only to an empty FIFO that is filled (receive FIFO). It does not apply to a full FIFO that is emptied (transmit FIFO). The transmit FIFO is guaranteed to signal a service request when it has four or more empty entries and negate the request when the FIFO contains five or more entries that are filled.

If the DMA is used to service either one or both of the UART's FIFOs, the burst size must be set to 4 words even though more than four entries of data may exist within the receive FIFO. If programmed I/O is used to service the FIFOs, a maximum of 4 words may be added to the transmit FIFO without checking if more space is available. Likewise, a maximum of 4 words may be

removed from the receive FIFO without checking if more data is available. After this point, the user must poll a set of status bits that indicates if any data remains in the receive FIFO or if space is available in the transmit FIFO before emptying or filling the FIFOs any further.

11.11.1.6 CPU and DMA Register Access Sizes

Bit positioning, byte ordering, and addressing of the UART is described in terms of little endian ordering. All UART registers are 8 bits wide and are located in the least significant byte of individual words. The ARM peripheral bus does not support byte or half-word operations. All reads and writes of the UART by the CPU should be wordwide. Two separate dedicated DMA requests exist for both the transmit and the receive FIFO. If the DMA controller is used to service the transmit and/or receive FIFOs, the user must ensure the DMA is properly configured to perform bytewide accesses, using 4 bytes per burst.

11.11.2 UART Register Definitions

There are seven bytewide registers within the UART: four control registers, one data register, and two status registers. The control registers are used to program the baud rate, data length, number of stop bits, and odd or even parity. They are used to receive and transmit sample clock edge type, and to transmit a break. Also, they are used to enable or disable transmit and receive operation, parity, use of the sample clock input, and loopback mode. The data register is 8 bits and addresses the top location of the transmit FIFO and bottom location of the receive FIFO. When it is read, the receive FIFO is accessed, and when it is written, the transmit FIFO is accessed. The status registers contain bits that signal the transmit FIFO service request, receive FIFO service request, receiver idle, the begin and end of break detect, and error in FIFO conditions. Each of these status conditions signal an interrupt request to the interrupt controller. The status registers also flag when the UART is actively transmitting characters, when the transmit FIFO is not full, when the receive FIFO is not empty, and when a parity, framing, or overrun error was detected for the data value currently located in the bottom entry of the receive FIFO (no interrupt generated).

11.11.3 UART Control Register 0

UART control register 0 (UTCR0) contains seven different bit fields that control various functions within the UART.

11.11.3.1 Parity Enable (PE)

The parity enable (PE) bit is used to enable or disable parity checking by the receive data logic as well as parity generation by the transmit logic. When parity is enabled (PE=1), the odd/even parity select (OES) control bit is decoded to determine which type of parity should be checked and generated. The parity of each data frame received is checked. If the parity type programmed in the OES bit does not match the parity of the data received, the parity error (PRE) bit is set in the same entry in the receive FIFO where the errant data resides. When parity is disabled (PE=0), the parity check and generation logic is disabled, parity bits are not inserted into transmitted frames, and the receive logic expects a stop bit to occur after the MSB of each data value is received.

11.11.3.2 Odd/Even Parity Select (OES)

The odd/even parity select (OES) bit is used to select whether odd or even parity should be used by the transmit and receive logic. When OES=0, odd parity is selected; when OES=1, even parity is selected. When parity is enabled (PE=1), the parity bit is placed after the data's MSB in each frame.

The transmit logic sets or clears the parity bit to make the total number of ones transmitted (including the parity bit) match the parity type programmed using OES (if even parity is selected (OES=1) and there is an odd number of ones in the data to be transmitted, the parity bit is set). The receive data logic counts the number of ones encountered in the incoming data stream (including the parity bit), then strips the parity bit from the data. If the parity type of the frame does not match the parity selected by OES, the parity error bit is set (bit 8) within the FIFO entry corresponding to the data that produced the parity error.

11.11.3.3 Stop Bit Select (SBS)

The stop bit select (SBS) bit selects whether one or two stop bits should be used in transmission. When SBS=0, one stop bit is inserted in the transmit frame for each character. When SBS=1, two stop bits are inserted. SBS does not affect the UART's receive logic. The receiver always checks to make sure there is at least one stop bit per character.

11.11.3.4 Data Size Select (DSS)

The data size select (DSS) bit is programmed to select the size of the data transmitted and received within each frame. Data can be 7 or 8 bits in length. When 7-bit data is programmed, the data is right justified within the FIFOs. The unused bit is zero filled within the receive FIFO, and is ignored within the transmit FIFO. Note that the user must right justify data supplied to the transmit FIFO when 7-bit data is selected.

11.11.3.5 Sample Clock Enable (SCE)

The sample clock enable (SCE) bit is used to enable or disable the use of a clock input from a GPIO pin to synchronously sample and drive data to and from the UART. When SCE=0, the on-chip 3.6864-MHz PLL, the UART's programmable baud rate generator, and the receive logic's digital PLL are used. When SCE=1, a clock is input from a GPIO pin and is used to synchronously drive both the transmit and receive logic. Note that the user must configure the GPIO pin as an input by clearing the corresponding bit in the GPIO pin direction register (GPDR) and switch control of the GPIO pin to the UART by setting the corresponding bit in the GPIO alternate function register (GAFR). See Chapter 9, "System Control Module".

For the receive logic, the RCE bit is decoded to select which edge of the input clock is used to latch each bit of the incoming frame. Note that the clock is not embedded within the data stream and the digital PLL is shut down to conserve power. For the transmit logic, the TCE bit is decoded to select which edge of the input clock is used to drive each bit of the outgoing frame. Note that the clock driving the programmable baud rate generator is shut down when SCE=1 to conserve power. Also note that SCE does not affect the frame format of data being transmitted and received by the UART.

The SA-1100 has a total of three UARTs (serial ports 1, 2 and 3). When the external sample clock function is enabled, serial port 1 uses the GPIO<18> pin and serial port 3 uses GPIO<19>. Serial port 2 does not support the sample clock function.

11.11.3.6 Receive Clock Edge Select (RCE)

When SCE=1, the receive clock edge select (RCE) bit is used to select which edge of the clock input from the GPIO pin to use (rising or falling) to synchronously sample data from the receive pin. When RCE=0, each bit received is sampled on the rising edge of the sample input clock; when RCE=1, bits are sampled on the clock's falling edge. Note that the internal baud rate generator and receive logic's digital PLL are not used in this mode. RCE is ignored when SCE=0.

11.11.3.7 Transmit Clock Edge Select (TCE)

When SCE=1, the transmit clock edge select (TCE) bit is used to select which edge of the clock input from the GPIO pin to use (rising or falling) to synchronously drive data onto the transmit pin. When TCE=0, each bit transmitted is driven on the rising edge of the sample input clock; when TCE=1, bits are driven on the clock's falling edge. Note that the internal baud rate generator is not used in this mode. TCE is ignored when SCE=0.

The following table shows the bit locations corresponding to the seven different control bit fields within UART control register 0. The UART must be disabled (RXE=TXE=0) when changing the state of any bit within this register. The reset state of these control bits is unknown (indicated by question marks) and must be initialized before enabling the UART. Note that writes to bit 7 are ignored and reads return zero.

Address: 0h 8005 0000 **UTCR0** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	Res	TCE	RCE	SCE	DSS	SBS	OES	PE
Reset	0	?	?	?	?	?	?	?

Bit	Name	Description
0	PE	Parity enable. 0 – Parity checking on received data and parity generation on transmitted data is disabled. 1 – Parity checking on received data and parity generation on transmitted data is enabled.
1	OES	Odd/even parity select. 0 – Odd parity checking/generation selected. Parity error bit set if even number of ones counted in data field (including the parity bit). 1 – Even parity checking/generation selected. Parity error bit set if odd number of ones counted in data field (including the parity bit).
2	SBS	Stop bit select. 0 – One stop bit transmitted per frame. 1 – Two stop bits transmitted per frame. **Note:** Receiver not affected by SBS; always checks for one stop bit.
3	DSS	Data size select. 0 – 7-bit data. 1 – 8-bit data. **Note:** For 7-bit mode, the data is right justified within the FIFO entries, the MSBs in the receive FIFO are zero filled, and the MSBs in the transmit FIFO are ignored.
4	SCE	Sample clock enable. 0 – on-chip baud rate generator and digital PLL used to transmit and receive asynchronous data. 1 – A clock is input via GPIO pin 20 and is used synchronously to sample receive data and drive transmit data. **Note:** Serial port 1's UART uses GPIO pin 18 for the sample clock input; serial port 2 does not support the sample clock function. The user must also program the appropriate bits in the GPDR and GAFR registers within the system control module.
5	RCE	Receive clock edge select. 0 – Rising edge of clock input on GPIO pin 20 used to latch data from the receive pin if SCE=1. 1 – Falling edge of clock input on GPIO pin 20 used to latch data from the receive pin if SCE=1.
6	TCE	Transmit clock edge select. 0 – Rising edge of clock input on GPIO pin 20 used to drive data onto the transmit pin if SCE=1. 1 – Falling edge of clock input on GPIO pin 20 used to drive data onto the transmit pin if SCE=1.
7	—	Reserved.

11.11.4 UART Control Registers 1 and 2

UART control register 1 (UTCR1) contains the upper 4 bits and UTCR2 the lower 8 bits of the baud rate divisor field.

11.11.4.1 Baud Rate Divisor (BRD)

The 12-bit baud rate divisor (BRD) field is used to select the baud or bit rate of the UART. A total of 4096 different baud rates can be selected, ranging from a minimum of 56.24 bps to a maximum of 230.4 Kb/ps. The baud rate generator uses the 3.6864-MHz clock generated by the on-chip PLL divided by 16 to generate the bit clock. A digital PLL is used to synchronize the baud rate of the receiver each time the start bit is detected on the receive pin and each bit of the receive data stream is sampled on the eighth clock of the divide by 16 counter (halfway through the bit period). The resultant baud rate, given a specific BRD value or required BRD value and given a desired baud rate, can be calculated using the following two respective equations, where BRD is the decimal equivalent of the binary value programmed within the bit field:

$$BaudRate = \frac{3.6864 \times 10^6}{16x(BRD+1)}$$

$$BRD = \frac{3.6864 \times 10^6}{16xBaudRate} - 1$$

The following tables show the bit locations corresponding to the baud rate divisor field that is split between two 8-bit registers. The upper four bits of BRD reside within UTCR1 and the lower eight bits reside within UTCR2. The UART must be disabled (RXE=TXE=0) whenever these registers are written. The reset state of the BRD field is unknown (indicated by question marks) and must be initialized before enabling the UART. Note that writes to reserved bits are ignored and reads return zeros.

Address: 0h 8005 0004 **UTCR1** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	Reserved				BRD<11:8>			
Reset	0	0	0	0	?	?	?	?

Bit	Name	Description
3..0	BRD<11:8>	Baud rate divisor. Encoded value (from 0 to 4096) used to generate the baud rate of the UART. Baud Rate = $3.6864 \times 10^6/(16 \times (BRD+1))$, where BRD is a decimal value.
7..4	—	Reserved.

Address: 0h 8005 0008 **UTCR2** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	BRD<7:0>							
Reset	?	?	?	?	?	?	?	?

Bit	Name	Description
7..0	BRD<7:0>	Baud rate divisor. Encoded value (from 0 to 4096) used to generate the baud rate of the UART. Baud Rate = $3.6864 \times 10^6/(16 \times (BRD+1))$, where BRD is a decimal value.

11.11.5 UART Control Register 3

UART control register 3 (UTCR3) contains six different bit fields that control various functions within the UART.

11.11.5.1 Receiver Enable (RXE)

The receiver enable (RXE) bit is used to enable and disable all UART receive operations. When RXE=1, the UART receive logic is enabled; when RXE=0, it is disabled. When the receiver is disabled, control of the RXD3 pin is given to the peripheral pin controller (PPC) so that it may be used for general-purpose input and output (noninterruptible). See the Section 11.13, “Peripheral Pin Controller (PPC)” on page 11-184 for a description of the PPC.

It is required that the user first program all other control bits before setting RXE (even the transmit bits). If the RXE bit is cleared to zero while the UART is actively receiving data, reception is stopped immediately and the remaining bits within the receive serial shifter are reset. In addition, all entries within the receive FIFO are reset (all other control/status/flag bits remain intact).

11.11.5.2 Transmitter Enable (TXE)

The transmitter enable (TXE) bit is used to enable and disable all UART transmit operations. When TXE=1, UART transmit logic is enabled; when TXE=0, it is disabled. When the transmitter is disabled, control of the TXD3 pin is given to the peripheral pin controller (PPC) for general-purpose input and output use (noninterruptible). See the Section 11.13, “Peripheral Pin Controller (PPC)” on page 11-184 for a description of the PPC.

It is required that the user first program all other control bits before setting TXE (even the receive bits). If the TXE bit is cleared to zero while the UART is actively transmitting data, transmission is stopped immediately and the remaining bits within the transmit serial shifter are reset. In addition, all entries within the transmit FIFO are reset (all other control/status/flag bits remain intact).

11.11.5.3 Break (BRK)

The break (BRK) control bit is used to continuously transmit a break by forcing the transmit pin (TXD3) low. When the BRK bit is set, the transmit pin is forced low immediately. If the transmitter is actively transmitting data, the remaining bits in the serial shifter continue to be shifted out, but the bits are ignored (not placed on the transmit pin). Asserting BRK also prevents the transmit logic from fetching any additional data from the transmit FIFO once the shifter is empty. The transmit pin remains low until the BRK bit is cleared, or alternatively, if the transmitter is disabled (TXE=0, or a reset occurs). Once BRK is negated, transmission starts again. The user must ensure that the BRK bit is asserted long enough to cause the off-chip receiver to detect the break condition. The user should also check the transmitter busy (TBY) flag in the status register to ensure that no bits remain in the transmitter’s serial shifter before negating BRK. TBY is asserted as long as the transmitter is actively clocking data through the serial shifter. Once the TBY bit becomes zero, the BRK bit can be negated, and data is once again fetched from the transmit FIFO. Break does not affect the receive portion of the FIFO; normal operation on the receive line continues during the signalling of a break.

11.11.5.4 Receive FIFO Interrupt Enable (RIE)

The receive FIFO interrupt enable (RIE) bit is used to mask or enable both the receive FIFO service request interrupt and receiver idle interrupt. When RIE=0, the interrupts are masked and the receive FIFO service request (RFS) and receiver idle status (RID) bits are ignored by the interrupt controller. When RIE=1, the interrupts are enabled and whenever RFS or RID is set (one), an interrupt request is made to the interrupt controller. Note that programming RIE=0 does not affect the current state of RFS or RID nor the receive logic’s ability to set and clear these bits; it only blocks the generation of the interrupt request. Also note that RIE does not affect generation of the receive FIFO DMA request that is asserted whenever RFS=1.

11.11.5.5 Transmit FIFO Interrupt Enable (TIE)

The transmit FIFO interrupt enable (TIE) bit is used to mask or enable the transmit FIFO service request interrupt. When TIE=0, the interrupt is masked and the state of the transmit FIFO service request (TFS) bit is ignored by the interrupt controller. When TIE=1, the interrupt is enabled, and whenever TFS is set (one), an interrupt request is made to the interrupt controller. Note that programming TIE=0 does not affect the current state of TFS nor the transmit FIFO logic's ability to set and clear TFS; it only blocks the generation of the interrupt request. Also note that TIE does not affect generation of the transmit FIFO DMA request that is asserted whenever TFS=1.

11.11.5.6 Loopback Mode (LBM)

The loopback mode (LBM) bit is used to enable and disable the ability of the UART transmit and receive logic to communicate. When LBM=0, the UART operates normally. The transmit and receive data paths are independent and communicate via their respective pins. When LBM=1, the output of the transmit serial shifter is directly connected to the input of the receive serial shifter internally, and control of the TXD3 and RXD3 pins is given to the peripheral pin control (PPC) unit.

The following table shows the bit location of the bits within UART control register 3. RXE and TXE are the only control bits that are reset to a known state to ensure the UART is disabled following a reset of the SA-1100. The reset state of all other control bits is unknown (indicated by question marks) and must be initialized before enabling the UART. Note that UTCR3 is the only control register that may be written while the UART is enabled. Also note that writes to reserved bits are ignored and reads return zeros.

Address: 0h 8005 000C **UTCR3** **Read/Write**

Bit	7	6	5	4	3	2	1	0
	Reserved		LBM	TIE	RIE	BRK	TXE	RXE
Reset	0	0	?	?	?	?	0	0

Bit	Name	Description
0	RXE	Receiver enable. 0 – UART receive operation disabled; PPC is given control of **RXD3**. 1 – UART receive operation enabled.
1	TXE	Transmitter enable. 0 – UART transmit operation disabled; PPC is given control of **TXD3**. 1 – UART transmit operation enabled.
2	BRK	Break. 0 – UART in normal operation. 1 – Force TXD3 low (all bits in the frame are a zero) to generate a break.
3	RIE	Receive FIFO interrupt enable. 0 – Receive FIFO one- to two-thirds full (or more) and receiver idle conditions do not generate an interrupt (RFS and RID bit ignored). 1 – Receive FIFO one- to two-thirds full (or more) and receiver idle conditions generate an interrupt (state of RFS and RID sent to interrupt controller).
4	TIE	Transmit FIFO interrupt enable. 0 –Transmit FIFO half-full or less condition does not generate an interrupt (TFS bit ignored). 1 – Transmit FIFO half-full or less condition generates an interrupt (state of TFS sent to interrupt controller).
5	LBM	Loopback mode. 0 – Normal serial port operation enabled. 1 – Output of transmit serial shifter is connected to input of receive serial shifter internally and control of TXD3 and RXD3 pins is given to the PPC unit.
7.. 6	—	Reserved.

11.11.6 UART Data Register

The UART data register (UTDR) is an 8-bit register corresponding to both the top and bottom entries of the transmit and receive FIFOs, respectively.

When UTDR is read, the lower 8 bits of the bottom entry of the 10-bit receive FIFO are accessed. As data enters the top of the receive FIFO, bits 8..10 are used to indicate various error conditions that occur during reception of each piece of data. The error bits are transferred down the FIFO along with the value that caused the error. When data reaches the bottom, bit 8 of the bottom FIFO entry is automatically transferred to the parity error (PRE) flag, bit 9 to the framing error (FRE) flag, and bit 10 to the receiver overrun (ROR) flag, all within the UART status register. The user can read these flags to determine if the value at the bottom of the FIFO encountered an error during reception. After checking the flags, the FIFO value can then be read, which causes the data in the next location of the receive FIFO to automatically be transferred down to the bottom entry and its error bits to be transferred to the status register. The error in FIFO (EIF) flag bit is set whenever one or more of the error bits (8..10) is set within any of the bottom four entries of the receive FIFO and is cleared when no error bits are set in the bottom four entries of the FIFO. When EIF is set, an interrupt is generated and receive FIFO DMA requests are disabled so that the user can manually empty the FIFO, always checking the parity, framing, and overrun flags in the status register first before removing the data values from the FIFO. After each entry is removed, the user should check the EIF bit to see if any errors remain, and repeat the procedure until all errors are flushed from the FIFO. Once EIF is cleared, servicing of the receive FIFO by the DMA controller is automatically reenabled.

When UTDR is written, the topmost entry of the 8-bit transmit FIFO is accessed. After a write, data is automatically transferred down to the lowest location within the transmit FIFO that does not already contain valid data. Data is removed from the bottom of the FIFO one piece at a time by the transmit logic and is loaded into the transmit serial shifter along with start and stop bits (and the optional parity and second stop bits), then is serially shifted out onto the TXD3 pin at the programmed baud rate.

The following table shows the bit locations corresponding to the data field, parity, framing, and receiver overrun error bits within the UART data register. Note that both FIFOs are cleared when the SA-1100 is reset, the transmit FIFO is cleared when writing TXE=0, and the receive FIFO is cleared when writing RXE=0.

Address: 0h 8005 0014 **UTDR** **Read/Write**

Bit	10	9	8	7	6	5	4	3	2	1	0
	ROR	FRE	PRE	Bottom of receive FIFO data							
Reset	0	0	0	0	0	0	0	0	0	0	0

Read Access

(**Note:** ROR, FRE, PRE are not read, but rather are transferred to corresponding status bits in UTSR1 each time a data value is transferred to UTDR.)

Bit	7	6	5	4	3	2	1	0
	Top of transmit FIFO data							
Reset	0	0	0	0	0	0	0	0

Write Access

Bit	Name	Description
7..0	DATA	Top/bottom of transmit/receive FIFO data. Read – Bottom of receive FIFO data. Write – Top of transmit FIFO data.
8	PRE	Parity error. 0 – No parity errors encountered in the receipt of this data (or parity disabled). 1 – Parity error encountered in the receipt of this data. **Note:** Each time an 11-bit value reaches the bottom of the receive FIFO, bit 8 from the last FIFO entry is transferred to the PRE bit in UTSR1.
9	FRE	Framing error. 0 – Stop bit for this frame was a one. 1 – Stop bit for this frame was a zero. **Note:** Each time an 11-bit value reaches the bottom of the receive FIFO, bit 9 from the last FIFO entry is transferred to the FRE bit in UTSR1.
10	ROR	Receiver overrun. 0 – No receiver overrun has been detected. 1 – Receive logic attempted to place data into receive FIFO while it was full; one or more data values following this entry were lost. **Note:** Each time an 11-bit value reaches the bottom of the receive FIFO, bit 10 from the last FIFO entry is transferred to the ROR bit in UTSR1.

11.11.7 UART Status Register 0

UART status register 0 (UTSR0) contains bits that signal the transmit FIFO interrupt request, receive FIFO interrupt request, receiver idle detect, the begin and end of receiver break detect conditions, and the error in receive FIFO condition. Each of these hardware-detected events signals an interrupt request to the interrupt controller.

Interruptible status bits signal an interrupt requested as long as the bit is set. Once the bit is cleared, the interrupt is cleared. Read/write bits are called status bits, read-only bits are called flags. Status bits are referred to as “sticky” (once set by hardware, must be cleared by software). Writing a one to a sticky status bit clears it; writing a zero has no effect. Read-only flags are set and cleared by hardware; writes have no effect. Additionally, some bits that cause interrupts have corresponding enable/mask bits in the control registers and are indicated in the following section headings. Note that the user has the ability to mask all UART interrupts by clearing bit 17 within the interrupt controller mask register (ICMR). See the Section 9.2, “Interrupt Controller” on page 9-11.

11.11.7.1 Transmit FIFO Service Request Flag (TFS) (read-only, maskable interrupt)

The transmit FIFO service request flag (TFS) is a read-only bit that is set when the transmit FIFO is nearly empty and requires service to prevent an underrun. TFS is set any time the transmit FIFO has four or fewer entries of valid data (half-full or less), and is cleared when it has five or more (more than half-full) entries of valid data. When the TFS bit is set, a DMA service request is made. An interrupt request is also made unless the transmit FIFO interrupt request mask (TIE) bit is cleared. After the DMA or CPU fills the FIFO such that five or more locations are filled within the transmit FIFO, the TFS flag (as well as the DMA and interrupt request) is automatically cleared.

11.11.7.2 Receive FIFO Service Request Flag (RFS) (read-only, maskable interrupt)

The receive FIFO service request flag (RFS) is a read-only bit that is set when the receive FIFO is nearly filled and requires service to prevent an overrun. The amount of data that causes RFS to be set is nondeterministic. However, the range in which RFS will be set is guaranteed. RFS is set at some point when the receive FIFO is one- to two-thirds full (or more). The UART’s FIFOs are self-timed to reduce cost and save power. As a result, the depth at which the receive FIFO service request is generated is variable. This is the reason the receive FIFO is 12 entries deep instead of eight like the transmit FIFO. At which entry in the FIFO the request is actually triggered is dependent on IC process, operating temperature, and so on. The receive FIFO is designed to signal the RFS bit to be set when it contains eight entries of valid data. However, because of the variability of the self-timed logic, RFS may also be set when seven, six, or five entries of valid data are present within the FIFO. Likewise, under normal circumstances, RFS is cleared when the receive FIFO has seven remaining entries of valid data. However, again due to variations, RFS may be cleared when six, five, or four entries of data remain.

When the RFS bit is set, a DMA service request is made. An interrupt request is also made unless the receive FIFO interrupt request enable (RIE) bit is cleared. Even though more than four entries of data may exist within the receive FIFO, the user must configure the DMA burst size to 4 words. If programmed I/O is used to service the receive FIFO, a maximum of 4 words may be removed without checking if data is valid. After this point, the receive FIFO not empty (RNE) flag must be polled before each read to see if more data remains. After the DMA or CPU empties the FIFO such that five or more empty locations are available within the receive FIFO, the RFS flag (as well as the DMA and interrupt request) is automatically cleared.

11.11.7.3 Receiver Idle Status (RID) (read/write, maskable interrupt)

The receiver idle status bit (RID) is set when the receiver is enabled (RXE=1), the receive FIFO is not empty (contains at least one entry of data), and three frame periods elapse without any data having being received. When RID is set, an interrupt request is made unless the receive FIFO interrupt request mask (RIE) bit is cleared.

11.11.7.4 Receiver Begin of Break Status (RBB) (read/write, nonmaskable interrupt)

The receiver begin of break status bit (RBB) is set when the receive logic detects a null character (contains all zeros, including the parity bit), followed by a framing error, which indicates the start bit is zero. In other words, a begin of break is detected when the receive line is held low for one frame duration (whatever size the frame is programmed to). When RBB is set, an interrupt is signalled, a single null frame is placed in the receive FIFO, the framing error bit is set, and all subsequent null frames with framing errors are ignored (not placed within the FIFO). After RBB is cleared by the user, it cannot be set again until the receiver end of break status (REB) bit is set. This interlock is used to prevent added null characters from entering the receive FIFO, and also allows the user to clear the RBB bit (clearing the interrupt) and wait for the receiver end of break interrupt (described in the next section). This interlock is cleared when REB is set, when RXE is cleared, or when the SA-1100 is reset.

11.11.7.5 Receiver End of Break Status (REB) (read/write, nonmaskable interrupt)

The receiver end of break status bit (REB) is set when the receive pin transitions high (rising edge) and the RBB interlock is currently set (described in the preceding section). In other words, an end of break is detected after a begin of break is detected and the receive line transitions from low to high (indicating a new frame is about to occur or the receiver is entering the idle state). When REB is set, an interrupt is signalled, and the RBB interlock is cleared, allowing any future data frame to be stored to the receive FIFO. After the bit is cleared, it cannot be set again until the receiver begin of break status (RBB) bit is once again set.

11.11.7.6 Error in FIFO Flag (EIF) (read-only, nonmaskable interrupt)

The error in FIFO flag (EIF) is a read-only bit that is set when any error bits (8 through 10) are set within the bottom four entries of the receive FIFO and is cleared when no error bits are set within the bottom four entries of the FIFO. When EIF is set, an interrupt is signalled and DMA requests to empty the receive FIFO are disabled until EIF is cleared. To discover the source of the errors, the user should check the state of the FRE, PRE, and ROR bits in UTSR1, then read the corresponding value from UTDR. This procedure should be repeated until EIF is cleared because errors that are present within *any* of the four lowest entries in the receive FIFO will set EIF. Once all error tags are cleared from the bottom half of the receive FIFO, EIF is automatically cleared, which in turn, clears the interrupt and reenables the receive FIFO DMA request.

The following table shows the bit locations corresponding to the status bits within UART status register 0. Note that the reset state of all writable status bits is unknown (indicated by question marks) and must be cleared (by writing a one to them) before enabling the UART. Also note that writes to reserved bits are ignored and reads return zeros.

Address: 0h 8005 001C **UTSR0** **Read/Write & Read-Only**

Bit	7	6	5	4	3	2	1	0
	Reserved		EIF	REB	RBB	RID	RFS	TFS
Reset	0	0	0	?	?	?	0	0

Bit	Name	Description
0	TFS	Transmit FIFO service request (read-only). 0 – Transmit FIFO is more than half-full (five or more entries filled) or transmitter disabled. 1 – Transmit FIFO is half-full (four or fewer entries filled) and transmitter operation is enabled, DMA service request signalled, and interrupt request signalled if not masked (if TIE=1).
1	RFS	Receive FIFO service request (read-only). 0 – Receive FIFO contains seven or fewer entries of data or receiver disabled. 1 – Receive FIFO is one- to two-thirds full (contains 5, 6, 7, or 8 entries of data) or more, and receiver operation is enabled, DMA service request signalled, and interrupt request signalled if not masked (if RIE=1).
2	RID	Receiver idle. 0 – Receiver is busy, receive FIFO is empty, or receiver is disabled. 1 – Receiver is enabled, receive FIFO not empty, 3 frame times elapsed without receiving data, request interrupt.
3	RBB	Receiver begin of break. 0 – No break detected. 1 – Null character followed by parity and stop bits containing zeroes received, request interrupt. **Note:** Setting this bit allows the setting of REB, and also prevents further null characters with framing errors from being stored in the receive FIFO (only one stored).
4	REB	Receiver end of break. 0 – No end of break detected. 1 – Beginning of break was detected (interlock set) and a rising edge detected on the receive pin, request interrupt. **Note:** Setting of this bit allows the setting of RBB, and also allows characters to once again be stored in the receive FIFO.
5	EIF	Error in FIFO (read-only). 0 – Bits 8..10 are not set within any of the four bottom entries of the receive FIFO, receive FIFO DMA service requests are enabled. 1 – One or more error bits (8..10) are set within one or more of the bottom four entries of the receive FIFO, request interrupt, disable receive FIFO DMA service requests.
7.. 6	—	Reserved.

11.11.8 UART Status Register 1

UART status register 1 (UTSR1) contains flags that indicate when the UART is actively transmitting characters, that the transmit FIFO is not full, that the receive FIFO is not empty, and when parity, framing, overrun, and underrun errors have occurred. All bits within UTSR1 are read-only and are noninterruptible.

11.11.8.1 Transmitter Busy Flag (TBY) (read-only, noninterruptible)

The transmitter busy (TBY) flag is a read-only bit that is set when the transmitter is actively processing data for transmission (the serial shifter contains data), and is cleared when the transmitter is idle or is disabled (TXE=0). This bit does not request an interrupt.

11.11.8.2 Receive FIFO Not Empty Flag (RNE) (read-only, noninterruptible)

The receive FIFO not empty flag (RNE) is a read-only bit that is set when the receive FIFO contains one or more bytes of valid data and is cleared when it no longer contains any valid data. This bit can be polled when using programmed I/O to remove remaining bytes of data from the receive FIFO because DMA service and CPU interrupt requests are made only when 8, 7, 6, or 5 bytes reside within the FIFO. This bit does not request an interrupt.

11.11.8.3 Transmit FIFO Not Full Flag (TNF) (read-only, noninterruptible)

The transmit FIFO not full flag (TNF) is a read-only bit that is set when the transmit FIFO contains one or more entries that do not contain valid data and is cleared when the FIFO is completely full. This bit can be polled when using programmed I/O to fill the transmit FIFO over its halfway mark. This bit does not request an interrupt.

11.11.8.4 Parity Error Flag (PRE) (read-only, noninterruptible)

The parity error flag (PRE) is set when parity is enabled (PE = 1), and the parity type programmed using OES does not correspond to the parity check of the incoming serial data stream, which is calculated by the receive logic. The parity error bit is set when PE=1, OES=0, and UTDR<7:0>, and the incoming parity bit contain an even number of ones, or PE=1, OES=1, and UTDR<7:0>, and the incoming parity bit contain an odd number of ones.

The receive FIFO contains three bits (8, 9, and 10) that are not directly readable. The 8th bit in the FIFO is set at the top of the FIFO whenever a byte of data that incurs a parity error is moved from the receive serial shifter to the top of the receive FIFO. This tag travels along with the errant data value as it moves down the FIFO. Each time a data value is transferred to the bottom of the FIFO (caused by a read of the previous value), the state of this bit is moved from the FIFO to the PRE bit in the status register. After the error in FIFO (EIF) status bit is set, the user should always read UTSR1 first to check PRE before reading the data value from UDR because PRE corresponds to the current data byte at the bottom of the receive FIFO and is updated each time data is removed from the FIFO.

11.11.8.5 Framing Error Flag (FRE) (read-only, noninterruptible)

The framing error status bit (FRE) is set when the stop bit within a frame of incoming serial data is a zero instead of a one.

The receive FIFO contains three bits (8, 9, and 10) that are not directly readable. The 9th bit in the FIFO is set at the top of the FIFO whenever a byte of data that incurs a framing error is moved from the receive serial shifter to the top of the receive FIFO. This tag travels along with the errant data value as it moves down the FIFO. Each time a data value is transferred to the bottom of the FIFO (caused by a read of the previous value), the state of this bit is moved from the FIFO to the FRE bit in the status register. After the error in FIFO (EIF) status bit is set, the user should always read UTSR1 first to check FRE before reading the data value from UDR because FRE corresponds to the current data byte at the bottom of the receive FIFO and is updated each time data is removed from the FIFO.

11.11.8.6 Receiver Overrun Flag (ROR) (read-only, noninterruptible)

The receiver overrun status bit (ROR) is set when the receive logic attempts to place data into the receive FIFO after it has been completely filled.

The receive FIFO contains three bits (8, 9, and 10) that are not directly readable. The 10th bit in the FIFO is set within the top entry of the receive FIFO whenever an overrun occurs. This tag travels along with the last “good” data value before the overflow occurred as it moves down the FIFO. Each time a data value is transferred to the bottom of the FIFO (caused by a read of the previous value), the state of this bit is moved from the FIFO to the ROR bit in the status register, indicating that the next value in the FIFO is the last “good” piece of data before the overflow occurred. After the error in FIFO (EIF) status bit is set, the user should always read UTSR1 first to check ROR before reading the data value from UDR because ROR corresponds to the current data byte at the bottom of the receive FIFO and is updated each time data is removed from the FIFO.

The following table shows the bit locations corresponding to the flag bits within UART status register 1. Note that these flags do not generate interrupts, all bits are read-only, writes are ignored, and reads of reserved bits return zeros.

Address: 0h 8005 0020 **UTSR1** **Read-Only**

Bit	7	6	5	4	3	2	1	0
	Reserved		ROR	FRE	PRE	TNF	RNE	TBY
Reset	0	0	0	0	0	1	0	0

Bit	Name	Description
0	TBY	Transmitter busy flag (read-only). 0 – Transmitter is idle or UART is disabled. 1 – Transmit logic is currently transmitting a frame (data within the serial shifter); no interrupt generated.
1	RNE	Receive FIFO not empty (read-only). 0 – Receive FIFO is empty. 1 – Receive FIFO is not empty (no interrupt generated).
2	TNF	Transmit FIFO not full (read-only). 0 – Transmit FIFO is full. 1 – Transmit FIFO is not full (no interrupt generated).
3	PRE	Parity error (read-only). 0 – No parity errors encountered in the receipt of the next data value in the FIFO (or parity disabled). 1 – Parity error encountered in the receipt of the next data value in the FIFO (no interrupt generated).
4	FRE	Framing error (read-only). 0 – Stop bit for the next frame in the FIFO was a one. 1– Stop bit for the next frame in the FIFO was a zero (no interrupt generated).
5	ROR	Receive FIFO overrun (read-only). 0 – Receive FIFO has not experienced an overrun. 1 – Receive logic attempted to place data into receive FIFO while it was full, the next data value in the FIFO is the last piece of “good” data before the FIFO was overrun (no interrupt generated).
7..6	—	Reserved.

11.11.9 UART Register Locations

Table 11-18 shows the registers associated with serial port 3 and the physical addresses used to access them.

Table 11-18. Serial Port 3 Control, Data, and Status Register Locations

Address	Name	Description
0h 8005 0000	UTCR0	UART control register 0
0h 8005 0004	UTCR1	UART control register 1
0h 8005 0008	UTCR2	UART control register 2
0h 8005 000C	UTCR3	UART control register 3
0h 8005 0010	—	Reserved
0h 8005 0014	UTDR	UART data register
0h 8005 0018	—	Reserved
0h 8005 001C	UTSR0	UART status register 0
0h 8005 0020	UTSR1	UART status register 1
0h 8005 0024 – 0h 8005 FFFF	—	Reserved

11.12 Serial Port 4 – MCP / SSP

Serial port 4 contains two separate full-duplex synchronous serial interfaces. The multimedia communications port (MCP) provides an interface to the Philips UCB1100 and UCB1200 codecs. Both devices have an audio codec, a telecom codec, a touch-screen interface, four general-purpose analog-to-digital converter inputs, and ten programmable digital I/O lines. The MCP interface is used by the SA-1100 both to input and output digital data to and from the codec, and to configure and acquire status information from the codecs' 16 registers. The synchronous serial port (SSP) is used to interface to a variety of analog-to-digital converters, audio and telecom codecs, memory chips, and keypad controllers as well as other miscellaneous serial devices. The SSP supports the National Microwire and Texas Instruments* synchronous serial protocols as well as a subset of the Motorola* serial peripheral interface (SPI) protocol.

In MCP mode, serial port 4 controls communication between the SA-1100 and either the UCB1100 or UCB1200. The MCP produces two 64-bit subframes per frame (totalling 128 bits per frame) using a bit clock and frame synchronization signal. Data is communicated full-duplex via a separate transmit and receive data line. Selecting the on-chip clock, a bit clock frequency of either 9.585 Mbps or 11.981 Mbps can be programmed. Alternatively, GPIO pin 21 can be used to input a bit clock from an off-chip source. This feature allows users to select a frame rate that is an exact multiple of the desired audio/telecom sample rate. The MCP communicates to the codec in the first of the two subframes. The second subframe is used in high-end applications to communicate with a second stereo codec; however, this feature is not supported by the MCP. Each 64-bit subframe contains seven different fields of information. These fields include: audio conversion data, telecom conversion data, data valid flags, control register address, control register data, and read/write control. Both transmit and receive data contains these seven fields. The transmit frame contains data for D-to-A conversion as well as address, data, and control signals to write to or read from the codec's registers, and the receive frame contains A-to-D samples and the data returned from a read of a codec register.

Both the MCP and the off-chip codec contain programmable 7-bit divisors, one each for the telecom and audio data. These values are used to divide the bit clock to generate a desired sampling frequency. When the codec is enabled, the divisor pairs are synchronously transferred to their respective modulus registers within the MCP and off-chip codec, and decrement using the bit clock. This technique allows telecom and audio data to be transferred between the MCP and codec, lock-step in sync with the sampling/conversion frequency of the codec.

The MCP contains two pairs of transmit FIFOs and two pairs of receive FIFOs, one each for audio and telecom data, totalling four separate 8-entry x 16-bit FIFOs. The MCP also contains a 21-bit data register used to transmit codec register reads and writes, as well as another 21-bit register to receive the results of codec register reads. Touch-screen and ADC conversions are triggered, the digital I/O lines are controlled using codec register writes, and the converted data and the state of digital I/O lines are accessed using a codec register read.

In SSP mode, serial port 4 controls full-duplex synchronous serial transfers between the SA-1100 and off-chip devices that support National Microwire*, Texas Instruments* synchronous serial, or the Motorola* SPI protocol. The SSP functions as a master only and communicates to the off-chip slave device by driving a serial bit rate clock ranging from 7.2 kHz to 1.8432 MHz along with a frame synchronization pulse to denote the start of each frame transfer, and supports any data format between 4 and 16 bits. Transmit and receive data is stored/collected using two separate 8-entry x 16-bit FIFOs. MCP operation takes precedence over SSP operation. If use of both the MCP and SSP is required at the same time, the user can configure the SSP to take over control of GPIO pins 10 through 13, and the MCP uses the serial port 4 pins for transmission.

The external pins dedicated to this interface are TXD4, RXD4, SCLK, and SFRM. If use of both the MCP and SSP is not required and serial port 4 is disabled, control of these pins is given to the peripheral pin controller (PPC) to be used to perform general-purpose input/output (noninterruptible). See the section 11.13 on page 184 for a description of the programming and operation of the PPC. The MCP operation takes precedence over the SSP if both units are enabled. Both the MCP and SSP support word reads/writes of their registers, and half-word DMA transfers to or from their FIFOs that are 16-bits wide.

11.12.1 MCP Operation

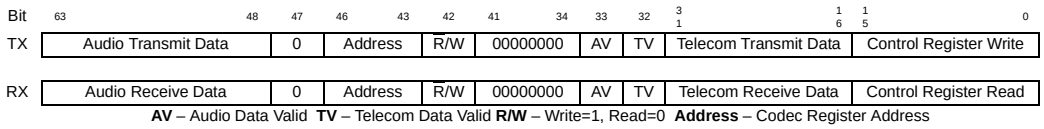
Following reset, both the MCP and SSP logic within serial port 4 is disabled and control of its pins is given to the PPC, which configures all four pins as inputs. To enable MCP operation, the programmer should first clear any interruptible status bits, which are set following the reset, by writing a one to them. Next, the user should program the MCP control register with the desired mode of operation using word writes, ensuring that the enable bit is programmed last. The user can choose to either “prime” the audio and telecom transmit FIFOs, before enabling the MCP, by writing up to eight 16-bit values each, or allow the FIFO service requests to interrupt the CPU or trigger a DMA transfer to fill the FIFOs. Once the off-chip codec is programmed and data resides within the bottom entries of the audio and/or telecom FIFOs, transmission/reception of data begins on the transmit (TXD4) and receive (RXD4) pins, and is synchronously controlled by the serial clock (SCLK) pin and a serial frame (SFRM) pin at a rate of 9.585 MHz or 11.981 MHz. The serial clock rate is selected by programming a control bit. Note that the two SCLK rates are derived by first multiplying the 3.6864-MHz on-chip oscillator by 13, then by dividing either by 5 (9.58464 MHz) or by 4 (11.9808 MHz). Also note that an off-chip clock can be used to drive the MCP when a sample rate that is not a multiple of 3.6864 MHz is required.

11.12.1.1 Frame Format

Each MCP data frame is 128 bits long and is divided into two subframes: 0 and 1. Subframe 0 is used by the MCP to communicate data to and from the UCB1100 or UCB1200. Subframe 1 is not used by the MCP because it is typically used to interface to high-performance stereo codecs such as Crystal's CS4216/18.

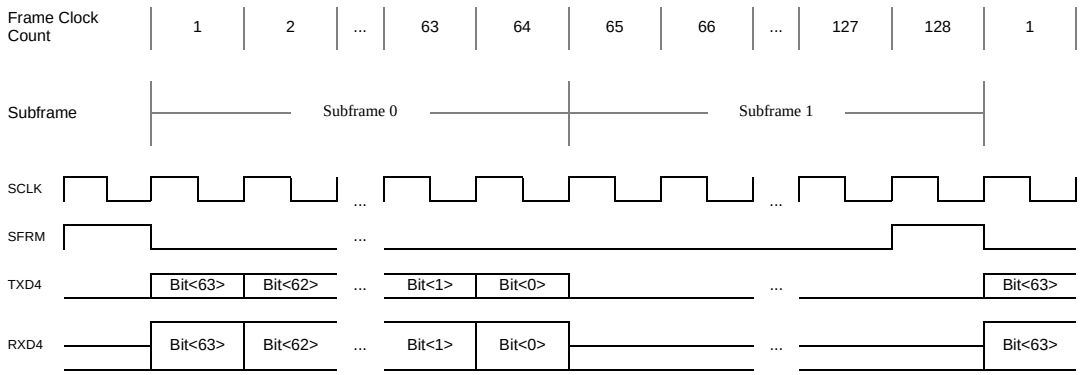
After the MCP is enabled, SCLK begins to transition at the programmed clock rate and the start of the first frame is signalled by pulsing the SFRM pin high for one SCLK period. The rising edge of SFRM coincides with the rising edge of SCLK. The SFRM pulse causes the MCP to transfer any available audio and/or telecom data from their respective transmit FIFOs to a 64-bit serial shifter, setting the appropriate audio/telecom valid flags as well. If the codec control register contains valid data, the register value and address are placed within the appropriate fields in the shifter, and the read/write bit is configured to indicate which type of register access is to be made. For any field that does not have valid data available, the previous value transmitted is used. As long as the MCP is enabled, data frames are continuously transferred, even if no valid data is available for transmission. The format of data transmitted and received in subframe 0 is shown in Figure 11-31. Note that the UCB110 or USB1200 data sheets use big-endian notation; little-endian notation is used in the following figure to remain consistent with the rest of the SA-1100 specification.

Figure 11-31. MCP Frame Data Format



Both the MCP and the off-chip codec drive data on the rising edge of SCLK and latch data on its falling edge. After SFRM is negated, subframe 0 begins and the data within the 64-bit shifter is driven onto the TXD4 pin a bit at a time, starting with the MSB (bit<63>). As each bit of data is shifted onto the TXD4 pin from one side of the shifter, a bit is also shifted into the opposite end of the shifter from the RXD4 pin. After 64 SCLK cycles elapse, all data within the shifter has been transmitted, and the shifter contains the 64-bit receive data frame. The MCP takes the data from each field and places it in its respective receive FIFO or data register. The next 64 **SCLK** cycles make up subframe 1. When subframe 1 is active, the clocks to all MCP resources that are not needed are turned off in order to conserve power. Figure 11-32 shows the pin timing of the MCP.

Figure 11-32. MCP Frame Pin Timing



Note that the transmit line is pulled low any time data is not being driven onto the pin. The UCB1100 and UCB1200 have a programming option that allows them to either tristate or drive the receive line low when data is not being driven onto RXD4. As shown in Figure 11-32, MCP frames occur back-to-back. The SFRM pin is pulsed high during the last clock (128th) of the frame to indicate the start of a new frame the following SCLK period. Values contained within the transmit FIFOs are loaded to the shift register on the rising edge of SFRM.

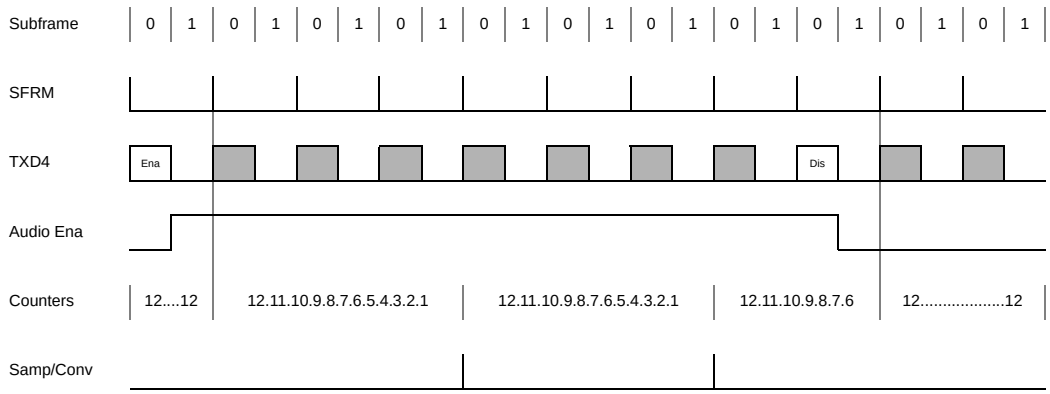
11.12.1.2 Audio and Telecom Sample Rates and Data Transfer

The UCB1100 and UCB1200 contain both an audio and telecom codec with sample rates that can be individually programmed, and are derived from the programmed serial clock (SCLK) that is supplied by the MCP. For the audio codec, the sample rate is derived by dividing the serial clock first by a fixed value of 32, then by a value from 6 to 127. The same is true for the telecom codec, except that the programmable divisor ranges from 16 to 127. The codec and the MCP *both* contain an audio and a telecom sample rate counter. These counters are used to achieve conversion rate synchronization between the codec and MCP so that data may be coherently transferred between the MCP and the codec. For the remainder of this description, references made to the audio codec also apply to the telecom portion of the codec and MCP.

Before enabling the audio codec, the audio sample rate counters within the codec and MCP must programmed with the same divisor value so that they have the same clock rate. The codec's audio sample rate divisor is programmed by issuing a control register write transfer, and the MCP's divisor is programmed using the CPU by writing to the MCP's control register. Both the MCP and the codec's audio counters are reloaded with the programmed modulus value any time the audio portion of the codec is enabled (which is also accomplished by performing a control register write transfer), or whenever the sample rate counters reach zero.

The MCP and the audio codec decrement their counters in lock-step with one another, both starting on the occurrence of the first SFRM pulse after the audio codec is enabled. Samples/conversions are made each time the audio codec's counter reaches zero. Figure 11-33 shows the timing of the audio codec enable and decrements of the MCP and audio codec's sample counter.

Figure 11-33. MPC/Codec Sampling Counter Synchronization



In the preceding figure, "Ena," within the data frame on TXD4, represents a control register write to the codec to enable the input portion of the audio codec. The register is updated with the write at the end of subframe and the audio enable signal within the codec goes high. Both the MCP and codec's audio sample rate counters then start to decrement on the next SFRM pulse. In the example, a divisor value of 12 is used, causing the counter to decrement to zero after 384 (32*12=384) SCLK cycles occur.

If the input portion of the audio codec is enabled, when the counter reaches zero, a sample and A-to-D conversion is made and the converted value is placed within the correct field of the codec's serial shift register for transmission back to the MCP in the next data frame. If the output portion of the audio codec is enabled, an audio data value is taken from the received data supplied by the MCP and is used for a D-to-A conversion. Data used in the D-to-A conversion is always taken from the previous MCP input frame. If no new data is available within the MCP's audio transmit FIFO since the last D-to-A conversion, then the same data is used again (causing audio distortion).

Samples and conversions occur twice in the preceding figure. However, while the counter is decrementing for the third time, the CPU disables the audio codec by issuing another control register write, represented by the "Dis" data frame on TXD4. The SFRM pulse following the write causes the disable to take effect, and the MCP and codec's audio sample rate counters are stopped and reset to their modulus values.

The MCP and the codec's audio sample rate counters must be enabled coherently so that synchronization is achieved between the two. This is accomplished by first programming both the MCP and codec's sample rate modulus values, then performing a codec control register write to enable the audio sampling rate counter within the codec. The MCP automatically decodes a write to the audio codec input and output enable bits, and enables the MCP's audio sample rate counter at the same time as the codec's counter to ensure synchronization.

The UCB1100 and UCB1200 each have an individual data valid bit for audio and telecom A/D samples. Whenever these bits are set in the data frame returned from the codec to the MCP, the audio and telecom data is taken from the frame and placed in their respective receive FIFOs. The UCB1100 and UCB1200 have two different modes of operation to control the setting of the audio and telecom data valid bits. In the first mode, a data valid bit is set any time a frame contains "reliable" data (the codec is enabled and at least one A-to-D sample has been taken). In this mode, once the data valid bit is set, it remains set until the codec A-to-D input is disabled. In the second mode, the codec only sets the data valid bit corresponding to a new A-to-D sample. Once the data is transmitted to the MCP within a receive data frame, the data valid bit is reset to zero for subsequent data frames until a new A-to-D sample is triggered.

11.12.1.3 MCP Transmit and Receive FIFO Operation

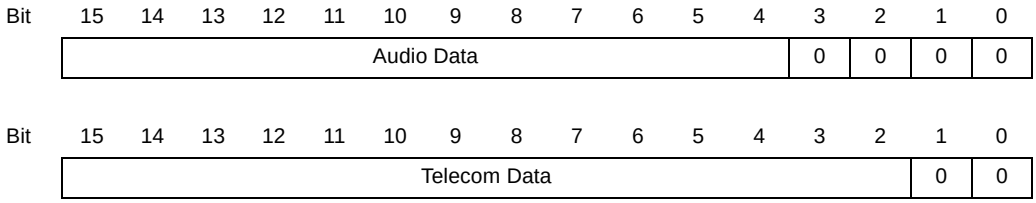
The MCP contains four 8-entry x 16-bit FIFOs: one for audio and one for telecom A-to-D samples received by the MCP, as well as one for audio and one for telecom D-to-A conversions transmitted to the codec. For the remainder of this description, references made to the audio codec also apply to the telecom portion of the codec and MCP.

For each incoming data frame, if the audio data valid bit is set, the 16-bit audio A-to-D sample is extracted and placed in the audio receive FIFO. Note that the MCP also supports a mode in which the audio data valid bit is ignored after the first conversion has been saved to the FIFO, and the MCP's audio sample rate counter is used to signal when a new A-to-D sample has been taken and is available within the incoming frame. Audio data is transferred from the incoming data frames to the receive FIFO only if the audio enable bit is set within the MCP's status register.

The MCP's audio and telecom sample rate counters are used to trigger when new D/A conversions are to be transmitted to the codec. The user should take care in ensuring sample rate counters in the MCP are synchronized with the respective sample rate counters in the codec as described in preceding sections. When the audio enable status bit transitions from a 0 to a 1 within the MCP status register, the next available entry of data is taken from the audio transmit FIFO and is placed within the correct field in the MCP's serial shifter. This value is then continuously transferred by the MCP in each data frame to the codec. The codec uses the value only when its audio sample rate counter decrements to zero. After the audio D-to-A conversion is made, both the codec and the MCP's audio sample rate counters reload with their modulus values. This reload triggers the audio transmit FIFO to transfer the next available entry of data to the MCP's serial shifter. Again, this value is continuously transmitted to the codec in each data frame until it is used in the next audio D-to-A conversion.

The width of each entry within the audio and telecom FIFOs is 16 bits. However, the audio codec's sample/conversion data size is 12 bits and the telecom is 14 bits. Conversions and samples are left justified within the 16-bit audio and telecom data fields in the MCP frame as well as within the transmit and receive FIFOs. Figure 11-34 shows the required data alignment for the transmit and receive audio and telecom FIFOs. The user must left justify data to be transmitted, and shift received data to the right before using the results.

Figure 11-34. Audio/Telecom Transmit/Receive FIFO Data Format



To reduce chip size as well as power consumption, the MCP's FIFOs use self-timed logic (not clocked). Because of process and environmental variations, the depth at which a service request is triggered to empty the receive FIFOs is variable. This variation spans a maximum of four FIFO entries, thus the audio and telecom receive FIFO service requests can be made at four different FIFO depths.To compensate for this variability and guarantee that at least four valid entries of data exist within either FIFO before generating a service request, an extra four entries have been added to both receive FIFOs (four entries more than the transmit FIFOs). Thus the audio and telecom transmit FIFOs are 8-entries deep and the audio and telecom receive FIFOs are 12-entries deep. The point at which the receive FIFO service requests are triggered spans one-third (four entries) of the 12-entry FIFOs. The service request is signalled at a depth from one-third full to two-thirds full (when the FIFOs contains five, six, seven, or eight entries of data).

11.12.1.4 Codec Control Register Data Transfer

The UCB1100 and UCB1200 contain sixteen 16-bit registers used to configure the chip, and store touch-screen and ADC samples as well as digital I/O pin state and edge interrupt status. These registers are read and written via the MCP's serial interface using three fields that exist within the MCP's data frame. In Figure 11-31, bits 15:0 contain the value read from or written to the off-chip codec, bits 46:43 contain the register address of the current read or write, and bit 42 is used by the MCP to signal a read or write cycle to the codec. These fields are configured by the CPU by writing to MCP control register 2, and are then transmitted to the off-chip codec. These fields are also received every data frame by the MCP from the codec and are placed in MCP control register 2, which can be read by the CPU. Note that the contents of the addressed register are returned in the receive data frame regardless of the state of the read/write bit. Thus for write cycles, both a write and a read occurs, and for read cycles, only a read occurs.

A register write is performed by writing a value to the MCP control register 2 that contains the value to store to the register, the address of the register, and the read/write bit set to one. Once this register is written, its contents are transferred to the correct fields within the serial shifter on the next rising edge of the SFRM signal. The register information is transmitted to the UCB1100 or UCB1200 during subframe 0, and the value is written to the selected codec register at the end of subframe 0 (during the 65th bit of the frame). The control register value and address are also returned to the MCP and stored in MCP control register 2. The read/write bit is zero in the return frame. Because the addressed register is updated at the end of subframe 0, the data returned during the frame in which the write occurred represents the *previous* contents of the register. The updated value is returned during the next data frame.

A register read is performed by writing a value to MCP data register 2 that contains the address of the register and the read/write bit set to a zero. Again, the data is transferred to the serial shifter on the next rising edge of the SFRM signal and is transmitted to the UCB1100 or UCB1200 during subframe 0. Because the address and read/write control bit fields occur near the beginning of the serial stream output, the codec performs the read immediately after the read/write bit is received (during the 41st bit of the frame) and the value contained within the addressed register is sent back to the MCP in the *same* data frame.

Once the codec control register is written with a value to execute a read or write, the operation is performed every MCP data frame until a new value is written to the register. Thus, continual reads or writes are made to the addressed codec register until a new read or write operation is configured.

11.12.1.5 External Clock Operation

Under normal operation, the MCP is programmed to use one of two on-chip clocks to produce a 9.585-Mbps or 11.981-Mbps bit rate. This clock is also used to increment the audio and telecom sample rate counters. The MCP also supports a special mode that allows the user to control the MCP's frame rate and audio/telecom sample rates. This mode is useful when sample rates that are not an integer multiple of 12 MHz are required. In this mode, the MCP uses GPIO<21> to input a clock supplied from off-chip. The frequency of the off-chip clock can be any value within the allowable frequency range of the UCB100, up to 12 MHz. When using GPIO pin 21 for the input clock, the user must also set bit 21 of the GPIO alternate function register (GAFR) and clear bit 21 of the GPIO pin direction register (GPDR). See the Section 9.1, "General-Purpose I/O" on page 9-1.

11.12.1.6 Alternate SSP Pin Assignment

MCP operation takes precedence over SSP operation. Thus if both are enabled, serial port 4 defaults to MCP mode. However, if the MCP and SSP both need to be used at the same time, general-purpose I/O pins 10..13 (GPIO<10..13>) can be reassigned by programming the PPC pin assignment register (PPAR). This allows the MCP dedicated use of the four pins assigned to serial port 4, and the SSP dedicated use of the GPIO pins. When the SSP pin reassignment (SPR) bit is set in PPAR, the following pin assignments are made: GPIO<10> is used for transmit, GPIO<11> for receive, GPIO<12> for serial clock, and GPIO<13> for serial frame. Note that the user must also set bits 10 through 13 in the GPIO alternate function register (GAFR) as well as set bits 10, 12, and 13, and clear bit 11 in the GPIO pin direction register (GPDR). Once the reassignment is made, these pins are no longer usable by the GPIO unit. See the Section 9.1, "General-Purpose I/O" on page 9-1 for a description of how to program the system control module and the Section 11.13, "Peripheral Pin Controller (PPC)" on page 11-184 for a description of how to program the PPC unit.

11.12.1.7 CPU and DMA Register Access Sizes

Bit positioning and addressing of the MCP is described in terms of little endian ordering. All MCP registers are 32 bits wide. The ARM peripheral bus does not support byte or half-word operations. All reads and writes of the MCP by the CPU should be wordwide. Four separate dedicated DMA requests exist for the audio and telecom transmit and receive FIFOs. If the DMA controller is used to service the transmit and/or receive FIFOs, the user must ensure the DMA is properly configured to perform half-word accesses, using 4 half-words per burst (half the size of the FIFOs). Note that a separate set of registers also exist to configure SSP operation. See the following sections for a full description of programming and operation of serial port 4 as an SSP, a summary of serial port 4's MCP registers, and a summary of its SSP registers.

11.12.2 MCP Register Definitions

There are six registers within the MCP: two control registers, three data registers, and one status register. The control register is used to program the audio and telecom sample rates, to mask or unmask interrupt requests to service the MCP's FIFOs, to select whether an on-chip or off-chip clock is used to drive the bit rate, and to enable/disable operation. The first data register addresses the top of the audio transmit FIFO and the bottom of the audio receive FIFO. Likewise, the second data register addresses the top/bottom of the telecom transmit/receive FIFOs, respectively. A read accesses the receive FIFOs; a write accesses the transmit FIFOs. Note that these are four physically separate FIFOs to allow full-duplex transmission. The third data register is 21 bits and is used to transmit read and write operations to the codec's control, data, and status registers. Values written to the register are used in the transmit data frame and values read are taken from the received data frame. The status register contains bits that signal FIFO overrun and underrun errors, and transmit and receive FIFO service requests. Each of these status conditions signals an interrupt request to the interrupt controller. The status register also flags when audio and telecom transmit FIFOs are not full, when the audio and telecom receive FIFOs are not empty, when a codec control register read or write is complete, and when the audio or telecom portion of the codec is enabled (no interrupt generated).

11.12.3 MCP Control Register

The MCP control register (MCCR) contains 11 different bit fields that control various functions within the MCP.

11.12.3.1 Audio Sample Rate Divisor (ASD)

The 7-bit audio sample rate divisor (ASD) bit field is used to synchronize the MCP with the sample rate of the audio codec. Sample rate synchronization is required such that the MCP's audio transmit FIFO logic knows when to load a new value for D-to-A conversion to the MCP's serial shifter for transmission. This field is programmed with the same value that is written to the codec's sample rate divisor via a codec control register write. When the audio codec is enabled, the first audio transmit value is placed in the serial output stream by the transmit FIFO, and both the MCP's and codec's sample rate counters begin to decrement in lock-step with one another. When the audio codec's counter decrements to zero, it uses the value transmitted to it by the MCP to perform the D-to-A conversion. After the conversion is made, the MCP and codec's counters reset to their modulus values, and the MCP's audio transmit FIFO loads the next value to the serial shifter for transmission. This new value is then transmitted to the audio codec and is used for the next D-to-A conversion, which is signalled when the sample rate counter decrements to zero again.

A total of 122 different audio sample rates can be selected, ranging from a minimum of 2.358 K samples per second using the 9.585-MHz internal clock to a maximum of 62.401 K samples per second using the 11.981-MHz internal clock. Note that slower sample rates can be achieved using an externally supplied clock. The sample rate clock generator uses either a 9.585-MHz or 11.981-MHz clock produced by the on-chip PLL or the clock supplied to the MCP via GPIO pin 21, and is divided by a fixed value of 32 and then by the programmable ASD value to generate the audio sample clock. This clock is automatically enabled when:

- A codec control register write to the audio control register B is made (address=0b100), which sets *either* the audio codec input or output enable bits (bit 14 = aud_in_ena, bit 15 = aud_out_ena), followed by
- The rising edge of the next **SFRM** pulse after the write has been made.

Once enabled, the MCP's audio sample rate clock decrements at the programmed frequency with a 50% duty cycle. The action outlined in the above first bullet item causes the MCP's audio transmit FIFO logic to transfer the next available value to the audio data field within the serial shifter. Each time the audio sample rate clock decrements to zero, it is reloaded with its programmed ASD modulus value, triggers the audio transmit FIFO logic to transfer the next available value to the audio data field within the serial shifter, and continues to decrement. The MCP's audio sample rate clock is automatically disabled when:

- A codec control register write to the audio control register B is made (address=0b100), which clears *both* the audio codec input and output enable bits (bit 14 = aud_in_ena, bit 15 = aud_out_ena), followed by
- The rising edge of the next SFRM pulse after the write has been made.

The resultant audio sample clock rate, given a specific ASD value, can be calculated using the following equation, where ASD is the decimal equivalent of the binary value programmed within the bit field. Note that ASD must be programmed with a value of 6 or larger. Unpredictable results occur for ASD values smaller than 6. Note that one of three clock frequencies can be selected. The first two frequencies are internal clocks selected by the CFS bit in MCCR1 and the third frequency is a user-defined clock that is input via GPIO pin 21 and is divided as defined by the ECP bit field described in following sections.

$$SampleRate = \frac{12 \times 10^6}{32 x ASD}$$

Valid ASD values are from 6 (00000110) to 127 (11111111)

Note: The $12x10^6$ value within the formula's numerator should be replaced with the frequency of the clock driven to GPIO pin 21 when an off-chip clock source is used to drive the MCP.

11.12.3.2 Telecom Sample Rate Divisor (TSD)

The 7-bit telecom sample rate divisor (TSD) bit field is used to synchronize the MCP with the sample rate of the telecom codec. The telecom sample rate clock is required for the same reason and works exactly like the audio sample rate clock, except for one minor difference. The valid TSD values range from 16 to 127 (instead of 6), allowing a total of 112 different audio sample rates to be selected, ranging from a minimum of 2.358 K samples per second using the 9.585-MHz internal clock to a maximum of 23.400 K samples per second using the 11.98-MHz internal clock. Note that slower sample rates can be achieved using an externally supplied clock.

The resultant telecom sample clock rate, given a specific TSD value, can be calculated using the following equation, where TSD is the decimal equivalent of the binary value programmed within the bit field. Note that TSD must be programmed with a value of 16 or larger. Unpredictable results occur for TSD values smaller than 16. Note that one of three clock frequencies can be selected. Thr first two frequencies are internal clocks selected by the CFS bit in MCCR1 and the third frequency is a user-defined clock that is input via GPIO pin 21 and is divided by the ECP bit field described in the following sections.

$$SampleRate = \frac{12 \times 10^6}{32 x TSD}$$

Valid TSD values are from 16 (00010000) to 127 (11111111)

Note: The $12x10^6$ value within the formula's numerator should be replaced with the frequency of the clock driven to GPIO pin 21 when an off-chip clock source is used to drive the MCP.

11.12.3.3 Multimedia Communications Port Enable (MCE)

The MCP enable (MCE) bit is used to enable and disable all MCP operation. Since the MCP and SSP both share the same pins, only one can be enabled at a time. If the user enables both at the same time, the MCP has precedence and the SSP remains disabled. However, both can be enabled when the SSP pin reassignment (SPR) bit within the PPC unit is set, which assigns the SSP to GPIO pins. See the following sections for a description of the SSP enable (SSE) bit.

When the MCP is disabled, all of its clocks are powered down to minimize power consumption. If the SSP is also disabled, the TXD4, RXD4, SCLK, and SFRM pins can be used for general-purpose input/output. See the Section 11.13, “Peripheral Pin Controller (PPC)” on page 11-184 for a description of how to program the PPC unit to reassign the SSP’s pins and to use serial port 4’s pins as I/Os. Note that MCE and CFS are the only control bits within the MCP that are reset to a known state. MCE is cleared to zero to ensure the MCP is disabled following a reset of the SA-1100.

When the MCP is enabled, SCLK begins to transition and the start of the first frame is signalled by pulsing the SFRM pin high for one SCLK period. The rising edge of SFRM coincides with the rising edge of SCLK. As long as the MCE bit is set, the MCP operates continuously, transmitting and receiving 128 bit data frames. When the MCE bit is cleared, the MCP is disabled immediately, causing the current frame, which is being transmitted, to be terminated and control of serial port 4’s pins to be given to the PPC unit. Clearing MCE resets the MCP’s FIFOs. However, MCP data register 3, the control, and the status registers are not reset. The user must ensure these registers are properly reconfigured before reenabling the MCP.

11.12.3.4 External Clock Select (ECS)

The external clock select (ECS) bit selects whether one of the two on-chip clocks derived by the 3.6864-MHz oscillator is used by the MCP or if an off-chip clock is supplied via GPIO pin 21. When ECS=0, the MCP can be programmed to select one of two frequencies: either 9.585 MHz or 11.981 MHz. This clock is also used to increment the audio and telecom sample rate counters. (See preceding sections.) When ECS=1, the MCP uses GPIO<21> to input a clock supplied from off-chip. The frequency of the off-chip clock after being scaled by the ECP bit field can be any value within the allowable frequency range of the UCB100 up to 12 MHz. This off-chip clock is useful when a sample rate frequency, which is not a multiple of 9.585 MHz or 11.981 MHz is required for synchronization with either the audio and/or telecom portion of the UCB1100 or UCB1200 codecs. When using GPIO pin 21 for the input clock, the user must also set bit 21 of the GPIO alternate function register (GAFR) and clear bit 21 of the GPIO pin direction register (GPDR). See the Section 9.1, “General-Purpose I/O” on page 9-1.

11.12.3.5 A/D Sampling Mode (ADM)

The A/D sampling mode (ADM) bit selects whether the MCP takes audio and telecom data from the incoming frame only when their respective data valid bits are set or whenever the MCP’s audio and telecom sample rate counters time-out, indicating that the data in the next incoming frame is valid. When ADM=0, data is taken from the incoming frame and is placed into the audio or telecom FIFO whenever the incoming audio or telecom data valid bit is set. When ADM=1, after the MCP is enabled, data is taken from the incoming frame when the data valid bit is set for the *first* time. After this point, the data valid bit is ignored, and samples are stored each time the audio or telecom sample rate counters decrement to zero, indicating that a new A-to-D sample was taken and will be available in the next frame.

The UCB1100 and UCB1200 have two different modes of operation to control the setting of the audio and telecom data valid bits. In one mode, the codec only sets the data valid bit when a new A-to-D sample is contained within the incoming data frame. Once the data is transmitted to the

MCP within a receive data frame, the data valid bit is reset to zero for subsequent data frames until a new A-to-D sample is triggered and transmitted to the MCP. In this mode, the user should program ADM=0. In the other mode, the data valid bit is set once when the first A-to-D conversion is made and is placed in the receive data frame. However, the data valid bit *remains* set and the MCP cannot determine when new A-to-D conversions are available within the incoming frame. Programming ADM=1 prevents multiple copies of the same A-to-D conversion to be placed in the FIFO, storing samples only when the sample rate counter times out.

11.12.3.6 Telecom Transmit FIFO Interrupt Enable (TTE)

The telecom transmit FIFO interrupt enable (TTE) bit is used to mask or enable the telecom transmit FIFO service request interrupt. When TTE=0, the interrupt is masked and the state of the telecom transmit FIFO service request (TTS) bit within the MCP status register is ignored by the interrupt controller. When TTE=1, the interrupt is enabled, and whenever TTS is set (one), an interrupt request is made to the interrupt controller. Note that programming TTE=0 does not affect the current state of TTS or the telecom transmit FIFO logic's ability to set and clear TTS; it only blocks the generation of the interrupt request. Also note that TTE does not affect generation of the telecom transmit FIFO DMA request, which is asserted any time TTS=1.

11.12.3.7 Telecom Receive FIFO Interrupt Enable (TRE)

The telecom receive FIFO interrupt enable (TRE) bit is used to mask or enable the telecom receive FIFO service request interrupt. When TRE=0, the interrupt is masked, and the state of the telecom receive FIFO service request (TRS) bit within the MCP status register is ignored by the interrupt controller. When TRE=1, the interrupt is enabled, and whenever TRS is set (one), an interrupt request is made to the interrupt controller. Note that programming TRE=0 does not affect the current state of TRS or the telecom receive FIFO logic's ability to set and clear TRS; it only blocks the generation of the interrupt request. Also note that TRE does not affect generation of the telecom receive FIFO DMA request, which is asserted any time TRS=1.

11.12.3.8 Audio Transmit FIFO Interrupt Enable (ATE)

The audio transmit FIFO interrupt enable (ATE) bit is used to mask or enable the audio transmit FIFO service request interrupt. When ATE=0, the interrupt is masked and the state of the audio transmit FIFO service request (ATS) bit within the MCP status register is ignored by the interrupt controller. When AT=1, the interrupt is enabled, and whenever ATS is set (one), an interrupt request is made to the interrupt controller. Note that programming ATE=0 does not affect the current state of ATS or the audio transmit FIFO logic's ability to set and clear ATS; it only blocks the generation of the interrupt request. Also note that ATE does not affect generation of the audio transmit FIFO DMA request, which is asserted any time ATS=1.

11.12.3.9 Audio Receive FIFO Interrupt Enable (ARE)

The audio receive FIFO interrupt enable (ARE) bit is used to mask or enable the audio receive FIFO service request interrupt. When ARE=0, the interrupt is masked, and the state of the audio receive FIFO service request (ARS) bit within the MCP status register is ignored by the interrupt controller. When ARE=1, the interrupt is enabled, and whenever ARS is set (one), an interrupt request is made to the interrupt controller. Note that programming ARE=0 does not affect the current state of ARS or the audio receive FIFO logic's ability to set and clear ARS; it only blocks the generation of the interrupt request. Also note that ARE does not affect generation of the audio receive FIFO DMA request, which is asserted any time ARS=1.

11.12.3.10 Loopback Mode (LBM)

The loopback mode (LBM) bit is used to enable and disable the ability of the MCP's transmit and receive logic to communicate. When LBM=0, the MCP operates normally. The transmit and receive data paths are independent and communicate via their respective pins. When LBM=1, the output of the serial shifter (MSB) is directly connected to the input of the serial shifter (LSB) internally and control of the TXD4, RXD4, SCLK, and SFRM pins are given to the peripheral pin control (PPC) unit.

11.12.3.11 External Clock Prescaler (ECP)

The 2-bit external clock select (ECP) field is used to divide the clock input via GPIO pin 21 when the external clock function is enabled. When ECS=1, ECP is decoded to divide the clock input on the **GPIO<21>** pin by 1, 2, 3, or 4 before being used to drive the MCP's frame rate. When ECP=00, the input clock is divided by 1; when ECP=01, it is divided by 2; when ECP=10, it is divided by 3; and when ECP=11, it is divided by 4. Note that the ECP bit field is ignored when the internal clock (ECS=0) is used to drive the MCP's frame rate. Also note that the resultant clock frequency *after* the divide has taken place can be any value within the allowable frequency range of the UCB1100 or UCB1200 (up to 12 MHz).

The following table shows the bit locations corresponding to the 10 different control bit fields within the MCP control register. Note that the MCE bit is the only control bit that is reset to a known state to ensure the MCP is disabled following a reset of the SA-1100. The reset state of all other control bits is unknown (indicated by question marks) and must be initialized before enabling the MCP. The user can program all 11 bit fields and enable the MCP using a single word write to MCCR0. Writes to reserved bits are ignored and reads return zeros.

Address: 0h 8006 0000 **MCP Control Register 0: MCCR** **Read/Write**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved							ECP		LBM	ARE	ATE	TRE	TTE	ADM	ECS	MCE
Reset	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Res.	TSD							Res.	ASD						
Reset	0	?	?	?	?	?	?	?	0	?	?	?	?	?	?	?

Bit	Name	Description
6..0	ASD	Audio sample rate divisor. Value (from 6 to 127) used to match the sample rate of the audio codec within the UCB1100 or UCB1200 to time when audio D/A data should be supplied by the audio transmit FIFO. Sample Rate = Programmed clock rate/(32xASD), where ASD is a decimal value.
7	—	Reserved.
14..8	TSD	Telecom sample rate divisor. Value (from 16 to 127) used to match the sample rate of the telecom codec within the UCB1100 or UCB1200 to time when telecom D/A data should be supplied by the telecom transmit FIFO. Sample Rate = Programmed clock rate/(32xTSD), where TSD is a decimal value.
15	—	Reserved.

Bit	Name	Description
16	MCE	Multimedia communications port enable. 0 – MCP operation disabled, control of the TXD4, RXD4, SCLK, and SFRM pins given to the PPC to be used as general-purpose I/O pins. 1 – MCP operation enabled. Note that the MCP has precedence over the SSP, if MCE=1; SSE is ignored unless the SPR bit is set within the PPC, which allows the SSP to use GPIO pins while the MCP uses serial port 4's pin for transmission.
17	ECS	External clock select. 0 – on-chip clock used to produce the frame rate as further programmed by the CFS control bit in MCCR1. It is also used to clock the audio and telecom sample rate counters. 1 – Clock input using GPIO pin 21 to select a frame rate that is an exact multiple of the desired audio/telecom sample rate. Frame Rate = Input Clock Freq /(ECP x 32). Sample Rate = Input Clock Freq /(ECP x 32 x ASD or TSD).
18	ADM	A/D data sampling mode. 0 – Audio and telecom receive data is stored to their respective FIFOs whenever their receive data valid bits are valid. 1– Audio and telecom receive data is stored when the receive data valid bit is set the first time, and from that point on whenever the MCP's audio and telecom sample rate counters time out.
19	TTE	Telecom transmit FIFO interrupt enable. 0 – Telecom transmit FIFO half-full or less condition does not generate an interrupt (TTS bit ignored). 1 – Telecom transmit FIFO half-full or less condition generates an interrupt (state of TTS sent to interrupt controller).
20	TRE	Telecom receive FIFO interrupt enable. 0 – Telecom receive FIFO one- to two-thirds full or more condition does not generate an interrupt (TRS bit ignored). 1 – Telecom receive FIFO one- to two-thirds full or more condition generates an interrupt (state of TRS sent to interrupt controller).
21	ATE	Audio transmit FIFO interrupt enable. 0 – Audio transmit FIFO half-full or less condition does not generate an interrupt (ATS bit ignored). 1 – Audio transmit FIFO half-full or less condition generates an interrupt (state of ATS sent to interrupt controller).
22	ARE	Audio receive FIFO interrupt enable. 0 – Audio receive FIFO one- to two-thirds full or more condition does not generate an interrupt (ARS bit ignored). 1 – Audio receive FIFO one- to two-thirds full or more condition generates an interrupt (state of ARS sent to interrupt controller).
23	LBM	Loopback mode. 0 – Normal serial port operation enabled. 1 – Output of serial shifter is connected to input of serial shifter internally and control of TXD4, RXD4, SCLK, and SFRM pins is given to the PPC unit.
25..24	ECP	External clock prescaler. 00 – Clock input using GPIO pin 21 is divided by one before being used to drive the frame rate. 00 – Clock input using GPIO pin 21 is divided by two before being used to drive the frame rate. 00 – Clock input using GPIO pin 21 is divided by three before being used to drive the frame rate. 00 – Clock input using GPIO pin 21 is divided by four before being used to drive the frame rate. **Note:** ECP is used only when ECS=1. Also, the maximum clock frequency allowed to drive the frame rate after ECS has divided down the input clock is 12 MHz.
31.. 26	—	Reserved.

11.12.4 MCP Control Register 1

The MCP control register 1 (MCCR1) contains one bit that selects one of two fixed frequencies to drive the MCP. Note that this register resides within the PPC's address space.

11.12.4.1 Clock Frequency Select (CFS)

When the on-chip clock is enabled (ECS=0), the clock frequency select (CFS) bit is used to select either a 9.585-MHz or an 11.981-MHz clock to drive the MCP's serial clock rate. When ECS=0 and CFS=0, the on-chip 3.6864-MHz oscillator is first multiplied by 13 then divided by 4, resulting in an 11.9808-MHz bit clock frequency. When ECS=0 and CFS=1, the on-chip 3.6864 MHz oscillator is first multiplied by 13 then divided by 5, resulting in a 9.58464-MHz bit clock frequency. Note that when ECS=1, CFS is ignored and an external clock is input to the MCP via GPIO pin 21. Also note that CFS is cleared following a reset of the SA-1100 so that the MCP defaults to 11.981-MHz operation, which is standard for the UCB1100/1200.

The following table shows the location of the CFS control bit within the MCP control register 1. The CFS is cleared to zero selecting 11.981-MHz operation following a reset of the SA-1100. Writes to reserved bits are ignored and reads return zeros. MCCR1 resides within the PPC's address space.

Address: 0h 9006 0030 **MCP Control Register 1: MCCR1** **Read/Write**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved											CFS	Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit	Name	Description
19..0	—	Reserved.
20	CFS	Clock frequency select. 0 – If ECS=0, bit rate clock frequency of 11.981 MHz is selected. 1 – If ECS=0, bit rate clock frequency of 9.585 MHz is selected. If ECS=1, CFS is ignored and an external clock supplied by **GPIO** pin 21 is used.
31..21	—	Reserved.

11.12.5 MCP Data Registers

The MCP contains three data registers. MCDR0 addresses the top entry of the audio transmit FIFO and bottom entry of the audio receive FIFO, MCDR1 addresses the top and bottom entries of the telecom transmit and receive FIFOs respectively, and MCDR2 is used to perform reads and writes to any of the codec's 16 registers via the MCP's serial interface.

11.12.5.1 MCP Data Register 0

When MCP data register 0 (MCDR0) is read, the bottom entry of audio receive FIFO is accessed. As data is removed by the MCP's receive logic from the incoming data frame, it is placed into the top entry of the audio receive FIFO and is transferred down an entry at a time until it reaches the last empty location within the FIFO. Data is removed by reading MCDR, which accesses the bottom entry of the audio FIFO. After MCDR0 is read, the bottom entry is invalidated and all remaining values within the FIFO automatically transfer down one location.

When MCDR0 is written, the topmost entry of the audio transmit FIFO is accessed. After a write, data is automatically transferred down to the lowest location within the transmit FIFO, which does not already contain valid data. Data is removed from the bottom of the FIFO one value at a time by the transmit logic, is loaded into the correct position within the 64-bit transmit serial shifter, and then is serially shifted out onto the TXD4 pin during subframe 0.

Audio data is 12 bits wide and must be left justified by the user before writing it to the transmit FIFO (MSB of audio data corresponds to bit 16 of transmit FIFO). The lower four bits of the FIFO are automatically zero filled by the transmit logic when a 16-bit value is written to MCDR0 for transmission. The UCB1100 or UCB1200 automatically forces bits 0 through 3 to zero before transmitting the value to the MCP. The user must right justify received audio data before using it.

The following table shows MCDR0. Note that the transmit and receive audio FIFOs are cleared when the SA-1100 is reset or by writing a zero to MCE (MCP disabled). Also note that writes to reserved bits are ignored and reads return zeros.

Address: 0h 8006 0008 **MCP Data Register 0: MCDR0** **Read/Write**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Bottom of Audio Receive FIFO												0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Read Access

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Top of Audio Transmit FIFO												0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Write Access

Bit	Name	Description
3..0	—	Reserved for future enhancements. Read – Data returned, but UCB1100 and UCB1200 currently zero fill these four bits. Write – MCP's transmit logic automatically zero fills these bits.
15..4	Audio Data	Transmit/receive audio FIFO data. Read – Bottom of audio receive FIFO data. Write – Top of audio transmit FIFO data.
31..16	—	Reserved.

11.12.5.2 MCP Data Register 1

When MCP data register 1 (MCDR1) is read, the bottom entry of the telecom receive FIFO is accessed. As data is removed by the MCP's receive logic from the incoming data frame, it is placed into the top entry of the telecom receive FIFO and is transferred down an entry at a time until it reaches the last empty location within the FIFO. Data is removed by reading MCDR1, which accesses the bottom entry of the telecom FIFO. After MCDR1 is read, the bottom entry is invalidated, and all remaining values within the FIFO automatically transfer down one location.

When MCDR1 is written, the topmost entry of the telecom transmit FIFO is accessed. After a write, data is automatically transferred down to the lowest location within the transmit FIFO, which does not already contain valid data. Data is removed from the bottom of the FIFO one value at a time by the transmit logic, is loaded into the correct position within the 64-bit transmit serial shifter, and then is serially shifted out onto the TXD4 pin during subframe 0.

Telecom data is 14 bits wide and must be left justified by the user before writing it to the transmit FIFO (MSB of telecom data corresponds to bit 16 of transmit FIFO). The lower two bits of the FIFO are automatically zero filled by the transmit logic when a 16-bit value is written to MCDR1 for transmission. The UCB1100 or UCB1200 automatically forces bits 0 and 1 to zero before transmitting the value to the MCP. The user must right justify received telecom data before using it.

The following table shows MCDR1. Note that the transmit and receive telecom FIFOs are cleared when the SA-1100 is reset, or by writing a zero to MCE (MCP disabled). Also note that writes to reserved bits are ignored and reads return zeros.

Address: 0h 8006 000C **MCP Data Register 1: MCDR1** **Read/Write**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Bottom of Telecom Receive FIFO														0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Read Access

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Top of Telecom Transmit FIFO														0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Write Access

Bit	Name	Description
1..0	—	Reserved for future enhancements. Read – Data returned, but UCB1100 or UCB1200 currently zero fills these two bits. Write – MCP's transmit logic automatically zero fills these bits.
15..2	Telecom Data	Transmit/receive telecom FIFO data. Read – Bottom of telecom receive FIFO data. Write – Top of telecom transmit FIFO data.
31..16	—	Reserved.

11.12.5.3 MCP Data Register 2

MCDR2 contains 21 bits and is used to perform reads and writes to any of the UCB1100's or UCB1200's registers. MCDR2 contains three separate fields: MCDR2<15:0> is the 16-bit register data field, MCDR2<16> is a 1-bit read/write control bit, and MCDR2<20:17> is the 4-bit register address field. A value written to MCDR2 is placed in the correct position within the 64-bit subframe 0, is transmitted to the off-chip codec, and is used to perform a read or write operation to the addressed codec register. Note that the contents of the addressed register are always returned in the receive data frame and placed in the MCDR2 regardless of the state of the read/write bit. Thus for write cycles, both a write and a read occurs, and for read cycles, only a read occurs. When MCDR2 is read, the value returned from the last read or write operation, which was completed to the codec, is returned.

A register write is performed by writing the correct value to each of the three fields within MCDR2 using one 16- or 32-bit write, ensuring that the read/write bit is set. Its contents are then transferred to the correct fields within the serial shifter on the next rising edge of the SFRM signal, and then to the codec via the TXD4 pin during subframe 0. The value within MCDR2<15:0> is written to the selected codec register at the end of subframe 0 (during the 65th bit of the frame). The data written to the control register and its address is returned to the MCP during the *next* data frame, and is placed back within MCDR2 with the read/write bit reset to zero. For a write operation, since the addressed register is written at the end of subframe 0, the data returned during the frame in which the write occurred represents the *previous* contents of the register. The updated value is returned during the next data frame.

A register read is performed by writing the address of the register to read while clearing the read/write bit to zero within MCDR2. Again, the data is transferred to the serial shifter on the next rising edge of the SFRM signal and is transmitted to the UCB1100 or UCB1200 during subframe 0. Because the address and read/write control bit fields are placed near the beginning of the serial stream output, the codec performs the read immediately after the read/write bit is received (during the 41st bit of the frame), and the value contained within the addressed register is sent back to the MCP in the *same* data frame, and is placed within MCDR2.

Once MCDR2 is written with a value to execute a read or write, the operation is performed every MCP data frame until a new value is written to the register. Thus continual reads or writes are made to the addressed codec register until a new read or write operation is configured.

The following table shows the location of MCP data register 2. Note that the reset state of all MCDR2 bits is unknown (indicated by question marks), writes to reserved bits are ignored, and reads return zeros.

Address: 0h 8006 0010 **MCP Data Register 2: MCDR2** **Read/Write**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved											Reg Address R/W				0
Reset	0	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Data Value Returned by a Codec Register Read or Write															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Read Access

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved											Reg Address R/W				$\overline{R/W}$
Reset	0	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Data Value to be Written to the Addressed Codec Register															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Write Access

Bit	Name	Description
15..0	Codec Register Read/ Write Data	Codec register read/write data. Read – If a codec write was last performed, contains data of previous register access; next frame contains the data that was written. If a codec read was last performed, contains data from the read register. Write – Used to specify what data to write to the addressed register, ignored for a codec register read.
16	R/W	Read/write. Read – Returns a zero. Write – Used to control whether the addressed register is read or written (write = 1, read = 0).
20..17	Codec Register Read/ Write Address	Codec register read/write address. Read – If a codec write was last performed, contains address of previous register access; next frame contains the address of the write. If a codec read was last performed, contains address of the register read. Write – Used to address a register to perform a read or write.
31.. 21	—	Reserved.

11.12.6 MCP Status Register

The MCP status register (MCSR) contains bits that signal FIFO overrun and underrun errors, and FIFO service requests. Each of these conditions signal an interrupt request to the interrupt controller. The status register also flags when transmit FIFOs are not full, when the receive FIFOs are not empty, when a codec control register read or write is complete, and when the audio or telecom portion of the codec is enabled (no interrupt generated).

A bit that can cause an interrupt signals the interrupt request as long as the bit is set. Once the bit is cleared, the interrupt is cleared. Read/write bits are called status bits; read-only bits are called flags. Status bits are referred to as “sticky” (once set by hardware, must be cleared by software). Writing a one to a sticky status bit clears it; writing a zero has no effect. Read-only flags are set and cleared by hardware; writes have no effect. Additionally, some bits that cause interrupts have corresponding mask/enable bits in the control register and are indicated in the following section headings. Note that the user has the ability to mask all MCP interrupts by clearing bit 18 within the interrupt controller mask register (ICMR). See the Section 9.2, “Interrupt Controller” on page 9-11.

11.12.6.1 Audio Transmit FIFO Service Request Flag (ATS) (read-only, maskable interrupt)

The audio transmit FIFO service request flag (ATS) is a read-only bit that is set when the audio transmit FIFO is nearly empty and requires service to prevent an underrun. ATS is set any time the audio transmit FIFO has four or fewer entries of valid data (half-full or less), and is cleared when it has five or more entries of valid data. When the ATS bit is set, an interrupt request is made unless the audio transmit FIFO interrupt request mask (ATE) bit is cleared. The state of ATS is also sent to the DMA controller, and can be used to signal a DMA service request. Note that ATE has no effect on the generation of the DMA service request. After the DMA or CPU fills the FIFO such that four or more locations are filled within the audio transmit FIFO, the ATS flag (and the service request and/or interrupt) is automatically cleared.

11.12.6.2 Audio Receive FIFO Service Request Flag (ARS) (read-only, maskable interrupt)

The audio receive FIFO service request flag (ARS) is a read-only bit that is set when the audio receive FIFO is nearly filled and requires service to prevent an overrun. ARS is set whenever the audio receive FIFO has four or more entries of valid data (half-full or more), and is cleared when it has three or fewer (less than half-full) entries of data. When the ARS bit is set, an interrupt request is made unless the audio receive FIFO interrupt request mask (ARE) bit is cleared. The state of ARS is also sent to the DMA controller, and can be used to signal a DMA service request. Note that ARE has no effect on the generation of the DMA service request. After the DMA or CPU fills the FIFO such that four or more locations are filled within the receive FIFO, the ARS flag (and the service request and/or interrupt) is automatically cleared.

11.12.6.3 Telecom Transmit FIFO Service Request Flag (TTS) (read-only, maskable interrupt)

The telecom transmit FIFO service request flag (TTS) is a read-only bit that is set when the telecom transmit FIFO is nearly empty and requires service to prevent an underrun. TTS is set whenever the telecom transmit FIFO has four or fewer entries of valid data (half-full or less), and is cleared when it has five or more entries of valid data. When the TTS bit is set, an interrupt request is made unless the telecom transmit FIFO interrupt request mask (TTE) bit is cleared. The state of TTS is also sent to the DMA controller, and can be used to signal a DMA service request. Note that TTE has no effect on the generation of the DMA service request. After the DMA or CPU fills the FIFO such that four or more locations are filled within the telecom transmit FIFO, the TTS flag (and the service request and/or interrupt) is automatically cleared.

11.12.6.4 Telecom Receive FIFO Service Request Flag (TRS) (read-only, maskable interrupt)

The telecom receive FIFO service request flag (TRS) is a read-only bit that is set when the telecom receive FIFO is nearly filled and requires service to prevent an overrun. TRS is set whenever the telecom receive FIFO has four or more entries of valid data (half-full or more), and is cleared when it has three or fewer (less than half-full) entries of data. When the TRS bit is set, an interrupt request is made unless the telecom receive FIFO interrupt request mask (TRE) bit is cleared. The state of TRS is also sent to the DMA controller, and can be used to signal a DMA service request. Note that TRE has no effect on the generation of the DMA service request. After the DMA or CPU fills the FIFO such that four or more locations are filled within the receive FIFO, the TRS flag (and the service request and/or interrupt) is automatically cleared.

11.12.6.5 Audio Transmit FIFO Underrun Status (ATU) (read/write, nonmaskable interrupt)

The audio transmit FIFO underrun status bit (ATU) is set when the audio transmit logic attempts to fetch data from the FIFO after it has been completely emptied. When an underrun occurs, the audio transmit logic continuously transmits the last valid audio value, which was transmitted before the underrun occurred. Once data is placed in the FIFO and it is transferred down to the bottom, the audio transmit logic uses the new value within the FIFO for transmission. When the ATU bit is set, an interrupt request is made.

11.12.6.6 Audio Receive FIFO Overrun Status (ARO) (read/write, nonmaskable interrupt)

The audio receive FIFO overrun status bit (ARO) is set when the audio receive logic attempts to place data into the audio receive FIFO after it has been completely filled. Each time a new piece of data is received, the set signal to the ARO status bit is asserted, and the newly received data is discarded. This process is repeated for each new piece of data received until at least one empty FIFO entry exists. When the ARO bit is set, an interrupt request is made.

11.12.6.7 Telecom Transmit FIFO Underrun Status (TTU) (read/write, nonmaskable interrupt)

The telecom transmit FIFO underrun status bit (TTU) is set when the telecom transmit logic attempts to fetch data from the FIFO after it has been completely emptied. When an underrun occurs, the telecom transmit logic continuously transmits the last valid telecom value, which was transmitted before the underrun occurred. Once data is placed in the FIFO and it is transferred down to the bottom, the telecom transmit logic uses the new value within the FIFO for transmission. When the TTU bit is set, an interrupt request is made.

11.12.6.8 Telecom Receive FIFO Overrun Status (TRO) (read/write, nonmaskable interrupt)

The telecom receive FIFO overrun status bit (TRO) is set when the telecom receive logic places data into the telecom receive FIFO after it has been completely filled. Each time a new piece of data is received, the set signal to the TRO status bit is asserted, and the newly received data is discarded. This process is repeated for each new piece of data received until at least one empty FIFO entry exists. When the TRO bit is set, an interrupt request is made.

11.12.6.9 Audio Transmit FIFO Not Full Flag (ANF) (read-only, noninterruptible)

The audio transmit FIFO not full flag (ANF) is a read-only bit that is set whenever the audio transmit FIFO contains one or more entries that do not contain valid data and is cleared when the FIFO is completely full. This bit can be polled when using programmed I/O to fill the audio transmit FIFO over its halfway mark. This bit does not request an interrupt.

11.12.6.10 Audio Receive FIFO Not Empty Flag (ANE) (read-only, noninterruptible)

The audio receive FIFO not empty flag (ANE) is a read-only bit that is set whenever the audio receive FIFO contains one or more entries of valid data and is cleared when it no longer contains any valid data. This bit can be polled when using programmed I/O to remove remaining bytes of data from the receive FIFO because DMA service and CPU interrupt requests are made only when four or more bytes reside within the FIFO (3, 2, or 1 bytes may remain at the end of a frame). This bit does not request an interrupt.

11.12.6.11 Telecom Transmit FIFO Not Full Flag (TNF) (read-only, noninterruptible)

The telecom transmit FIFO not full flag (TNF) is a read-only bit that is set whenever the telecom transmit FIFO contains one or more entries that do not contain valid data and is cleared when the FIFO is completely full. This bit can be polled when using programmed I/O to fill the telecom transmit FIFO over its halfway mark. This bit does not request an interrupt.

11.12.6.12 Telecom Receive FIFO Not Empty Flag (TNE) (read-only, noninterruptible)

The telecom receive FIFO not empty flag (TNE) is a read-only bit that is set whenever the telecom receive FIFO contains one or more entries of valid data and is cleared when it no longer contains any valid data. This bit can be polled when using programmed I/O to remove remaining bytes of data from the receive FIFO because DMA service and CPU interrupt requests are made only when four or more bytes reside within the FIFO (3, 2, or 1 bytes may remain at the end of a frame). This bit does not request an interrupt.

11.12.6.13 Codec Write Completed Flag (CWC) (read-only, noninterruptible)

The codec write completed (CWC) flag is set after the following sequence occurs: a register write command is issued to the codec by writing to MCDR2; the write command is sent to the codec via subframe 0; the data value is latched within the addressed codec register at the beginning of subframe 1 (the 65th bit of the frame); the address and value that was written is returned to the MCP via the next subframe 0; and the returned values are latched in MCDR2. CWC is automatically cleared when MCDR2 is read or written. This bit does not request an interrupt.

11.12.6.14 Codec Read Completed Flag (CRC) (read-only, noninterruptible)

The codec read completed (CRC) flag is set after the following sequence occurs: a register read command is issued to the codec by writing to MCDR2; the read command is sent to the codec via subframe 0; the data value contained within the addressed codec register is loaded into the codec's serial shift register during subframe 0 (the 41st bit of the frame); the address and value that was read is returned to the MCP via the same subframe 0; and the returned values are latched in MCDR2. CRC is automatically cleared when MCDR2 is read or written. This bit does not request an interrupt.

11.12.6.15 Audio Codec Enabled Flag (ACE) (read-only, noninterruptible)

The audio codec enabled (ACE) flag indicates when the audio codec input and/or output is enabled, which in turn, indicates that the audio sample rate counter is enabled. This flag is set after the following sequence occurs: a register write command is issued to Audio Control Register B (register 8), and either bit 14 or 15 is set (aud_in_ena or aud_out_ena) by writing to MCDR2; the write command is sent to the codec via subframe 0; the data value is latched within codec register 8; and **SFRM** is asserted to indicate the start of the next frame. ACE is automatically cleared using the same sequence with the exception that bits 14 and 15 are cleared, disabling both the input and output paths of the audio codec. This bit does not request an interrupt.

11.12.6.16 Telecom Codec Enabled Flag (TCE) (read-only, noninterruptible)

The telecom codec enabled (TCE) flag indicates when the telecom codec input and/or output is enabled, which in turn, indicates that the telecom sample rate counter is enabled. This flag is set after the following sequence occurs: a register write command is issued to Telecom Control Register B (register 6), and either bit 14 or 15 is set (tel_in_ena or tel_out_ena) by writing to MCDR2; the write command is sent to the codec via subframe 0; the data value is latched within codec register 6; and SFRM is asserted to indicate the start of the next frame. TCE is automatically cleared using the same sequence with the exception that bits 14 and 15 are cleared, disabling both the input and output paths of the telecom codec. This bit does not request an interrupt.

The following table shows the bit locations corresponding to the status and flag bits within the MCP status register. MCSR contains a collection of read/write, read-only, interruptible, and noninterruptible bits (refer to the bit descriptions above). Writes to read-only bits have no effect. The user must clear set status bits before enabling the MCP. Note that writes to reserved bits are ignored and reads return zeros; question marks indicate that the values are unknown at reset.

Address: 0h 8006 0018 **MCP Status Register: MCSR** **Read/Write & Read-Only**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TCE	ACE	CRC	CWC	TNE	TNF	ANE	ANF	TRO	TTU	ARO	ATU	TRS	TTS	ARS	ATS
Reset	0	0	0	0	0	1	0	1	?	?	?	?	0	0	0	0

Bit	Name	Description
0	ATS	Audio transmit FIFO service request flag (read-only). 0 – Audio transmit FIFO is more than half-full (five or more entries filled) or MCP disabled. 1 – Audio transmit FIFO is half-full or less (four or fewer entries filled) and MCP operation is enabled, DMA service request signalled, interrupt request signalled if not masked (if ATE=1).
1	ARS	Audio receive FIFO service request (read-only). 0 – Audio receive FIFO is less than half-full (three or fewer entries filled) or MCP disabled. 1 – Audio receive FIFO is half-full or more (four or more entries filled) and MCP operation is enabled, DMA service request signalled, interrupt request signalled if not masked (if ARE=1).
2	TTS	Telecom transmit FIFO service request flag (read-only). 0 – Telecom transmit FIFO is more than half-full (five or more entries filled) or MCP disabled. 1 – Telecom transmit FIFO is half-full or less (four or fewer entries filled) and MCP operation is enabled, DMA service request signalled, interrupt request signalled if not masked (if TTE=1).
3	TRS	Telecom receive FIFO service request (read-only). 0 – Telecom receive FIFO is less than half full (three or fewer entries filled) or MCP disabled. 1 – Telecom receive FIFO is half full or more (four or more entries filled) and MCP operation is enabled, DMA service request signalled, interrupt request signalled if not masked (if TRE=1).
4	ATU	Audio transmit FIFO underrun. 0 – Audio transmit FIFO has not experienced an underrun. 1 – Audio transmit logic attempted to fetch data from transmit FIFO while it was empty request interrupt.
5	ARO	Audio receive FIFO overrun. 0 – Audio receive FIFO has not experienced an overrun. 1 – Audio receive logic attempted to place data into receive FIFO while it was full, request interrupt.

Bit	Name	Description
6	TTU	Telecom transmit FIFO underrun. 0 – Telecom transmit FIFO has not experienced an underrun. 1 – Telecom transmit logic attempted to fetch data from transmit FIFO while it was empty, request interrupt.
7	TRO	Telecom receive FIFO overrun. 0 – Telecom receive FIFO has not experienced an overrun. 1 – Telecom receive logic attempted to place data into receive FIFO while it was full, request interrupt.
8	ANF	Audio transmit FIFO not full (read-only). 0 – Audio transmit FIFO is full. 1– Audio transmit FIFO is not full.
9	ANE	Audio receive FIFO not empty (read-only). 0 – Audio receive FIFO is empty. 1 – Audio receive FIFO is not empty.
10	TNF	Telecom transmit FIFO not full (read-only). 0 – Telecom transmit FIFO is full. 1 – Telecom transmit FIFO is not full.
11	TNE	Telecom receive FIFO not empty (read-only). 0 – Telecom receive FIFO is empty. 1 – Telecom receive FIFO is not empty.
12	CWC	Codec write completed (read-only). 0 – A write to a codec register has not completed since the last time this bit was cleared. 1 – A write to a codec register has been transmitted and has updated the register.
13	CRC	Codec read completed (read-only). 0 – The value read from the addressed codec register has not been returned to MCDR2. 1 – The value read from the addressed codec register is now in MCDR2.
14	ACE	Audio codec enabled (read-only). 0 – The audio codec input and output is disabled (bits 14 and 15 are 0 in Audio Control Reg B). 1 – Audio codec input and/or output is enabled (bits 14 and/or 15 is 1 in Audio Control Reg B).
15	TCE	Telecom codec enabled. 0 – The telecom codec input and output is disabled (bits 14 and 15 are 0 in Telecom Cntl Reg B). 1 – Telecom codec input and/or output is enabled (bits 14 and/or 15 is 1 in Telecom Cntl Reg B).
31..16	—	Reserved.

11.12.7 SSP Operation

Following reset, both the MCP and SSP logic within serial port 4 is disabled and control of its pins is given to the PPC that configures all four pins as inputs. To enable SSP operation, the programmer should first clear any interruptible status bits, which are set following the reset by writing a one to them. Next, the user should program the SSP's control registers with the desired mode of operation, ensuring that the register containing the SSP enable bit is programmed last. Note that the MCP has precedence over the SSP and must be disabled first before enabling the SSP. The user can choose to either "prime" the transmit FIFO by writing up to eight 16-bit values, or allow the transmit FIFO service request to interrupt the CPU or trigger a DMA transfer to fill the FIFO. Once enabled, transmission/reception of data begins on the transmit (TXD4) and receive (RXD4) pins, and is synchronously controlled by the serial clock (SCLK) and serial frame (SFRM) pins.

11.12.7.1 Frame Format

Each data frame is between 4 and 16 bits long depending on the size of data programmed, and is transmitted starting with the MSB. There are three basic frame types that can be selected: Motorola* SPI, Texas Instruments* synchronous serial, and National Microwire*. For all three formats, the serial clock (SCLK) is held low or inactive, while the SSP is idle and transitions at the programmed frequency only during active transmission of data. For Motorola* SPI and National Microwire* frame formats, the serial frame (SFRM) pin is active low, and is asserted (pulled down) during the entire frame's transmission. In these modes, the SFRM pin is used to select the off-chip slave serial device, enabling it for transmission. For Texas Instruments* format, the SFRM pin is pulsed for one serial clock period starting at its rising edge, prior to each frame's transmission. The type of serial clock edges used to drive and sample data are different for all three modes. For National Microwire* format, both the SSP and the off-chip slave device drive their output data on the falling edge of SCLK, and latch data from the other device on the rising edge. For Texas Instruments* format, both the SSP and the off-chip slave device drive their output data on the rising edge of SCLK, and latch data from the other device on the falling edge. For Motorola* SPI format, the user has the option of which edge of SCLK to drive and sample data, as well as the phase of the SCLK signal (whether it is shifted one-half period to the left or right during the frame transmission).

Unlike the full-duplex transmission of the other two frame formats, the National Microwire* format uses a special master-slave messaging technique that operates at half-duplex. In this mode, when a frame begins, an 8-bit control message is transmitted to the off-chip slave. During this transmit, no incoming data is received by the SSP. After the message has been sent, the off-chip slave decodes it and responds with the requested data after waiting one serial clock after the last bit of the 8-bit control message has been sent. The returned data can be 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

Figure 11-35 shows the Texas Instruments* synchronous serial frame format for a single transmitted frame and when back-to-back frames are transmitted. In this mode, SCLK and SFRM are forced low, and the transmit data line SA-1100. Once the bottom entry of the transmit FIFO contains data, SFRM is pulsed high for one SCLK period and the value to be transmitted is transferred from the transmit FIFO to the transmit logic's serial shift register. On the next rising edge of SCLK, the MSB of the 4- to 16-bit data frame is shifted to the TXD4 pin. Likewise, the MSB of the received data is shifted onto the RXD4 pin by the off-chip serial slave device. Both the SSP and the off-chip serial slave device then latch each data bit into their serial shifter on the falling edge of each SCLK. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SCLK after the LSB has been latched. Note that the transmit pin retains the last value it transmits (the value of bit <0>, when the frame completes and the SSP enters idle mode). If the SSP is disabled or a reset occurs, the transmit pin is reset to zero.

Figure 11-35. Texas Instruments* Synchronous Serial Frame Format

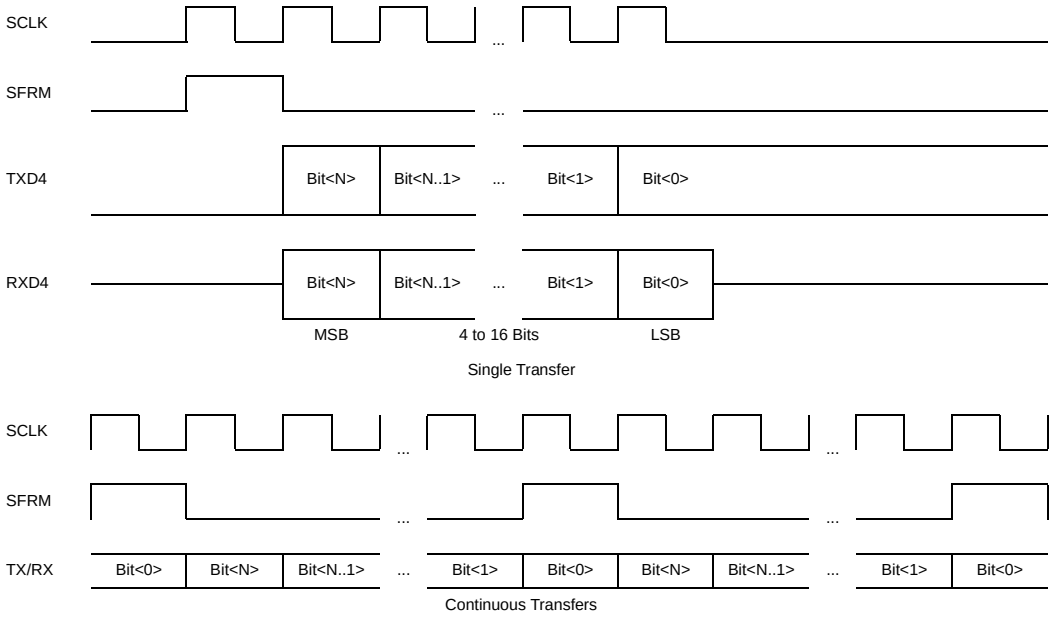
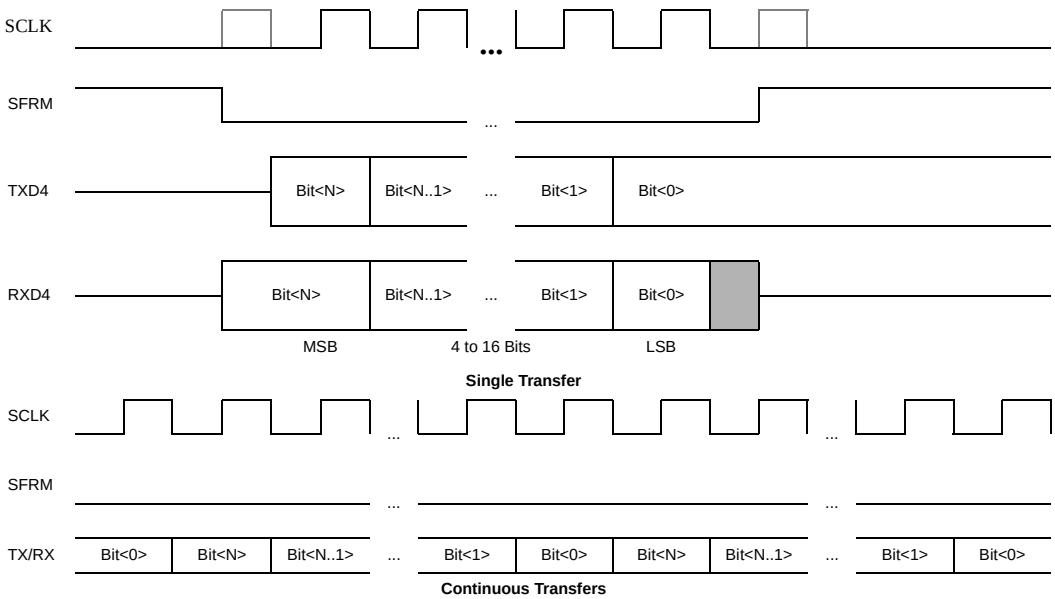


Figure 11-36 shows one of the four possible configurations for the Motorola* SPI frame format for a single transmitted frame and when back-to-back frames are transmitted. In this mode, SCLK and the transmit data line (TXD4) are forced low and SFRM is forced high, whenever the SSP is disabled or the SA-1100 is reset. Once the bottom entry of the transmit FIFO contains data, SFRM is pulled low and remains low for the duration of the frame's transmission. The falling edge of SFRM causes the value for transmission to be transferred from the bottom transmit FIFO entry to the transmit logic's serial shift register, and the MSB of the 4- to 16-bit data frame is shifted onto the TXD4 pin a half an SCLK period later (note that the SCLK pin does not transition at this point). The MSB of the received data is shifted onto the RXD4 pin by the off-chip serial slave device as soon as the serial framing signal goes low. Both the SSP and the off-chip serial slave device then latch each data bit into their serial shifter on the rising edge of each SCLK. At the end of the frame, the SFRM pin is pulled high one SCLK period after the last bit has been latched in the receive serial shifter, which causes the data to be transferred to the receive FIFO. Note that the off-chip slave device can tristate the receive line either on the falling edge of SCLK after the LSB has been latched by the receive shifter or when the SFRM pin goes high. Also note that the transmit pin retains the last value it transmits (the value of bit <0>, when the frame completes and the SSP enters idle mode). If the SSP is disabled or a reset occurs, the transmit pin is reset to zero. All four frame programming options are described in the SSP Control Register 1 section.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer; however, the SFRM line is continuously asserted (held low) and transmission of data occurs back-to-back (the MSB of the next frame follows directly after the LSB of the previous frame). In this example, each of the received data values is transferred from the receive shifter to the receive FIFO on the falling edge SCLK after the LSB of the frame has been latched into the SSP.

Figure 11-36. Motorola* SPI Frame Format



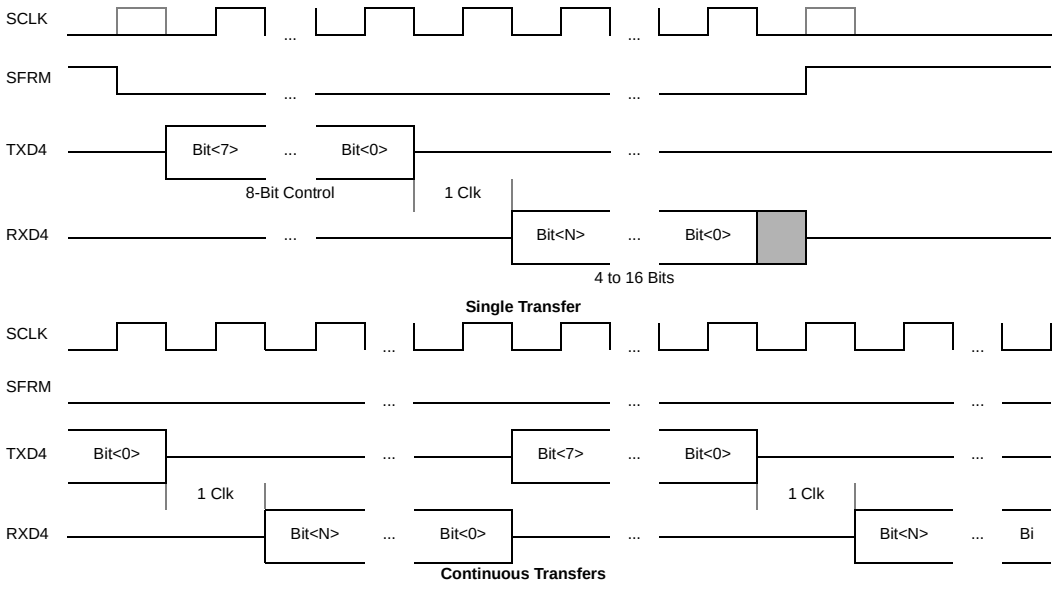
Note: The phase and polarity of **SCLK** can be configured for four different modes. This example shows just one of those modes. See the Section 11.12.10, "SSP Control Register 1" on page 11-177 for a complete description of each mode.

Figure 11-37 shows the National Microwire* frame format for a single transmitted frame and when back-to-back frames are transmitted. Microwire format is very similar to SPI format, except that transmission is half- instead of full-duplex, using a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSP to the off-chip slave device. During this transmit, no incoming data is received by the SSP. After the message has been sent, the off-chip slave decodes it and responds with the requested data after waiting one serial clock after the last bit of the 8-bit control message has been sent. The returned data is 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

SCLK and the transmit data line (TXD4) is forced low, and SFRM is forced high whenever the SSP is disabled or following a reset of the SA-1100. Once enabled, transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of SFRM causes the value contained within the bottom entry of the transmit FIFO to be transferred to the transmit logic's serial shift register and the MSB of the 8-bit control frame to be shifted onto the TXD4 pin. SFRM remains low for the duration of the frame's transmission. The RXD4 pin remains tristated during this transmission. The off-chip serial slave device latches each control bit into its serial shifter on the rising edge of each SCLK. After the last bit is latched by the slave device, the control byte is decoded during a one-clock waitstate, and the slave responds by transmitting data back to the SSP, driving each bit onto the RXD4 line on the falling edge of SCLK. The SSP, in turn, latches each bit on the rising edge of SCLK. At the end of the frame, for single transfers, the SFRM signal is pulled high one SCLK period after the last bit has been latched in the receive serial shifter, which causes the data to be transferred to the receive FIFO. Note that the off-chip slave device can tristate the receive line either on the falling edge of SCLK after the LSB has been latched by the receive shifter or when the SFRM pin goes high. Also note that the transmit pin retains the last value it transmits (the value of bit <0>, when the frame completes and the SSP enters idle mode). If the SSP is disabled or a rest occurs, the transmit pin is reset to zero.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer; however, the SFRM line is continuously asserted (held low) and transmission of data occurs back-to-back (the control byte of the next frame follows directly after the LSB of the received data from the previous frame). Each of the received data values is transferred from the receive shifter on the falling edge SCLK after the LSB of the frame has been latched into the SSP.

Figure 11-37. National Microwire* Frame Format



11.12.7.2 Baud Rate Generation

The baud or bit rate is derived by dividing down the 3.6864-MHz clock generated by the on-chip PLL. The clock is first divided by a fixed value of 2 and then by a programmable number between 1 and 256. This programmability provides a range of transmission rates ranging from 7.2 Kbps to 1.8432 Mbps. The resultant clock is used to drive the SCLK pin and by the transmit and receive logic's serial shifters to drive and latch data, respectively.

11.12.7.3 SSP Transmit and Receive FIFOs

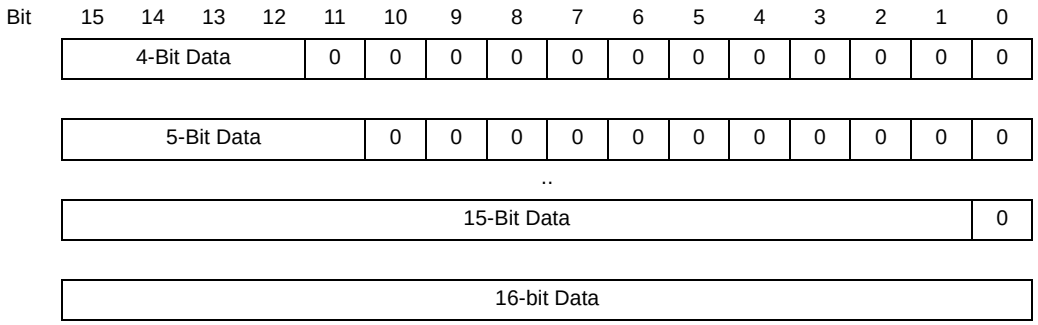
To reduce chip size as well as power consumption, the SSP's FIFOs use self-timed logic (they are not clocked). Because of process and environmental variations, the depth at which a service request is triggered to empty the receive FIFO is variable. This variation spans a maximum of four FIFO entries, thus the receive FIFO service request can be made at four different FIFO depths. To compensate for this variability and guarantee that at least four valid entries of data exist within the FIFO before generating a service request, an extra four entries have been added to the receive FIFO (four entries more than the transmit FIFO). Thus the transmit FIFO is 8 entries deep and the receive FIFO is 12 entries deep. The point at which the receive FIFO service request is triggered spans one-third (four entries) of the 12-entry FIFO. The service request is signalled at a depth from one-third full to two-thirds full (when the FIFO contains five, six, seven, or eight entries of data).

This service request variation only applies to an empty FIFO that is filled (receive FIFO). It does not apply to a full FIFO that is emptied (transmit FIFO). Thus the transmit FIFO is guaranteed to signal a service request when it has four or more empty entries and negate the request when the FIFO contains five or more entries that are filled.

If the DMA is used to service either one or both of the SSP's FIFOs, the burst size must be set to four half-words, even though more than four entries of data may exist within the receive FIFO. If programmed I/O is used to service the FIFOs, a maximum of four words may be added to the transmit FIFO without checking if more space is available. Likewise, a maximum of four words may be removed from the receive FIFO without checking if more data is available. After this point, the user must poll a set of status bits, which indicates if any data remains in the receive FIFO or if space is available in the transmit FIFO, before emptying or filling the FIFOs any further.

The width of each entry within the FIFOs is 16 bits. However, the SSP supports data sizes of 4 through 16 bits. Any data that is less than 16-bits wide must be left-justified when writing or DMAing data to the transmit FIFO. Likewise, data received by the SSP is left-justified when it is placed within the receive FIFO. Figure 11-38 shows the required data alignment for the transmit and receive FIFOs. The user must left-justify data to be transmitted, and shift received data to the right before using the results.

Figure 11-38. Transmit/Receive FIFO Data Format



11.12.7.4 CPU and DMA Register Access Sizes

Bit positioning, byte ordering, and addressing of the SSP are described in terms of little endian ordering. All SSP registers are 16-bits wide and are located in the least significant half-word of individual words. The ARM peripheral bus does not support byte or half-word operations. All reads and writes of the SSP by the CPU should be word wide. Two separate dedicated DMA requests exist for both the transmit and the receive FIFO. If the DMA controller is used to service the transmit and/or receive FIFOs, the user must ensure the DMA is properly configured to perform half-word wide accesses, using four half-words per burst (half the size of the FIFOs). Byte-wide DMA accesses for data widths of 4..8 bits are not permitted. For all data sizes 4..16 bits, the user must left-justify the data within each individual half-word in external memory for the DMA, starting with the most significant bit. Likewise, when using programmed I/O to service the SSP's transmit FIFO, the user must also left-justify the data written or read to/from the data register. Note that a separate set of registers also exist to configure MCP operation. See the following sections for a full description of programming and operation of serial port 4 as an MCP, a summary of serial port 4's MCP registers, and for a summary of its SSP registers.

11.12.7.5 Alternate SSP Pin Assignment

If the SSP and MCP both need to be used at the same time, general-purpose I/O pins 10 through 13 (GPIO<10-13>) can be reassigned by programming the PPC pin assignment register (PPAR). This allows the MCP dedicated use of the four pins assigned to serial port 4, and the SSP dedicated use of the GPIO pins. When the SSP pin reassignment (SPR) bit is set in PPAR, the following pin assignments are made: GPIO<10> is used for transmit, GPIO<11> for receive, GPIO<12> for serial clock, and GPIO<13> for serial frame. Note that the user must also set bits 10 through 13 in the GPIO alternate function register (GAFR) as well as set bits 10, 12, and 13 and clear bit 11 in the GPIO pin direction register (GPDR). Once the reassignment is made, these pins are no longer usable by the GPIO unit. See the "General-Purpose I/O" on page 9-1 for a description of how to program the system control module and the Section 11.13, "Peripheral Pin Controller (PPC)" on page 11-184 for a description of how to program the PPC unit.

11.12.8 SSP Register Definitions

There are four registers within the SSP: two control registers, one data register, and one status register. The control registers are used to program the baud rate, data length, and frame format, and to select whether the CPU or DMA is used to service the SSP, and to enable/disable operation. The data register is 16 bits and addresses both the transmit and receive buffers. A read accesses the receive buffer; a write accesses the transmit buffer. Note that these are two physically separate buffers to allow full-duplex transmission. The status register contains bits that signal an overrun error, a transmit buffer service request, and a receive buffer service request. Each of these status conditions signal an interrupt request to the interrupt controller. The status register also flags when the SSP is actively transmitting data, when the transmit FIFO is not full, and when the receive FIFO is not empty (no interrupt generated).

11.12.9 SSP Control Register 0

The SSP control register 0 (SSCR0) contains four different bit fields that control various functions within the SSP.

11.12.9.1 Data Size Select (DSS)

The 4-bit data size select (DSS) field is used to select the size of the data transmitted and received by the SSP. Data can be 4 to 16 bits in length. When data is programmed to be less than 16 bits, received data is automatically right justified and the upper bits in the receive FIFO are zero filled by the receive logic. Transmit data must be right justified by the user before being placed into the transmit FIFO; however, the upper unused bits are ignored by the SSP's transmit logic. Although it is possible to program data sizes of 1, 2, and 3 bits, these sizes are reserved and produce unpredictable results in the SSP. When National Microwire* frame format is selected, this bit field selects the size of the received data. Note that the size of the transmitted data is always 8 bits in this mode.

11.12.9.2 Frame Format (FRF)

The 2-bit frame format (FRF) bit field is used to select which frame format to use: Motorola* SPI (FRF=00), Texas Instruments* synchronous serial (FRF=01), or National Microwire* (FRF=10). See the preceding sections for a complete description of each frame format. Note that FRF=11 is reserved and produces unpredictable results.

11.12.9.3 Synchronous Serial Port Enable (SSE)

The SSP enable (SSE) bit is used to enable and disable all SSP operation. When SSE=0, the SSP is disabled; when SSE=1, it is enabled. Since the MCP and SSP both share the same pins, only one can be enabled at a time. If the user enables both at the same time, the MCP has precedence and the SSP remains disabled. However, both can be enabled when the SSP pin reassignment (SPR) bit within the PPC unit is set, which assigns the SSP to GPIO pins.

When the SSP is disabled, all of its clocks are powered down to minimize power consumption. If the MCP is also disabled, the TXD4, RXD4, SCLK, and SFRM pins can be used for general-purpose input/output. See the Section 11.13, "Peripheral Pin Controller (PPC)" on page 11-184 for a description of how to program the PPC unit to reassign the SSP's pins and use serial port 4's pins as I/Os. Note that SSE is the only control bit within the SSP that is reset to a known state. It is cleared to zero to ensure the SSP is disabled following a reset of the SA-1100.

When the SSE bit is cleared during active operation, the SSP is disabled immediately, causing the current frame, which is being transmitted, to be terminated and control of serial port 4's pins to be given to the PPC unit. Clearing SSE resets the SSP's FIFOs. However the SSP's control and status registers are not reset. The user must ensure these registers are properly reconfigured before reenabling the SSP.

11.12.9.4 Serial Clock Rate (SCR)

The 8-bit serial clock rate (SCR) bit field is used to select the baud or bit rate of the SSP. A total of 256 different bit rates can be selected, ranging from a minimum of 7.2 Kbps to a maximum of 1.8432 Mbps. The serial clock generator uses the 3.6864-MHz clock produced by the on-chip PLL, divided by a fixed value of 2, and then the programmable SCR value to generate the serial clock (SCLK). The resultant clock rate is driven out on the SCLK pin and is also used by the SSP's transmit logic to drive data out on the TXD4 pin, and latch data on the RXD4 pin. Depending on the frame format selected, each transmitted bit is either driven on the rising or falling edge of SCLK, and is sampled on the opposite clock edge. The resultant serial clock rate, given a specific SCR value or required SCR value given a desired bit rate, can be calculated using the following two respective equations, where SCR is the decimal equivalent of the binary value programmed within the bit field:

$$BitRate = \frac{3.6864\times10^{6}}{2x(SCR+1)} \qquad SCR = \frac{3.6864\times10^{6}}{2xBitRate} - 1$$

The following table shows the bit locations corresponding to the five different control bit fields within SSP control register 0. Note that the SSE bit is the only control bit that is reset to a known state to ensure the SSP is disabled following a reset of the SA-1100. The reset state of all other control bits is unknown (indicated by question marks) and must be initialized before enabling the SSP. Reads of bit 6, which is reserved, return zero; writes have no effect.

Address: 0h 8007 0060 **SSP Control Register 0: SSCR0** **Read/Write**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SCR								SSE	Res.	FRF		DSS			
Reset	?	?	?	?	?	?	?	?	0	0	?	?	?	?	?	?

Bit	Name	Description
3..0	DSS	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.
5..4	FRF	Frame Format. 00 – Motorola SPI frame format. 01 – Texas Instruments Synchronous serial frame format. 10 – National Microwire frame format. 11 – Reserved, undefined operation.
6	—	Reserved.
7	SSE	Synchronous serial port enable. 0 – SSP operation disabled, control of pins given to PPC if MCP is also disabled. 1 – SSP operation enabled if MCP disabled or if the PPC SSP pin reassignment bit is set (reassigns GPIO<13..10> to the SSP).
15..8	SCR	Serial clock rate. Value (from 0 to 255) used to generate the transmission rate of the SSP. Bit Rate = $3.6864x10^6/(2x(SCR+1))$, where SCR is a decimal value.

11.12.10 SSP Control Register 1

The SSP control register 1 (SSCR1) contains six different bit fields that control various functions within the SSP.

11.12.10.1 Receive FIFO Interrupt Enable (RIE)

The receive FIFO interrupt enable (RIE) bit is used to mask or enable the receive FIFO service request interrupt. When RIE=0, the interrupt is masked and the state of the receive FIFO service request (RFS) bit within the SSP status register is ignored by the interrupt controller. When RIE=1, the interrupt is enabled, and whenever RFS is set (one), an interrupt request is made to the interrupt controller. Note that programming RIE=0 does not affect the current state of RFS or the receive FIFO logic's ability to set and clear RFS, it only blocks the generation of the interrupt request. Also note that RIE does not affect generation of the receive FIFO DMA request, which is asserted whenever RFS=1.

11.12.10.2 Transmit FIFO Interrupt Enable (TIE)

The transmit FIFO interrupt enable (TIE) bit is used to mask or enable the transmit FIFO service request interrupt. When TIE=0, the interrupt is masked and the state of the transmit FIFO service request (TFS) bit within the SSP status register is ignored by the interrupt controller. When TIE=1, the interrupt is enabled, and whenever TFS is set (one), an interrupt request is made to the interrupt controller. Note that programming TIE=0 does not affect the current state of TFS or the transmit FIFO logic's ability to set and clear TFS; it only blocks the generation of the interrupt request. Also note that TIE does not affect generation of the transmit FIFO DMA request, which is asserted whenever TFS=1.

11.12.10.3 Loopback Mode (LBM)

The loopback mode (LBM) bit is used to enable and disable the ability of the SSP transmit and receive logic to communicate. When LBM=0, the SSP operates normally. The transmit and receive data paths are independent and communicate via their respective pins. When LBM=1, the output of the transmit serial shifter is directly connected to the input of the receive serial shifter internally and control of the **TXD4**, **RXD4**, **SCLK**, and **SFRM** pins are given to the peripheral pin control (PPC) unit.

11.12.10.4 Serial Clock Polarity (SPO)

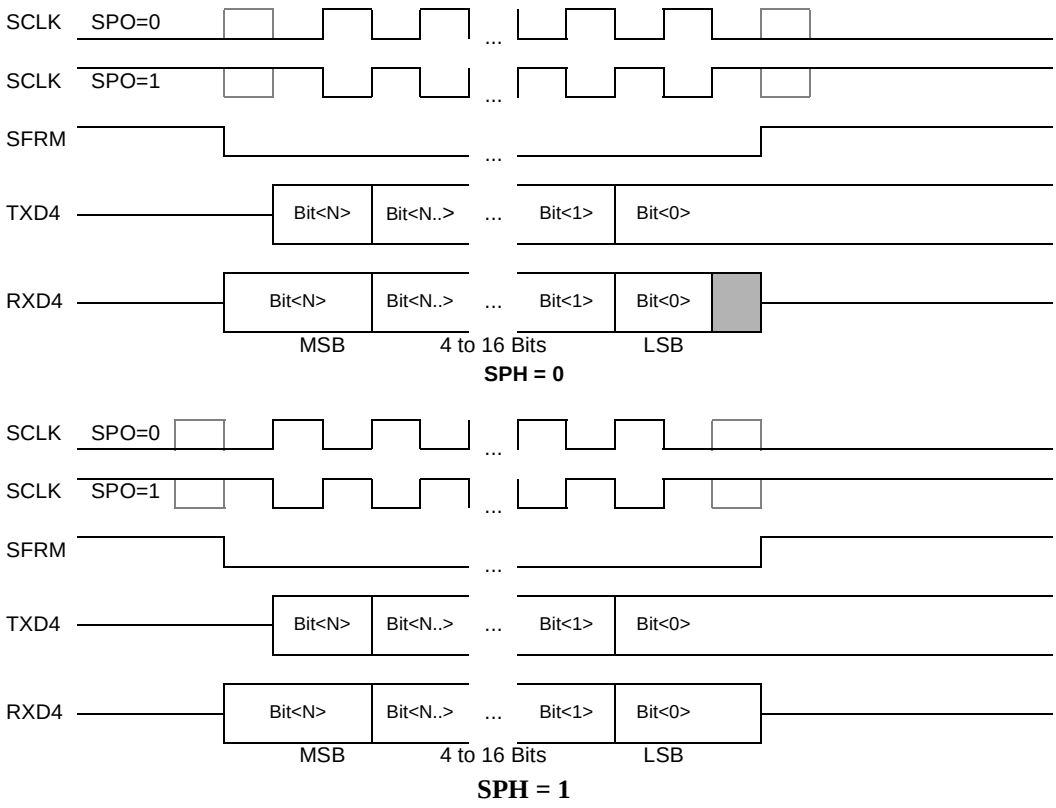
The serial clock polarity (SPO) bit selects the polarity or active/inactive state of the serial clock (SCLK) pin when Motorola* SPI format is selected (FRF=00). When SPO=0, the inactive or idle state of SCLK is low. Thus when the SSP is not actively transmitting/receiving data, the SCLK pin is held low. When SPO=1, the inactive or idle state of SCLK is high. Thus when the SSP is not actively transmitting/receiving data, the SCLK pin is held high. The programming of SPO alone does not determine which SCLK edges are used to drive and latch data to or from the transmit and receive pins. The programming of SPO and the serial clock phase (SPH) bit determines this. Note that SPO is ignored in all other modes except Motorola* SPI format (FRF=00).

11.12.10.5 Serial Clock Phase (SPH)

The serial clock phase (SPH) bit selects the phase relationship of the serial clock (SCLK) signal with the serial frame (SFRM) signal when Motorola* SPI format is selected (FRF=00). When SPH=0, SCLK remains in its inactive state (as programmed by SPO) for one full SCLK period duration after SFRM is asserted (driven low). SCLK continues to transition during the entire frame and is driven to its inactive state one-half SCLK period duration before SFRM is negated (driven high). When SPH=1, SCLK remains in its inactive state (as programmed by SPO) for one-half SCLK period duration after SFRM is asserted (driven low). SCLK continues to transition during the entire frame and is driven to its inactive state one full SCLK period duration before SFRM is negated (driven high). Using SPH and SPO together determine when SCLK is active during the assertion of SFRM and which edge of SCLK is used to drive data to the transmit pin as well as latch data from the receive pin. When SPO and SPH are the same value (both 0 or both 1), transmit data is driven on the falling edge of SCLK and receive data is latched on the rising edge of SCLK. Alternatively, when SPO and SPH are of opposite value (one 0 and the other 1), transmit data is driven on the rising edge of SCLK and receive data is latched on the falling edge of SCLK. Note that SPH is ignored in all other modes, except Motorola* SPI format (FRF=00).

Figure 11-39 shows the pin timing for all four programming combinations of SPO and SPH. Note that SPO inverts the polarity of the SCLK signal, and SPH determines the phase relationship between SCLK and SFRM, shifting the SCLK signal one-half phase to the left or right during the assertion of SFRM.

Figure 11-39. Motorola* SPI Frame Formats for SPO and SPH Programming



11.12.10.6 External Clock Select (ECS)

The external clock select (ECS) bit selects whether the on-chip 3.6864-MHz clock is used by the SSP or if an off-chip clock is supplied via GPIO pin 19. When ECS=0, the SSP uses the on-chip 3.6864-MHz clock to produce a range of serial transmission rates ranging from 7.2 Kbps to a maximum of 1.8432 Mbps. When ECS=1, the SSP uses GPIO<19> to input a clock supplied from off-chip. The frequency of the off-chip clock can be any value up to 3.6864 MHz. This off-chip clock is useful when a serial transmission rate, which is not an even multiple of 3.6864 MHz, is required for synchronization with the target off-chip slave device. When using GPIO pin 19 for the input clock, the user must also set bit 19 of the GPIO alternate function register (GAFR), and clear bit 19 of the GPIO pin direction register (GPDR). See the System Control Module chapter.

The following table shows the bit locations corresponding to the three different control bit fields within SSP control register 1. The reset state of all bits is unknown (indicated by question marks) and must be initialized before enabling the SSP. Note that writes to reserved bits are ignored and reads return zero.

Address: 0h 8007 0064 **SSP Control Register 1: SSCR1** **Read/Write**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved										ECS	SPH	SPO	LBM	TIE	RIE
Reset	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?

Bit	Name	Description
0	RIE	Receive FIFO interrupt enable. 0 – Receive FIFO one- to two-thirds full or more condition does not generate an interrupt (RFS bit ignored). 1 – Receive FIFO one- to two-thirds full or more condition generates an interrupt (state of RFS sent to interrupt controller).
1	TIE	Transmit FIFO interrupt enable. 0 – Transmit FIFO half-full or less condition does not generate an interrupt (TFS bit ignored). 1 – Transmit FIFO half-full or less condition generates an interrupt (state of TFS sent to interrupt controller).
2	LBM	Loopback mode. 0 – Normal serial port operation enabled. 1 – Output of transmit serial shifter is connected to input of receive serial shifter internally and control of TXD4, RXD4, SCLK, and SFRM pins is given to the PPC unit.
3	SPO	Serial clock polarity. 0 – The inactive or idle state of SCLK is low. 1 – The inactive or idle state of SCLK is high.
4	SP	Serial clock phase. 0 – SCLK is in its inactive state one full cycle at the start of the frame and one-half cycle at the end of the frame. 1 – SCLK is in its inactive state one-half cycle at the start of the frame and one full cycle at the end of the frame.
5	ECS	External clock select. 0 – on-chip clock used to product the SSP's serial clock and control all timing. 1 – Clock input using GPIO pin 19 to drive the serial clock and all timing when serial rates that are not a multiple of 3.6864 MHz are needed. Note that bit 19 within GFAR and GPDR must be correctly configured within the system control module.
15..6	—	Reserved.

11.12.11 SSP Data Register

The SSP data register (SSDR) is 16 bits wide and corresponds to the top and bottom entries of the transmit and receive FIFOs, respectively. When SSDR is read, the bottom entry of receive FIFO is accessed. As data is removed by the SSP's receive logic from the incoming data frame, it is placed into the top entry of the receive FIFO and is transferred down an entry at a time until it reaches the last empty location within the FIFO. Data is removed by reading SSDR, which accesses the bottom entry of the FIFO. After SSDR is read, the bottom entry is invalidated, and all remaining values within the FIFO automatically transfer down one location.

When SSDR is written, the topmost entry of the transmit FIFO is accessed. After a write, data is automatically transferred down to the lowest location within the transmit FIFO, which does not already contain valid data. Data is removed from the bottom of the FIFO one value at a time by the transmit logic, is loaded into the transmit serial shifter, and then is serially shifted onto the TXD4 pin at the programmed bit rate.

When a data size of less than 16 bits is selected, the user should left justify data written to the transmit FIFO. The transmit logic ignores the upper unused bits. Received data less than 16 bits is automatically right justified in the receive buffer and unused bits are zero filled. When the SSP is programmed for National Microwire* frame format, the default size for transmit data is 8 bits (the most significant byte is ignored) and the receive data size is controlled by the programmer.

The following table shows the location of the SSP data register. Note that both FIFOs are cleared when the SA-1100 is reset or by writing a zero to SSE (SSP disabled).

Address: 0h 8007 006C **SSP Data Register: SSDR** **Read/Write**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Bottom of Receive FIFO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Read Access

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Top of Transmit FIFO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Write Access

Bit	Name	Description
15..0	Data	Top/bottom of transmit/receive FIFO. Read – Bottom of receive FIFO. Write – Top of transmit FIFO. **Note:** User should left justify data when SSP programmed for a data size less than 16 bits. Top unused bits are ignored by transmit logic. Receive logic automatically right justifies data and zero fills unused bits.

11.12.12 SSP Status Register

The SSP status register (SSSR) contains bits that signal overrun errors as well as the transmit and receive FIFO service requests. Each of these hardware-detected events signals an interrupt request to the interrupt controller. The status register also contains flags that indicate when the SSP is actively transmitting characters, when the transmit FIFO is not full, and when the receive FIFO is not empty (no interrupt generated).

A bit that can cause an interrupt signals the interrupt request as long as the bit is set. Once the bit is cleared, the interrupt is cleared. Read/write bits are called status bits; read-only bits are called flags. Status bits are referred to as “sticky” (once set by hardware, must be cleared by software). Writing a one to a sticky status bit clears it; writing a zero has no effect. Read-only flags are set and cleared by hardware; writes have no effect. Additionally, some bits that cause interrupts have corresponding mask/enable bits in the control registers and are indicated in the following section headings. Note that the user has the ability to mask all SSP interrupts by clearing bit 19 within the interrupt controller mask register (ICMR). See the Section 9.2, “Interrupt Controller” on page 9-11.

11.12.12.1 Transmit FIFO Not Full Flag (TNF) (read-only, noninterruptible)

The transmit FIFO not full flag (TNF) is a read-only bit that is set whenever the transmit FIFO contains one or more entries that do not contain valid data and is cleared when the FIFO is completely full. This bit can be polled when using programmed I/O to fill the transmit FIFO over its halfway mark. This bit does not request an interrupt.

11.12.12.2 Receive FIFO Not Empty Flag (RNE) (read-only, noninterruptible)

The receive FIFO not empty flag (RNE) is a read-only bit that is set whenever the receive FIFO contains one or more entries of valid data and is cleared when it no longer contains any valid data. This bit can be polled when using programmed I/O to remove remaining bytes of data from the receive FIFO because DMA service and CPU interrupt requests are only made when four or more bytes reside within the FIFO (3, 2, or 1 bytes may remain at the end of a frame). This bit does not request an interrupt.

11.12.12.3 SSP Busy Flag (BSY) (read-only, noninterruptible)

The SSP busy (BSY) flag is a read-only bit that is set when the SSP is actively transmitting and/or receiving data, and is cleared when the SSP is idle or disabled (SSE=0). This bit does not request an interrupt.

11.12.12.4 Transmit FIFO Service Request Flag (TFS) (read-only, maskable interrupt)

The transmit FIFO service request flag (TFS) is a read-only bit that is set when the transmit FIFO is nearly empty and requires service to prevent an underrun. TFS is set whenever the transmit FIFO has four or fewer entries of valid data (half-full or less), and is cleared when it has five or more entries of valid data. When the TFS bit is set, an interrupt request is made unless the transmit FIFO interrupt request enable (TIE) bit is cleared. The state of TFS is also sent to the DMA controller, and can be used to signal a DMA service request. Note that TIE has no effect on the generation of the DMA service request. After the DMA or CPU fills the FIFO such that four or more locations are filled within the transmit FIFO, the TFS flag (and the service request and/or interrupt) is automatically cleared.

11.12.12.5 Receive FIFO Service Request Flag (RFS) (read-only, maskable interrupt)

The receive FIFO service request flag (RFS) is a read-only bit that is set when the receive FIFO is nearly filled and requires service to prevent an overrun. RFS is set whenever the receive FIFO has four or more entries of valid data (half-full or more), and is cleared when it has three or fewer (less than half-full) entries of data. When the RFS bit is set, an interrupt request is made unless the receive FIFO interrupt request enable (RIE) bit is cleared. The state of RFS is also sent to the DMA controller, and can be used to signal a DMA service request. Note that RIE has no effect on the generation of the DMA service request. After the DMA or CPU fills the FIFO such that four or more locations are filled within the receive FIFO, the RFS flag (and the service request and/or interrupt) is automatically cleared.

11.12.12.6 Receiver Overrun Status (ROR) (read/write, nonmaskable interrupt)

The receiver overrun status bit (ROR) is a read/write bit that is set when the receive logic attempts to place data into the receive FIFO after it has been completely filled. Each time a new piece of data is received, the set signal to the ROR bit is asserted, and the newly received data is discarded. This process is repeated for each new piece of data received until at least one empty FIFO entry exists. When the ROR bit is set, an interrupt request is made.

The following table shows the bit locations corresponding to the status and flag bits within the SSP status register. All bits are read-only except ROR, which is read/write. Writes to TNF, RNE, BSY, TFS, and RFS have no effect. The reset state of ROR is unknown (indicated by a question mark) and must be initialized before enabling the SSP. Note that writes to reserved bits are ignored and reads return zeros.

Address: 0h 8007 0074 — **SSP Status Register: SSSR** — **Read/Write & Read-Only**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved									ROR	RFS	TFS	BSY	RNE	TNF	Res
Reset	0	0	0	0	0	0	0	0	0	?	0	0	0	0	1	0

Bit	Name	Description
0	—	Reserved.
1	TNF	Transmit FIFO not full (read-only). 0 – Transmit FIFO is full. 1 – Transmit FIFO is not full.
2	RNE	Receive FIFO not empty (read-only). 0 – Receive FIFO is empty. 1 – Receive FIFO is not empty.
3	BSY	SSP busy flag (read-only). 0 – SSP is idle or disabled. 1 – SSP is currently transmitting and/or receiving a frame (no interrupt generated).
4	TFS	Transmit FIFO service request (read-only). 0 – Transmit FIFO is more than half-full (five or more entries filled) or SSP disabled. 1 – Transmit FIFO is half-full or less (four or fewer entries filled) and SSP operation is enabled, DMA service request signalled, interrupt request signalled if not masked (if TIE=1).
5	RFS	Receive FIFO service request (read-only). 0 – Receive FIFO is less than half-full (three or fewer entries filled) or SSP disabled. 1 – Receive FIFO is half-full or more (four or more entries filled) and SSP operation is enabled, DMA service request signalled, interrupt request signalled if not masked (if RIE=1).
6	ROR	Receive FIFO overrun. 0 – Receive FIFO has not experienced an overrun. 1 – Receive logic attempted to place data into receive FIFO while it was full, request interrupt.
15..7	—	Reserved.

11.12.13 MCP Register Locations

Table 11-19 shows the registers associated with the MCP and the physical addresses used to access them.

Table 11-19. MCP Control, Data, and Status Register Locations

Address	Name	Description
0h 8006 0000	MCCR0	MCP control register 0
0h 8006 0004	—	Reserved
0h 8006 0008	MCDR0	MCP data register 0
0h 8006 000C	MCDR1	MCP data register 1
0h 8006 0010	MCDR2	MCP data register 2
0h 8006 0014	—	Reserved
0h 8006 0018	MCSR	MCP status register
0h 8006 001C – 0h 8006 005C	—	Reserved

Note: MCCR1 resides within the same address space as the PPC.

0h 9006 0030	MCCR1	MCP control register 1

11.12.14 SSP Register Locations

Table 11-20 shows the registers associated with the SSP and the physical addresses used to access them.

Table 11-20. SSP Control, Data, and Status Register Locations

Address	Name	Description
0h 8007 0060	SSCR0	SSP control register 0
0h 8007 0064	SSCR1	SSP control register 1
0h 8007 0068	—	Reserved
0h 8007 006C	SSDR	SSP data register
0h 8007 0070	—	Reserved
0h 8007 0074	SSSR	SSP status register
0h 8007 0078 – 0h 8007 FFFF	—	Reserved

11.13 Peripheral Pin Controller (PPC)

The peripheral pin controller (PPC) takes individual control of the LCD's and serial port 1..4's pins when one or more of the units are disabled, allowing the user to utilize them as general-purpose digital I/O pins to communicate to off-chip resources. When controlled by the PPC, peripheral control module (PCM) pins operate similarly to GPIO pins except that they cannot perform edge detection and interrupt generation. The PPC is also used to specify the direction of the peripherals' pins when sleep mode is entered.

Note that serial ports 1..3 contain individual enables for their transmit and receive serial engines. Thus, if only half-duplex transmission is needed, one pin can be used for serial communication and the other for digital I/O communication. Also note that serial port 0's pins are dedicated to the USB device controller (UDC), which uses the pins to drive a differential transceiver, preventing them from being used as digital I/O pins when the UDC is disabled.

11.13.1 PPC Operation

Following a hardware reset of the SA-1100 (nRESET asserted then negated), all peripheral control module units are disabled, giving control of their pins to the PPC (except serial port 0). The PPC, in turn, configures all peripheral pins it controls as inputs. Once reset is negated, the user should program the peripherals as soon as possible, and configure the pins of any peripheral that is not usable to function as general-purpose I/O signals. This should be done quickly to limit the amount of power consumed at startup because pins that are intended to function as outputs within the system are initially configured as inputs, and the receiving device to which they are connected will float and consume power.

The PPC contains special resources to limit off-chip power consumption during and immediately following the assertion of sleep mode. The PPC contains a sleep mode direction register, which is programmed by the user, and individually configures 22 of the peripherals' pins either as inputs or outputs during sleep mode. When configured as an output, the pin is forced low in sleep mode. This special register is required because the first action taken when sleep mode is entered is the assertion of reset to all the peripherals, which would, in turn, errantly configure all peripheral pins as inputs. The sleep mode direction register is not reset; the user can maintain the correct direction programmed for each of the peripherals' pins while in sleep mode. When sleep mode is exited, the user can then reprogram the peripherals and the PPC registers to resume control of the peripherals' pins. To keep the same pin direction and state after sleep mode has been negated but before the user reprograms the peripherals, the system control module's power manager maintains the peripherals' pin direction and state following sleep negation until the release peripheral pin (RPP) bit, located in the power manager, is set by the user. Therefore, the pin direction and state established during sleep using the sleep mode direction register remains intact following the negation of sleep until the RPP bit is set. Once RPP is set, control of the peripherals' pins is given back to the individual peripherals and to the PPC unit.

Most of the SA-1100's peripherals can take control of one or more GPIO pins (which are normally controlled within the system control module) to act as input or output triggers, or to drive or supply clocks to the peripherals. The GPIO unit contains a GPIO alternate function register (GAFR) that the user must program to give control of the GPIO pins to the individual peripheral units for each of the alternate functions. The user must also program the GPIO pin direction register (GPDR) for the corresponding pins that are used by the peripheral units. The GPIO pin alternate functions are then enabled within the individual peripherals using a control bit. However, two control bits exist within the PPC that configure six of the GPIO unit's pins for peripheral alternate functions.

Serial port 1 and serial port 4 both contain two serial-to-parallel engines that operate independently. However, because each port contains only one set of serial pins, the user can assign these pins to only one of the two protocols at a time. To allow the user to utilize both protocols, the PPC can assign one of its two serial-to-parallel engines to the pins that are dedicated to the port, and the other to a set of GPIO pins. Serial port 1 contains an SDLC and a UART. By setting a bit in the PPC and the appropriate GAFR and GPDR bits in the GPIO unit, SDLC operation defaults to the TXD1 and RXD1 pins, and the UART transmits via the GPIO<14> pin and receives via the GPIO<15> pin. Likewise, serial port 4 contains the MCP and the SSP synchronous serial engines. The user can configure the PPC and GPIO units to cause the MCP to default to the TXD4, RXD4, SCLK, and SFRM pins, and the SSP is assigned to GPIO<10> for transmit, GPIO<11> for receive, GPIO<12> for serial clock, and GPIO<13> for serial frame.

When the SA-1100 is reset or enters sleep mode, the GPIO unit's registers are reset, which gives control of the GPIO pins back to the system control module.

11.13.2 PPC Register Definitions

There are five registers within the PPC: one pin direction register, one pin state register, one pin assignment register, one sleep mode pin direction register, and one pin flag register.

11.13.3 PPC Pin Direction Register

Pin direction is controlled by programming the PPC pin direction register (PPDR). The PPDR contains individual direction control bits for 22 of the 24 peripheral pins. Serial port 0 has dedicated pins (UDC+ and UDC-) that are not controlled by the PPC when the UDC is disabled. Each bit is used only if the corresponding peripheral that it controls is disabled. Provided the corresponding peripheral is disabled, if the direction bit is programmed to a one, the pin is an output. If it is programmed to a zero, it is an input. Following reset, all peripherals are disabled, which causes the PPC to take control of all of their pins. Serial ports 1..3 contain individual enables for their transmit and receive serial engines. Thus, if only half-duplex transmission is needed, one pin can be used for serial communication and the other for digital I/O communication. Note that PPDR is reset such that all the pins are configured as inputs. For reserved bits, writes are ignored and reads return zero. The following table shows the location of each pin direction bit and to which peripheral pin it corresponds.

Address: 0h 9006 0000 — **PPDR: PPC Pin Direction Register** — **Read/Write**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved										SFRM	SCLK	RXD4	TXD4	RXD3	TXD3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RXD2	TXD2	RXD1	TXD1	L_ BIAS	L_ FCLK	L_ LCLK	L_ PCLK	LDD <7>	LDD <6>	LDD <5>	LDD <4>	LDD <3>	LDD <2>	LDD <1>	LDD <0>
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
7..0	LDD<7:0>	LCD data pin direction. 0 – If LCD controller disabled, LCD data pin configured as general-purpose input. 1 – If LCD controller disabled, LCD data pin configured as general-purpose output.
8	L_PCLK	LCD pixel clock pin direction. 0 – If LCD controller disabled, LCD pixel clock pin configured as general-purpose input. 1 – If LCD controller disabled, LCD pixel clock pin configured as general-purpose output.
9	L_LCLK	LCD line clock pin direction. 0 – If LCD controller disabled, LCD line clock pin configured as general-purpose input. 1 – If LCD controller disabled, LCD line clock pin configured as general-purpose output.
10	L_FCLK	LCD frame clock pin direction. 0 – If LCD controller disabled, LCD frame clock pin configured as general-purpose input. 1 – If LCD controller disabled, LCD frame clock pin configured as general-purpose output.
11	L_BIAS	LCD AC bias pin direction. 0 – If LCD controller disabled, LCD ac bias pin configured as general-purpose input. 1 – If LCD controller disabled, LCD ac bias pin configured as general-purpose output.
12	TXD1	Serial port 1: SDLC/UART transmit pin direction. 0 – If serial port 1 transmitter disabled, transmit pin configured as general-purpose input. 1 – If serial port 1 transmitter disabled, transmit pin configured as general-purpose output.
13	RXD1	Serial port 1: SDLC/UART receive pin direction. 0 – If serial port 1 receiver disabled, receive pin configured as general-purpose input. 1 – If serial port 1 receiver disabled, receive pin configured as general-purpose output.
14	TXD2	Serial port 2: IPC transmit pin direction. 0 – If serial port 2 transmitter disabled, transmit pin configured as general-purpose input. 1 – If serial port 2 transmitter disabled, transmit pin configured as general-purpose output.
15	RXD2	Serial port 2: IPC receive pin direction. 0 – If serial port 2 receiver disabled, receive pin configured as general-purpose input 1 – If serial port 2 receiver disabled, receive pin configured as general-purpose output
16	TXD3	Serial port 3: UART transmit pin direction. 0 – If serial port 3 transmitter disabled, transmit pin configured as general-purpose input 1 – If serial port 3 transmitter disabled, transmit pin configured as general-purpose output
17	RXD3	Serial port 3: UART receive pin direction. 0 - If serial port 3 receiver disabled, receive pin configured as general-purpose input. 1 - If serial port 3 receiver disabled, receive pin configured as general-purpose output.
18	TXD4	Serial port 4: MCP/SSP transmit pin direction. 0 - If serial port 4 disabled, transmit pin configured as general-purpose input. 1 - If serial port 4 disabled, transmit pin configured as general-purpose output.
19	RXD4	Serial port 4: MPC/SSP receive pin direction. 0 – If serial port 4 disabled, receive pin configured as general-purpose input. 1 – If serial port 4 disabled, receive pin configured as general-purpose output.
20	SCLK	Serial port 4: MPC/SSP serial clock pin direction. 0 – If serial port 4 disabled, serial clock pin configured as general-purpose input. 1 – If serial port 4 disabled, serial clock pin configured as general-purpose output.
21	SFRM	Serial port 4: MPC/SSP serial frame pin direction. 0 – If serial port 4 disabled, serial frame pin configured as general-purpose input. 1 – If serial port 4 disabled, serial frame pin configured as general-purpose output.
31..22	—	Reserved.

11.13.4 PPC Pin State Register

Pin state is both monitored and controlled by reading/writing the PPC pin state register (PPSR). The PPSR contains 1 state bit for each of the 22 peripheral pins. This register may be read at any time to determine the current state of all peripheral pins, even when pins are controlled by the peripheral rather than the PPC. If a peripheral is disabled and its corresponding pin direction is programmed as an output in the PPDR, its PPSR bit is used to control the state of the peripheral pin. Writing a zero to the pin's state bit causes the pin to be forced low, and writing a one causes the pin to be forced high. Writing a value to a pin state bit that is an input or is not under the control of the PPC has no effect. To alter the state of an output pin, the user should first read the PPSR, then logically AND the value read with a mask, which contains ones in every bit position except the one the user wishes to clear. To set a pin, the user should logically OR the value read with a mask, which contains zeros in every bit position except the one the user wishes to set. This mechanism allows the user to set or clear individual pins without changing the state of other pins that are configured as outputs.

Serial port 2 contains two bits that control the polarity of data input via the receive pin (RXD2) and data output via the transmit pin (TXD2). The user must ensure that these polarity bits are set (RXP = TXP = 1), which selects true or noninverted data before using TXD2 or RXD2 as GPIO pins.

Note that PPSR is implemented as two separate registers. A write to PPSR addresses one of the registers and is used to set and clear pins configured as GPIO outputs, while a read addresses the other register that is used to store and monitor pin state. The register used to store pin state contains logic to synchronize the signal input from the pin to allow the user to read it. The pins are sampled at a rate of 7.3728 MHz; each synchronization cycle takes 135.6 ns. Depending on the CPU frequency programmed by the user, after changing the state of an output pin via a write, one or more dummy read cycle waitstates may need to be inserted to allow the value to be output to the pin and to allow the synchronizer to resample the pin.

The following table shows the location of each pin state bit and to which peripheral pin it corresponds. Note that this register is not reset and that for reserved bits, writes are ignored and reads return zero.

Address: 0h 9006 0000 **PPDR: PPC Pin Direction Register** **Read/Write**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved										SFRM	SCLK	RXD4	TXD4	RXD3	TXD3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RXD2	TXD2	RXD1	TXD1	L_ BIAS	L_ FCK	L_ LCK	L_ PCK	LDD <7>	LDD <6>	LDD <5>	LDD <4>	LDD <3>	LDD <2>	LDD <1>	LDD <0>
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
7..0	LDD<7:0>	LCD data pin state. Read – Current state of LCD data pin returned. Write – If LCD disabled and pin configured as an output, drive value to LCD data pin.
8	L_PCLK	LCD pixel clock pin state. Read – Current state of LCD pixel clock pin returned. Write – If LCD disabled and pin configured as an output, drive value to LCD pixel clock pin.
9	L_LCLK	LCD line clock pin state. Read – Current state of LCD line clock pin returned. Write – If LCD disabled and pin configured as an output, drive value to LCD line clock pin.
10	L_FCLK	LCD frame clock pin state. Read – Current state of LCD frame clock pin returned. Write – If LCD disabled and pin configured as an output, drive value to LCD frame clock pin.
11	L_BIAS	LCD AC bias pin state. Read – Current state of LCD AC bias pin returned. Write – If LCD disabled and pin configured as an output, drive value to LCD AC bias pin.
12	TXD1	Serial port 1: SDLC/UART transmit pin state. Read – Current state of serial port 1 transmit pin returned. Write – If serial port 1 transmitter disabled and pin configured as an output, drive value to transmit pin.
13	RXD1	Serial port 1: SDLC/UART receive pin state. Read – Current state of serial port 1 receive pin returned. Write – If serial port 1 receiver disabled and pin configured as an output, drive value to receive pin.
14	TXD2	Serial port 2: IPC transmit pin state. Read – Current state of serial port 1 transmit pin returned. Write – If serial port 2 transmitter disabled and pin configured as an output, drive value to transmit pin.
15	RXD2	Serial port 2: IPC receive pin state. Read – Current state of serial port 2 receive pin returned. Write – If serial port 2 receiver disabled and pin configured as an output, drive value to receive pin.
16	TXD3	Serial port 3: UART transmit pin state. Read – Current state of serial port 3 transmit pin returned. Write – If serial port 3 transmitter disabled and pin configured as an output, drive value to transmit pin.
17	RXD3	Serial port 3: UART receive pin state. Read – Current state of serial port 3 receive pin returned. Write – If serial port 3 receive disabled and pin configured as an output, drive value to receive pin.
18	TXD4	Serial port 4: MCP/SSP transmit pin state. Read – Current state of serial port 4 transmit pin returned. Write – If serial port 4 transmitter disabled and pin configured as an output, drive value to transmit pin.
19	RXD4	Serial port 4: MCP/SSP receive pin state. Read – Current state of serial port 4 receive pin returned. Write – If serial port 4 receive disabled and pin configured as an output, drive value to receive pin.
20	SCLK	Serial port 4: MCP/SSP serial clock pin state. Read – Current state of serial port 4 serial clock pin returned. Write – If serial port 4 disabled and pin configured as an output, drive value to serial clock pin.
21	SFRM	Serial port 4: MCP/SSP serial frame pin state. Read – Current state of serial port 4 serial frame pin returned. Write – If serial port 4 disabled and pin configured as an output, drive value to serial frame pin.
31..22	—	Reserved.

11.13.5 PPC Pin Assignment Register

The UART in serial port 1 and the SSP in serial port 4 can be reassigned to GPIO pins using the PPC pin assignment register (PPAR). The PPAR contains two bits that control the reassignment of each serial engine to an individual set of GPIO pins.

11.13.5.1 UART Pin Reassignment (UPR)

The UART pin reassignment (UPR) bit is used to select whether serial port 1's UART is assigned to GPIO pins 14 and 15. When UPR=0, serial port 1 uses its TXD1 and RXD1 pins, and the SDLC/UART select (SUS) bit is used to select which protocol is enabled. When UPR=1, SUS is ignored, serial port 1 defaults to SDLC operation using the TXD1 and RXD1 pins, and the UART is configured to use GPIO<14> for transmit and GPIO<15> for receive. Note that the user must set bits 14 and 15 in the GPIO alternate function register (GAFR) as well as set bit 14 and clear bit 15 in the GPIO pin direction register (GPDR). See the Section 9.1, "General-Purpose I/O" on page 9-1.

11.13.5.2 SSP Pin Reassignment (SPR)

The SSP pin reassignment (SPR) bit is used to select whether serial port 4's SSP is assigned to GPIO pins 10 through 13. When SPR=0, serial port 4 uses its TXD4, RXD4, SCLK, and SFRM pins; the MCP enable (MCE) and SSP enable (SSE) bits are used to select which protocol is enabled (MCE has precedence over SSE). When SPR=1, MCE and SSE must both be set; serial port 4 defaults to MCP operation using the TXD4, RXD4, SCLK, and SFRM pins, and the SSP is configured to use GPIO<10> for transmit, GPIO<11> for receive, GPIO<12> for serial clock, and GPIO<13> for serial frame. Note that the user must set bits 10 through 13 in the GPIO alternate function register (GAFR) as well as set bits 10, 12, and 13 and clear bit 11 in the GPIO pin direction register (GPDR). See the Section 9.1, "General-Purpose I/O" on page 9-1.

The following table shows the location of the two pin reassignment bits. Note that for reserved bits, writes are ignored and reads return zero. Both control bits are cleared to zero following a reset of the SA-1100, giving control of all GPIO pins to the system control module.

Address: 0h 9006 0008 — **PPAR: PPC Pin Assignment Register** — **Read/Write**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved													SPR	Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit	Name	Description
11..0	—	Reserved.
12	UPR	UART pin reassignment. 0 – No pin reassignment made, GPIO<14-15> controlled by GPIO unit, serial port 1 UART assigned to TXD1 and RXD1 if SUS=1. 1 – Pin reassignment made, serial port 1 defaults to SDLC operation (SUS ignored), UART transmit assigned to GPIO<14> and receive to GPIO<15>, GAFR and GPDR must be configured in GPIO unit.
17..13	—	Reserved.
18	SPR	SSP pin reassignment. 0 – No pin reassignment made, GPIO<10-13> controlled by GPIO unit, serial port 4 SSP assigned to TXD4, RXD4, SCLK, and SFRM if MCE=0 and SSE=1. 1– Pin reassignment made, serial port 4 defaults to MCP operation, SSP transmit assigned to GPIO<10>, receive to GPIO<11>, serial clock to GPIO<12>, and serial frame to GPIO<13>, GAFR and GPDR must be configured in GPIO unit.
31..19	—	Reserved.

11.13.6 PPC Sleep Mode Pin Direction Register

When sleep mode is entered, reset is asserted to all of the SA-1100's peripherals and to the PPC unit. The PPC pin direction register is cleared during a hard, soft, or sleep reset, causing the peripheral pins under the PPC's control to be configured as inputs. If this register were also used to determine pin direction during sleep, the pins would all be configured as inputs. This action would cause any off-chip device that expects data to be output from the SA-1100 to burn power during sleep because its input would float. The sleep mode pin direction register (PSDR) prevents this undesired power consumption by allowing the user to establish peripheral pin direction during and immediately following sleep mode.

When sleep mode is entered, both the peripherals and the PPC are reset; however, PSDR is not reset like PPDR. Once the user programs PSDR, it retains its data after sleep mode is entered and reset is asserted. The power manager uses the values in PSDR to determine the direction and state of the 22 peripheral pins. When a sleep mode direction bit is programmed to a zero, the corresponding pin is configured as an output and is driven low (zero). If it is programmed to a one, it is an input. The power manager latches the contents of PSDR before VDD is removed from the SA-1100 to maintain the peripheral pin direction and state after the main power supply is removed. Once VDD is removed, the data in PSDR is lost and must be reprogrammed after exiting sleep mode. The power manager contains a control bit called the release peripheral pin (RPP) bit. After exiting sleep mode, if this bit is zero, the power manager maintains the peripheral pin direction and state until the bit is set. Following sleep, the user should first reprogram the peripherals and the PPC, then set RPP in order to give control of the pins back to the peripheral units. Note that sleep mode invocation causes RPP to be cleared so that the pins are once again held in their sleep state until the user can set RPP. See Chapter 9, "System Control Module".

Because the peripherals are reset when sleep mode is entered, serial port 2's transmit and receive pin (TXD2 and RXD2) polarity bits (TXP and RXP) are both reset to one, which configures transmit and receive data as true or noninverted data. Thus the user need not reprogram these bits prior to the invocation of sleep mode.

Note that PPSR is initialized only by a hardware or power-on reset (negation of the nRESET pin). It is not affected by a software reset or a reset that occurs as a result of the SA-1100 entering sleep mode. Also note that for reserved bits, writes are ignored and reads return zero. The following table shows the location of each sleep mode pin direction bit and to which peripheral pin it corresponds.

Address: 0h 9006 000C **PPSR: PPC Sleep Mode Pin Direction Register** **Read/Write**

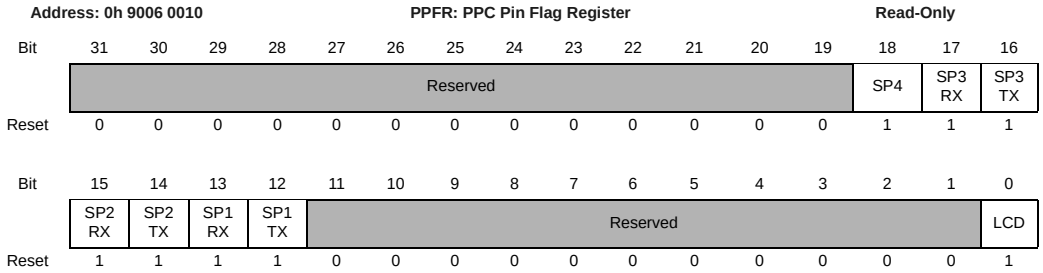
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved										SFRM	SCLK	RXD4	TXD4	RXD3	TXD3
Hard Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RXD 2	TXD2	RXD1	TXD1	L_ BIAS	L_ FCLK	L_ LCLK	L_ PCLK	LDD <7>	LDD <6>	LDD <5>	LDD <4>	LDD <3>	LDD <2>	LDD <1>	LDD <0>
Hard Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
7..0	LDD<7:0>	LCD data sleep mode pin direction. 0 – LCD data pin configured as output and is driven low during sleep. 1 – LCD data pin configured as input during sleep.
8	L_PCLK	LCD pixel clock sleep mode pin direction. 0 – LCD pixel clock pin configured as output and is driven low during sleep. 1 – LCD pixel clock pin configured as input during sleep.
9	L_LCLK	LCD line clock sleep mode pin direction. 0 – LCD line clock pin configured as output and is driven low during sleep. 1 – LCD line clock pin configured as input during sleep.
10	L_FCLK	LCD frame clock sleep mode pin direction. 0 – LCD frame clock pin configured as output and is driven low during sleep. 1 – LCD frame clock pin configured as input during sleep.
11	L_BIAS	LCD ac bias sleep mode pin direction. 0 – LCD ac bias pin configured as output and is driven low during sleep. 1 – LCD ac bias pin configured as input during sleep.
12	TXD1	Serial port 1: SDLC/UART transmit sleep mode pin direction. 0 – Transmit pin configured as output and is driven low during sleep. 1 – Transmit pin configured as input during sleep.
13	RXD1	Serial port 1: SDLC/UART receive sleep mode pin direction. 0 – Receive pin configured as output and is driven low during sleep. 1 – Receive pin configured as input during sleep.
14	TXD2	Serial port 2: IPC transmit sleep mode pin direction. 0 – Transmit pin configured as output and is driven low during sleep. 1 – Transmit pin configured as input during sleep.
15	RXD2	Serial port 2: IPC receive sleep mode pin direction. 0 – Receive pin configured as output and is driven low during sleep. 1 – Receive pin configured as input during sleep.
16	TXD3	Serial port 3: UART transmit sleep mode pin direction. 0 – Transmit pin configured as output and is driven low during sleep. 1 – Transmit pin configured as input during sleep.
17	RXD3	Serial port 3: UART receive sleep mode pin direction. 0 – Receive pin configured as output and is driven low during sleep. 1 – Receive pin configured as input during sleep.
18	TXD4	Serial port 4: MCP/SSP transmit sleep mode pin direction. 0 – Transmit pin configured as output and is driven low during sleep. 1 – Transmit pin configured as input during sleep.
19	RXD4	Serial port 4: MCP/SSP receive sleep mode pin direction. 0 – Receive pin configured as output and is driven low during sleep. 1 – Receive pin configured as input during sleep.
20	SCLK	Serial port 4: MCP/SSP serial clock sleep mode pin direction. 0 – Serial clock pin configured as output and is driven low during sleep. 1 – Serial clock pin configured as input during sleep.
21	SFRM	Serial port 4: MCP/SSP serial frame sleep mode pin direction. 0 – Serial frame pin configured as output and is driven low during sleep. 1 – Serial frame pin configured as input during sleep.
31..22	—	Reserved.

11.13.7 PPC Pin Flag Register

The PPC pin flag register (PPFR) is used to determine which peripherals are currently under the control of the PPC unit. The eight read-only flags denote whether or not each of the peripherals (except serial port 0) is enabled or is disabled and being controlled by the PPC. Note that serial ports 1..3 contain individual enables for their transmit and receive serial engines. Thus, separate flag bits exist for their transmit and receive pins. When a flag is set, it indicates that the corresponding peripheral is disabled and is controlled by the PPC; when it is cleared, it indicates that the peripheral is enabled and its pins are being used for serial transmission (serial ports 1..4) or for LCD operation. Note that for reserved bits, writes are ignored and reads return zero. The following table shows the location of each pin flag bit and to which peripheral pin it corresponds.



Bit	Name	Description
0	LCD	LCD controller flag (read-only). 0 – LCD controller enabled. 1 – LCD disabled, PPC currently controlling all 12 of its pins: LDD<7:0>, L_PCLK, L_LCLK, L_FCLK, L_BIAS.
11..1	—	Reserved.
12	SP1 TX	Serial port 1: SDLC/UART transmit flag (read-only). 0 – SDLC or UART transmit enabled. 1 – SDLC and UART transmitters disabled, PPC currently controlling the transmit pin: TXD1.
13	SP1 RX	Serial port 1: SDLC/UART receive flag (read-only). 0 – SDLC or UART receive enabled. 1 – SDLC and UART receivers disabled, PPC currently controlling the receive pin: RXD1.
14	SP2 TX	Serial port 2: ICP transmit flag (read-only). 0 – HSSP or UART transmit enabled. 1– HSSP and UART transmitters disabled, PPC currently controlling the transmit pin: TXD2.
15	SP2 RX	Serial port 2: ICP receive flag (read-only). 0 – HSSP or UART receive enabled. 1 – HSSP and UART receivers disabled, PPC currently controlling the receive pin: RXD2.
16	SP3 TX	Serial port 3: UART transmit flag (read-only). 0 – UART transmit enabled. 1 – UART transmit disabled, PPC currently controlling the transmit pin: TXD3.
17	SP3 RX	Serial port 3: UART receive flag (read-only). 0 – UART receive enabled. 1 – UART receive disabled, PPC currently controlling the receive pin: RXD3.
18	SP4	Serial port 4: MCP/SSP flag (read-only). 0 – MCP or SSP enabled. 1– MCP and SSP disabled, PPC currently controlling all 4 of its pins: TXD4, RXD4, SCLK, SFRM.
31..19	—	Reserved.

11.13.8 PPC Register Locations

Table 11-21 shows the registers associated with the PPC and the physical addresses used to access them. Note that serial port 2 (ICP) has implemented HSSP control register 2 and serial port 4 (MCP) has also implemented MCP control register 1 within the PPC's address space at 0h 9006 0028 and 0h 9006 0030 respectively. The user should ensure that these registers are not accidentally written by any PPC routines that may attempt to write to all of the PPC's address space, including its reserved registers during initialization.

Table 11-21. PPC Control and Flag Register Locations

Address	Name	Description
0h 9006 0000	PPDR	PPC pin direction register
0h 9006 0004	PPSR	PPC pin state register
0h 9006 0008	PPAR	PPC pin assignment register
0h 9006 000C	PSDR	PPC sleep mode direction register
0h 9006 0010	PPFR	PPC pin flag register
0h 9006 0014 – 0h 9006 FFFF	—	Reserved

DC Parameters 12

This chapter defines the dc parameters for the Intel® StrongARM® SA-1100 Microprocessor (SA-1100).

12.1 Absolute Maximum Ratings

Table 12-1 lists the absolute maximum ratings for the SA-1100.

Table 12-1. SA-1100 DC Maximum Ratings

Symbol	Parameter	Min	Max	Units	Note
VDD	Core supply voltage	VSS – 0.5	VSS + 2.1	V	1
VDDX	I/O voltage	MIN(VSS – 0.05, VDD – 0.3)	VSS + 3.6	V	1
Vip	Voltage applied to any pin	VSS – 0.5	VSS + 3.6	V	1
Vip (*XTAL)	Voltage applied to *XTAL pins	0	1	V	1
Ts	Storage temperature	– 40	125	°C	1

NOTE:
1. These are stress SA-1100 ratings only. Exceeding the absolute maximum ratings may permanently damage the device. Operating the device at absolute maximum ratings for extended periods may affect device reliability.

12.2 DC Operating Conditions

Table 12-2 lists the functional operating dc parameters for the SA-1100.

Table 12-2. SA-1100 DC Operating Conditions

Symbol	Parameter	Min	Nom	Max	Units	Notes
Vihc†	IC input high voltage	0.8 × VDDX	—	VDDX	V	1, 2
Vilc†	IC input low voltage	0.0	—	0.2 × VDDX	V	1, 2
Vohc	OCZ output high voltage	0.8 × VDDX	—	VDDX	V	1, 3
Volc	OCZ output low voltage	0.0	—	0.2 × VDDX	V	1, 3
Iohc	High-level output current	—	—	– 2	mA	—
Iolc	Low-level output current	—	—	2	mA	—
Ta	Ambient operating temperature	0	—	70	°C	—
Iin	IC input leakage current	—	10	—	µA	—
Ioh	Output high current (Vout = VDD – 0.4 V)	—	2	—	mA	—
Iol	Output high current (Vout = VSS + 0.4 V)	—	2	—	mA	—
Cin	Input capacitance	—	5	—	pF	4
ESD	HBM model ESD	—	1	—	KV	—

NOTES:
1. Voltages measured with respect to **VSS**.
2. IC – CMOS-level inputs (includes IC and ICOCZ pin types).
3. OCZ – Output, CMOS levels, tristateable.
4. Parameter guaranteed by design

† Not tested at this time

12.3 Power Supply Voltages and Currents

Table 12-3 specifies the power supply voltages and currents for the SA-1100. For power supply voltages and currents for 2.0-V devices, contact the Intel Massachusetts Customer Technology Center..

Table 12-3. SA-1100 Power Supply Voltages and Currents with TQFP Package

Parameter	SA-1100				Units
	AA/AB†	CA/CB†	DA/DB†	EA/EB†	
Maximum operating frequency	133	160	220	190	MHz
Maximum run mode power (total VDD + VDDX)	400	1100	1100	500	mW
Typical run mode power (total VDD + VDDX)	230	430	550	330	mW
Maximum idle mode power†† (total VDD + VDDX)	55	n/a	n/a	85	mW
Typical idle mode power†† (total VDD + VDDX)	50	n/a	n/a	65	mW
Maximum sleep mode current †† (total VDD + VDDX)	50	n/a	n/a	50	uA
Typical sleep mode current †† (total VDD + VDDX)	25	n/a	n/a	30	uA
VDD					
Minimum internal power supply voltage	1.42	1.90	1.90	1.42	V
Nominal internal power supply voltage	1.50	2.00	2.00	1.50	V
Maximum internal power supply voltage	1.58	2.10	2.10	1.58	V
VDDX					
Minimum external power supply voltage	3.00	3.00	3.00	3.00	V
Nominal external power supply voltage	3.30	3.30	3.30	3.30	V
Maximum external power supply voltage	3.60	3.60	3.60	3.60	V

† AA, CA, DA and EA refer to TQFP package. AB, CB, DB and EB refer to mBGA package.

†† Room temperature specification.

AC Parameters 13

This chapter defines the ac parameters for the Intel® StrongARM® SA-1100 Microprocessor (SA-1100).

13.1 Test Conditions

The AC timing diagrams presented in this chapter assume that the outputs of SA-1100 have been loaded with a 50-pF capacitive load on output signals. The output pads of SA-1100 are CMOS drivers that exhibit a propagation delay that increases with the increase in load capacitance. Table 13-1 lists the output derating figure for each output pad, showing the approximate rate of increase of delay with increasing or decreasing load capacitance for a typical process at room temperature. For derating figures for 2.0-V devices, contact the Intel Massachusetts Customer Technology Center.

Table 13-1. SA-1100 Output Derating

Output Signal	Load for Nominal Value	Output Derating (ns/pF) VDD = 1.5 V rising edge	Output Derating (ns/pF) VDD = 1.5 V falling edge	Output Derating (ns/pF) VDD = 2.0 V rising edge	Output Derating (ns/pF) VDD = 2.0 V falling edge	Note
All outputs	50 pF	0.086	0.077	0.08	0.072	1

NOTE:
1. Parameter guaranteed by design

13.2 Module Considerations

The edge rates for the SA-1100 processor are such that the lumped load model presented above can only be used for etch lengths up to one inch. Over one inch of etch, the signal is a transmission line and needs to be modeled as such.

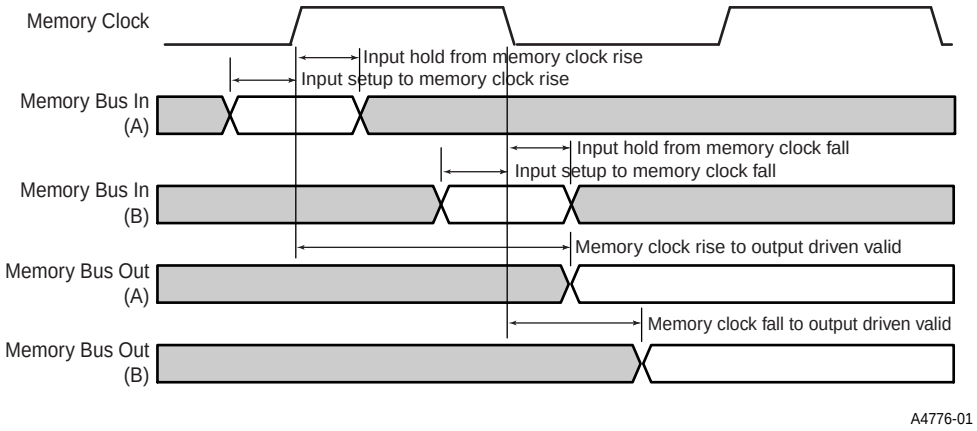
13.3 Memory Bus and PCMCIA Signal Timings

During production test, the SA-1100 is placed in testclock bypass mode by the assertion of the TCKBYP pin. This mode (not intended for use by customers) bypasses the 3.686-MHz oscillator and the main PLL and sources the processor clock from the TESTCLK pin. During this test mode, all clocks on the SA-1100 are synchronous to TESTCLK. In this mode, the basic functionality of the chip is tested and the pin timings relative to TESTCLK are measured. The ac parameters are measured in this way for each available processor clock speed and supply voltage at which the device is offered.

The ac specifications for the SA-1100 memory and PCMCIA interfaces are provided relative to the memory clock. In the testclock bypass mode, memory clock is one-half the frequency of TESTCLK. Under normal operation, memory clock is one-half the frequency of the processor clock generated by the main PLL.

Even though this clock is not visible to the user, the required pin timing may be inferred through these numbers. Input pins are specified by a required setup and hold to the memory clock. Outputs are specified by a propagation delay from the edge of the memory clock where the drive starts to the time the pin actually transitions. A 50-pF lumped load is assumed to be on each pin. Figure 13-1 shows the memory bus ac timing definitions and Table 13-2 describes the ac timing parameters.

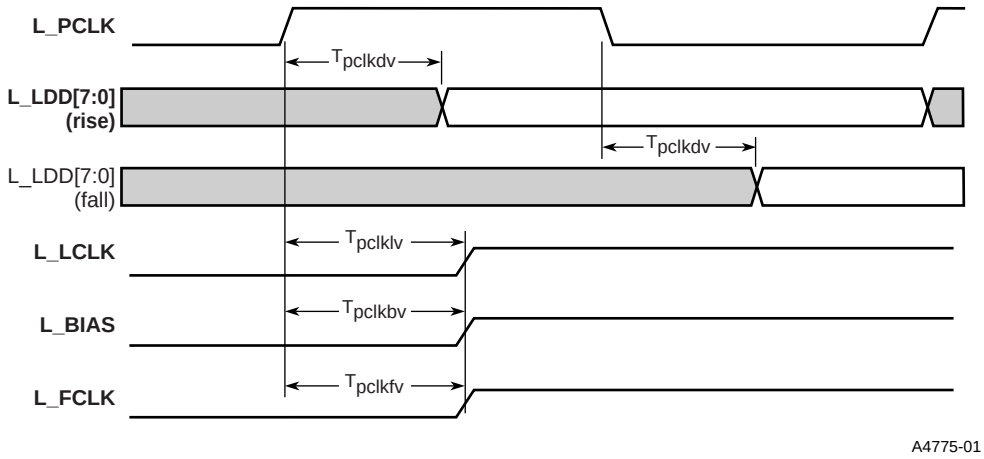
Figure 13-1. Memory Bus AC Timing Definitions



13.4 LCD Controller Signals

Figure 13-2 describes the LCD timing parameters. The LCD pin timing specifications are referenced to the pixel clock (L_PCLK).

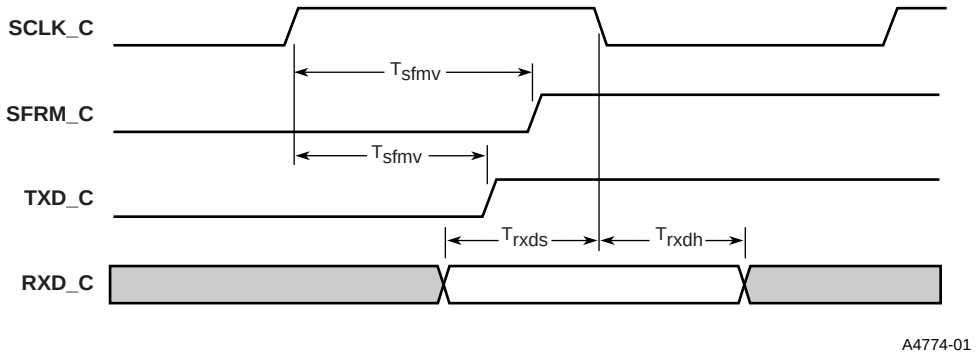
Figure 13-2. LCD AC Timing Definitions



13.5 MCP Signals

Figure 13-3 describes the MCP timing parameters. The MCP pin timing specifications are referenced to SCLK_C.

Figure 13-3. MCP AC Timing Definitions



13.6 Timing Parameters

Table 13-2 lists the ac timing parameters for the SA-1100 for AA and BA parts. For timing parameters for 2.0-V devices, contact the Intel Massachusetts Customer Technology Center.

Table 13-2. SA-1100 AC Timing Table for AA and BA Parts

Pin Name	Symbol	Parameter	Min	Max	Unit	Note
Memory Bus						
D>31:0>	Tdfov	Memory clock fall to D<31:0> driven valid	—	10	ns	—
	Tds	D<31:0> valid to memory clock rise/fall	0	—	ns	1
	Tdh	Memory clock rise/fall to data invalid (input	4	—	ns	1
nPOE, nPWE, nPIOR, nPIOW, PSKTSEL, nPREG, nPCE<1,2>, A<25:0>	Tmfov	Memory clock fall to output driven valid	—	10	ns	5 — — — —
nIOIS16	Tio16s	nIOIS16 valid to memory clock rise (input	1	—	ns	6
	Tio16h	Memory clock rise to nIOIS16 negated	3	—	ns	6
nWE, nOE	Tmrov	Memory clock rise to output driven valid	—	10	ns	— —
nRAS<3:0>	Tmrdv	Memory clock rise to output driven valid	—	12	ns	—
nCAS<3:0>	Tcasd	Memory clock rise/fall to nCAS<3:0> driven	—	12	ns	2
nCS<3:0>	Tcsd	Memory clock rise to nCS<3:0> driven valid	—	10	ns	—
MCP (CODEC) Interface						
SFRM_C	Tsfrmv	SCLK_C rise to SFRM_C driven valid	—	21	ns	—
RXD_C	Trxds	RXD_C valid to SCLK_C fall (input setup)	0	—	ns	—
	Trxdh	SCLK_C fall to RXD_C invalid (input hold)	4	—	ns	—
TXD_C	Ttxdv	SCLK_C rise to TXD_C valid	—	22	ns	—
LCD Controller						
L_LDD<7:0>	Tpclkdv	L_PCLK rise/fall to L_LDD<7:0> driven valid	—	14	ns	3
L_LCLK	Tpclklv	L_PCLK fall to L_LCLK driven valid	—	14	ns	4
L_FCLK	Tpclkfv	L_PCLK fall to L_LFCLK driven valid	—	14	ns	4
L_BIAS	Tpclkbv	L_PCLK rise to L_BIAS driven valid	—	14	ns	4

NOTES:
1. These input pins may be sampled on either the rising or falling edge of the memory clock.
2. These output pins may be driven on either the rising or falling edge of the memory clock.
3. The LCD data pins can be programmed to be driven on either the rising or falling edge of the pixel clock (L_PCLK).
4. These LCD signals can, at times, transition when L_PCLK is not clocking (between frames). At this time, they are clocked with the internal version of the pixel clock before it is driven out onto the L_PCLK pin.
5. These signals are PCMCIA outputs and are driven by a state machine clocked by BCLK. The user defines BCLK by programming the number of processor clocks per BCLK. Two processor clocks make one memory clock cycle. To ensure proper operation, the user must adhere to the protocol description.
6. These signals are PCMCIA inputs and are sampled by a state machine clocked by BCLK. The user defines BCLK by programming the number of processor clocks per BCLK. Two processor clocks make one memory clock cycle. To ensure proper operation, the user must adhere to the protocol description.

13.6.1 Asynchronous Signal Timing Descriptions

nPWAIT is an input and is received through a synchronizer. As such, it has no setup and hold specification. The user must adhere to the protocol definition.

When the peripheral pins are in GPIO mode, they are read or written under software control. As outputs, they are driven valid on the pin approximately 20 ns after they are written by software. When inputs, they are received by a synchronizer and must be valid for approximately 20 ns before they are able to be recognized by a CPU read.

nRESET must remain asserted for 150 ms after VDD and VDDX are stable to properly reset the SA-1100.

nRESET_OUT is asserted for all types of reset (hard, watchdog, sleep, and software) and appears on the pin asynchronously to all clocks.

BATT_FAULT and VDD_FAULT are asynchronous inputs and are synchronized to the 32.768-kHz clock after entering the SA-1100. They must be valid for approximately 60 ms before they are recognized by the SA-1100.

PWR_EN asserts when the SA-1100 enters sleep mode and is driven onto the pin following the rising edge of the 32.768-kHz clock. It negates on the same edge as sleep mode is exited.

GP<27:0> are read and written under software control. In addition, an asynchronous edge detect may be performed. When writing a value to these pins, the pin transitions approximately 20 ns after the write is performed. When reading these pins, the signal is first synchronized to the internal memory clock and must be valid for at least 20 ns before it is visible to a processor read. For edge detects, the value on the pin following an edge must be stable for at least 10 ns for the edge to be caught by the edge detect circuit.

UDC+, UDC-, TXD_1, RXD_1, TXD_2, RXD_2, TXD_3, and RXD_3 are asynchronous relative to any device outside the SA-1100. The output pins, like all outputs on the SA-1100, have been characterized while driving a 50-pF lumped load capacitance.

Package and Pinout 14

14.1 Mechanical Data and Packaging Information

Figure 14-1 shows the SA-1100 208-pin LQFP mechanical drawing. All measurements are in millimeters. Table 14-1 lists the SA-1100 pins in numeric order, showing the signal type for each pin.

Figure 14-1. Quad Flat Pack – 1.4mm (LQFP)

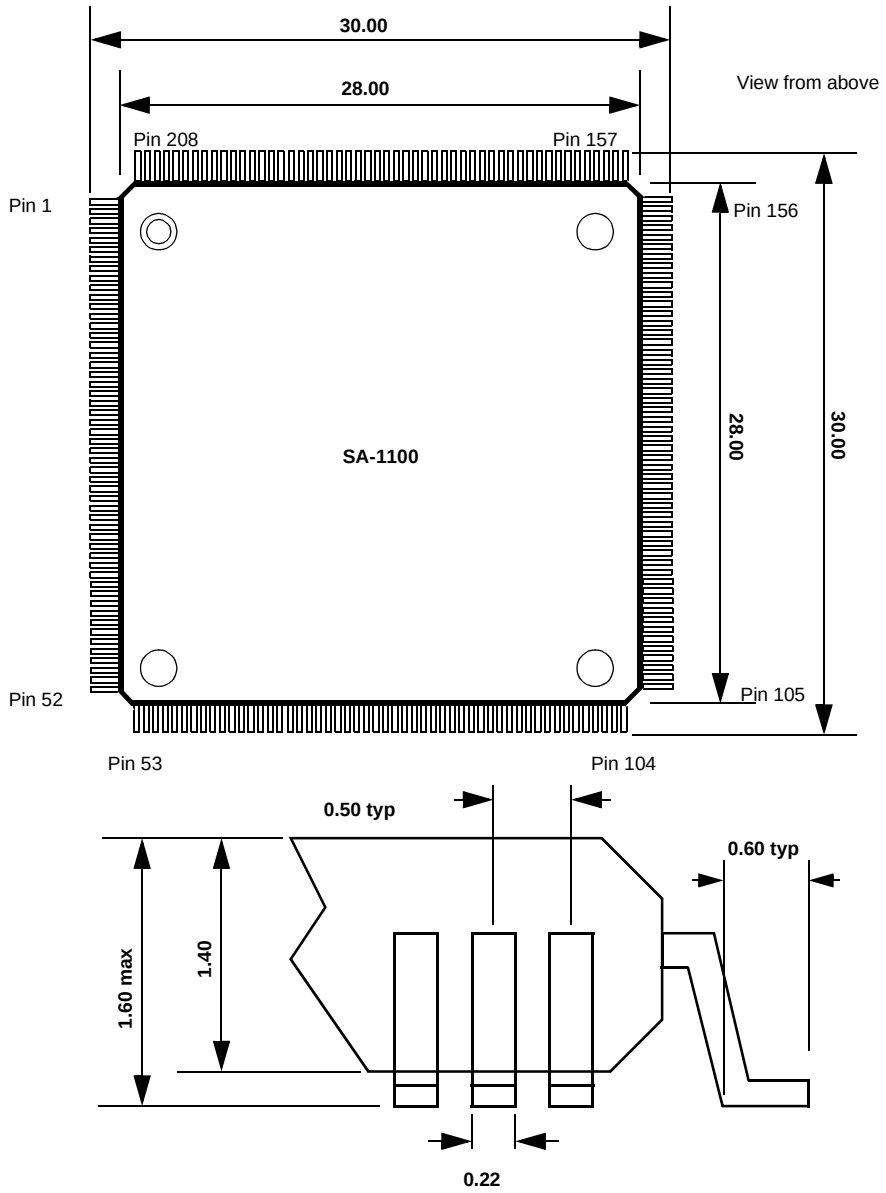


Table 14-1. SA-1100 Pinout – 208-Pin Quad Flat Pack

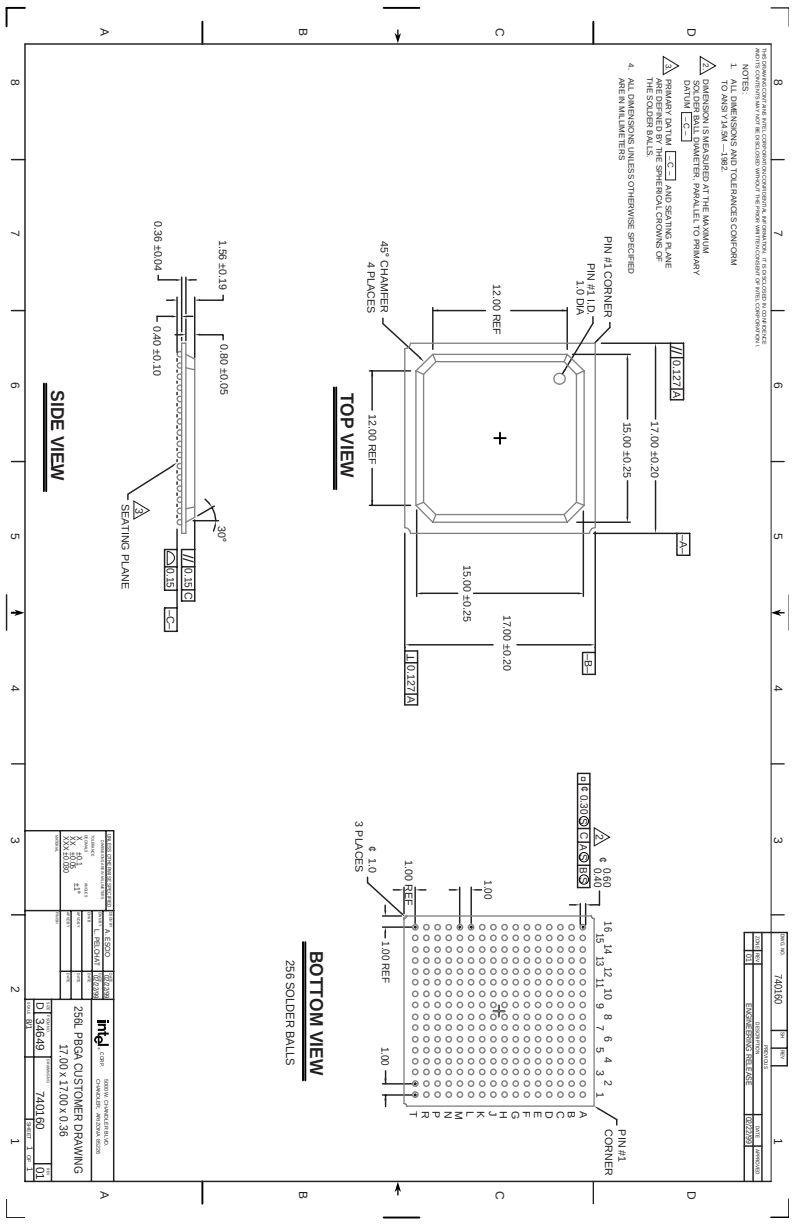
Pin	Signal	Type	Pin	Signal	Type	Pin	Signal	Type	Pin	Signal	Type
1	RXD_C	I/O	53	GP[25]	I/O	105	nPIOR	O	157	A[11]	O
2	TXD_C	I/O	54	GP[24]	I/O	106	nPIOW	O	158	A[10]	O
3	**VDDX2**	–	55	GP[23]	I/O	107	**VSSX**	–	159	A[9]	O
4	**VSSX**	–	56	GP[22]	I/O	108	**VDDX2**	–	160	A[8]	O
5	**VDD**	–	57	**VDDX1**	–	109	**VSS**	–	161	**VSSX**	–
6	**VSS**	–	58	**VSSX**	–	110	**VDD**	–	162	**VDDX1**	–
7	D[0]	I/O	59	GP[21]	I/O	111	PSKTSEL	O	163	A[7]	O
8	D[8]	I/O	60	GP[20]	I/O	112	nIOIS16	I	164	A[6]	O
9	D[16]	I/O	61	GP[19]	I/O	113	nPWAIT	I	165	A[5]	O
10	D[24]	I/O	62	GP[18]	I/O	114	nPREG	O	166	A[4]	O
11	D[1]	I/O	63	GP[17]	I/O	115	nPCE2	O	167	A[3]	O
12	D[9]	I/O	64	GP[16]	I/O	116	nPCE1	O	168	A[2]	O
13	D[17]	I/O	65	GP[15]	I/O	117	nWE	O	169	A[1]	O
14	D[25]	I/O	66	GP[14]	I/O	118	nOE	O	170	A[0]	O
15	**VDDX2**	–	67	**VDDX1**	–	119	**VSSX**	–	171	**VSSX**	–
16	**VSSX**	–	68	**VSSX**	–	120	**VDDX2**	–	172	**VDDX1**	–
17	D[2]	I/O	69	GP[13]	I/O	121	nRAS[3]	O	173	UDC-	I/O
18	D[10]	I/O	70	GP[12]	I/O	122	nRAS[2]	O	174	UDC+	I/O
19	D[18]	I/O	71	GP[11]	I/O	123	nRAS[1]	O	175	RXD_1	I/O
20	D[26]	I/O	72	GP[10]	I/O	124	nRAS[0]	O	176	TXD_1	I/O
21	D[3]	I/O	73	GP[9]	I/O	125	nCAS[3]	O	177	RXD_2	I/O
22	D[11]	I/O	74	GP[8]	I/O	126	nCAS[2]	O	178	TXD_2	I/O
23	D[19]	I/O	75	GP[7]	I/O	127	nCAS[1]	O	179	RXD_3	I/O
24	D[27]	I/O	76	GP[6]	I/O	128	nCAS[0]	O	180	TXD_3	I/O
25	**VDD**	–	77	**VDDX1**	–	129	**VSSX**	–	181	**VSSX**	–
26	**VSS**	–	78	**VSSX**	–	130	**VDDX2**	–	182	**VDDX1**	–
27	**VDDX2**	–	79	**VDD**	–	131	**VSS**	–	183	**VSS**	–
28	**VSSX**	–	80	**VSS**	–	132	**VDD**	–	184	TXTAL	I
29	D[4]	I/O	81	GP[5]	I/O	133	nCS[3]	O	185	TEXTAL	O
30	D[12]	I/O	82	GP[4]	I/O	134	nCS[2]	O	186	PEXTAL	O
31	D[20]	I/O	83	GP[3]	I/O	135	nCS[1]	O	187	PXTAL	I
32	D[28]	I/O	84	GP[2]	I/O	136	nCS[0]	O	188	**VDDP**	–
33	D[5]	I/O	85	GP[1]	I/O	137	A[25]	O	189	**VSS**	–
34	D[13]	I/O	86	GP[0]	I/O	138	A[24]	O	190	**VDD**	–
35	D[21]	I/O	87	L_BIAS	I/O	139	A[23]	O	191	nRESET	I
36	D[29]	I/O	88	L_PCLK	I/O	140	A[22]	O	192	nRESET_OUT	O
37	**VDDX2**	–	89	**VDDX1**	–	141	**VSSX**	–	193	**VDDX3**	I
38	**VSSX**	–	90	**VSSX**	–	142	**VDDX2**	–	194	ROMSEL	I
39	D[6]	I/O	91	LDD0	I/O	143	A[21]	O	195	TCK_BYP	I
40	D[14]	I/O	92	LDD1	I/O	144	A[20]	O	196	TESTCLK	I
41	D[22]	I/O	93	LDD2	I/O	145	A[19]	O	197	TMS	I
42	D[30]	I/O	94	LDD3	I/O	146	A[18]	O	198	TCK	I
43	D[7]	I/O	95	LDD4	I/O	147	A[17]	O	199	TDI	I
44	D[15]	I/O	96	LDD5	I/O	148	A[16]	O	200	TDO	O
45	D[23]	I/O	97	LDD6	I/O	149	A[15]	O	201	nTRST	I
46	D[31]	I/O	98	LDD7	I/O	150	A[14]	O	202	BATT_FAULT	I
47	**VDD**	–	99	**VDDX1**	–	151	**VSS**	–	203	**VSSX**	–
48	**VSS**	–	100	**VSSX**	–	152	**VDD**	–	204	**VDDX1**	–
49	**VDDX2**	–	101	L_LCLK	I/O	153	**VSSX**	–	205	VDD_FAULT	I
50	**VSSX**	–	102	L_FCLK	I/O	154	**VDDX2**	–	206	PWR_EN	O
51	GP[27]	I/O	103	nPOE	O	155	A[13]	O	207	SFRM_C	O
52	GP[26]	I/O	104	nPWE	O	156	A[12]	O	208	SCLK_C	O

Note: All **VDDX1**, **VDDX2**, and **VDDX3** pins should be connected directly to the **VDDX** power plane of the system board. **VDDP** should be connected directly to the **VDD** plane of the system board.

14.2 Mini-Ball Grid Array – (mBGA)

Figure 14-2 shows the SA-1100 256 mini-ball grid array (mBGA) mechanical drawing. Table 14-2 lists the SA-1100 pins in numeric order, showing the signal type for each pin.

Figure 14-2. SA-1100 256 Mini-Ball Grid Array Mechanical Drawing



A6843-01

Table 14-2. SA-1100 Pinout – 256-Pin Mini-Ball Grid Array

Pin	Signal	Type	BGA Pad	Pin	Signal	Type	BGA Pad	Pin	35	Type	BGA Pad	Pin	Signal	Type	BGA Pad
1	RXD_C	I/O	B1	65	GP[15]	I/O	N6	129	**VSSX**	–	G7	193	**VDDX3**	I	D7
2	TXD_C	I/O	C2	66	GP[14]	I/O	P6	130	**VDDX2**	–	L12	194	ROMSEL	I	D6
3	**VDDX2**	–	J13	67	**VDDX1**	–	D9	131	**VSS**	–	J16	195	TCK_BYP	I	A6
4	**VSSX**	–	A1	68	**VSSX**	–	F7	132	**VDD**	–	J14	196	TESTCLK	I	B6
5	**VDD**	–	C1	69	GP[13]	I/O	R6	133	nCS[3]	O	H14	197	TMS	I	C6
6	**VSS**	–	D3	70	GP[12]	I/O	R7	134	nCS[2]	O	H13	198	TCK	I	C5
7	D[0]	I/O	D2	71	GP[11]	I/O	T6	135	nCS[1]	O	H16	199	TDI	I	A5
8	D[8]	I/O	D1	72	GP[10]	I/O	P7	136	nCS[0]	O	H15	200	TDO	O	B5
9	D[16]	I/O	F4	73	GP[9]	I/O	T7	137	A[25]	O	G14	201	nTRST	I	B4
10	D[24]	I/O	E3	74	GP[8]	I/O	N8	138	A[24]	O	G16	202	BATT_FAULT	I	A4
11	D[1]	I/O	E2	75	GP[7]	I/O	P8	139	A[23]	O	G15	203	**VSSX**	–	H7
12	D[9]	I/O	E1	76	GP[6]	I/O	R8	140	A[22]	O	F15	204	**VDDX1**	–	E8
13	D[17]	I/O	F3	77	**VDDX1**	–	K10	141	**VSSX**	–	G8	205	**VDD_FAULT**	I	C4
14	D[25]	I/O	F2	78	**VSSX**	–	F8	142	**VDDX2**	–	L13	206	PWR_EN	O	A3
15	**VDDX2**	–	K5	79	**VDD**	–	T8	143	A[21]	O	F14	207	SFRM_C	O	B3
16	**VSSX**	–	B2	80	**VSS**	–	R9	144	A[20]	O	F13	208	SCLK_C	O	A2
17	D[2]	I/O	F1	81	GP[5]	I/O	P9	145	A[19]	O	F16	–	**VSSX**	–	H8
18	D[10]	I/O	G2	82	GP[4]	I/O	T9	146	A[18]	O	E15	–	**VSSX**	–	H9
19	D[18]	I/O	G3	83	GP[3]	I/O	N10	147	A[17]	O	E14	–	**VSSX**	–	H10
20	D[26]	I/O	H4	84	GP[2]	I/O	R10	148	A[16]	O	E16	–	**VSSX**	–	H11
21	D[3]	I/O	G1	85	GP[1]	I/O	P10	149	A[15]	O	D14	–	**VSSX**	–	J6
22	D[11]	I/O	H3	86	GP[0]	I/O	T10	150	A[14]	O	D15	–	**VSSX**	–	J7
23	D[19]	I/O	H2	87	L_BIAS	I/O	R11	151	**VSS**	–	D16	–	**VSSX**	–	J8
24	D[27]	I/O	J3	88	L_PCLK	I/O	P11	152	**VDD**	–	C15	–	**VSSX**	–	J9
25	**VDD**	–	H1	89	**VDDX1**	–	D11	153	**VSSX**	–	G9	–	**VSSX**	–	J10
26	**VSS**	–	J2	90	**VSSX**	–	F9	154	**VDDX2**	–	M5	–	**VSSX**	–	J11
27	**VDDX2**	–	D13	91	LDD0	I/O	N12	155	A[13]	O	C16	–	**VSSX**	–	K6
28	**VSSX**	–	C3	92	LDD1	I/O	T11	156	A[12]	O	B16	–	**VSSX**	–	K7
29	D[4]	I/O	J1	93	LDD2	I/O	R12	157	A[11]	O	C14	–	**VSSX**	–	K8
30	D[12]	I/O	K4	94	LDD3	I/O	P12	158	A[10]	O	B14	–	**VSSX**	–	K9
31	D[20]	I/O	K3	95	LDD4	I/O	P13	159	A[9]	O	B15	–	**VSSX**	–	L6
32	D[28]	I/O	K2	96	LDD5	I/O	T12	160	A[8]	O	A16	–	**VSSX**	–	L7
33	D[5]	I/O	K1	97	LDD6	I/O	R13	161	**VSSX**	–	G10	–	**VSSX**	–	L8
34	D[13]	I/O	L3	98	LDD7	I/O	T13	162	**VDDX1**	–	E6	–	**VSSX**	–	L9
35	D[21]	I/O	L2	99	**VDDX1**	–	K11	163	A[7]	O	A15	–	**VDDX1**	–	L11
36	D[29]	I/O	L1	100	**VSSX**	–	F10	164	A[6]	O	A14	–	**VDDX1**	–	E9
37	**VDDX2**	–	K12	101	L_LCLK	I/O	R14	165	A[5]	O	B13	–	**VDDX1**	–	E10
38	**VSSX**	–	D4	102	L_FCLK	I/O	T14	166	A[4]	O	C13	–	**VDDX1**	–	E11
39	D[6]	I/O	M4	103	nPOE	O	R15	167	A[3]	O	A13	–	**VDDX1**	–	M6
40	D[14]	I/O	M3	104	nPWE	O	T15	168	A[2]	O	B12	–	**VDDX1**	–	M7
41	D[22]	I/O	M2	105	nPIOR	O	P14	169	A[1]	O	C12	–	**VDDX1**	–	M8
42	D[30]	I/O	M1	106	nPIOW	O	P15	170	A[0]	O	D12	–	**VDDX1**	–	M9
43	D[7]	I/O	N3	107	**VSSX**	–	F11	171	**VSSX**	–	G11	–	**VDDX1**	–	M10
44	D[15]	I/O	N2	108	**VDDX2**	–	L4	172	**VDDX1**	–	E7	–	**VDDX1**	–	M11
45	D[23]	I/O	P3	109	**VSS**	–	T16	173	UDC-	I/O	A12	–	**VDDX1**	–	N7
46	D[31]	I/O	P2	110	VDD	–	R16	174	UDC+	I/O	C11	–	**VDDX1**	–	N9
47	**VDD**	–	N1	111	PSKTSEL	O	P16	175	RXD_1	I/O	B11	–	**VDDX1**	–	N11
48	**VSS**	–	P1	112	nIOIS16	I	N15	176	TXD_1	I/O	A11	–	**VDDX2**	–	E12
49	**VDDX2**	–	E4	113	nPWAIT	I	N16	177	RXD_2	I/O	B10	–	**VDDX2**	–	E13
50	**VSSX**	–	E5	114	nPREG	O	N14	178	TXD_2	I/O	D10	–	**VDDX2**	–	F5
51	GP[27]	I/O	R1	115	nPCE2	O	M13	179	RXD_3	I/O	C10	–	**VDDX2**	–	F12
52	GP[26]	I/O	T1	116	nPCE1	O	M15	180	TXD_3	I/O	A10	–	**VDDX2**	–	G4
53	GP[25]	I/O	R2	117	nWE	O	M14	181	**VSSX**	–	H6	–	**VDDX2**	–	G5
54	GP[24]	I/O	P4	118	nOE	O	M16	182	**VDDX1**	–	L10	–	**VDDX2**	–	G12
55	GP[23]	I/O	T2	119	**VSSX**	–	G6	183	VSS	–	A9	–	**VDDX2**	–	G13
56	GP[22]	I/O	R3	120	**VDDX2**	–	L5	184	TXTAL	I	B9	–	**VDDX2**	–	H5
57	**VDDX1**	–	D5	121	nRAS[3]	O	L15	185	TEXTAL	O	C9	–	**VDDX2**	–	H12
58	**VSSX**	–	F6	122	nRAS[2]	O	L14	186	PEXTAL	O	A8	–	**VDDX2**	–	J4
59	GP[21]	I/O	T3	123	nRAS[1]	O	L16	187	PXTAL	I	B8	–	**VDDX2**	–	J5
60	GP[20]	I/O	R4	124	nRAS[0]	O	K13	188	**VDDP**	–	C8	–	**VDDX2**	–	J12
61	GP[19]	I/O	T4	125	nCAS[3]	O	K15	189	**VSS**	–	D8	–	**VDDX2**	–	M12
62	GP[18]	I/O	P5	126	nCAS[2]	O	K14	190	**VDD**	–	A7	–	**VDDX2**	–	N4
63	GP[17]	I/O	R5	127	nCAS[1]	O	K16	191	nRESET	I	B7	–	**VDDX2**	–	N5
64	GP[16]	I/O	T5	128	nCAS[0]	O	J15	192	nRESET_OUT	O	C7	–	**VDDX2**	–	N13

Note: All **VDDX1**, **VDDX2**, and **VDDX3** pins should be connected directly to the **VDDX** power plane of the system board. **VDDP** should be connected directly to the **VDD** plane of the system board.

Debug Support 15

Due to the integration level of the Intel® StrongARM® SA-1100 Microprocessor (SA-1100), many functions are not directly visible on the external pins. Therefore, some basic debug facilities are provided that are not present on the Intel® StrongARM® SA-110 Microprocessor (SA-110). These facilities are in the form of breakpoints that provide the user with the ability to stop execution after seeing a specific reference in either the instruction or data streams. Execution then proceeds to an exception routine during which the user may examine the internal state of the machine. The instruction and data breakpoint facilities are described in this chapter. The breakpoints are enabled through additions to coprocessor 15.

15.1 Instruction Breakpoint

The instruction breakpoint allows the user to stop the processor execution after the execution of an instruction at a selected address. This address is programmed into the instruction breakpoint address and control register (IBCR). This register is 32 bits wide and contains the address value for the breakpoint, and a bit to enable the breakpoint. Bit 0 is the enable bit. When set, this bit enables the breakpoint and when cleared, it disables the breakpoint. Bit 1 is reserved and has no effect when written. Bits 31..2 are compared against the fetch address to qualify the breakpoint. When the breakpoint is enabled, the SA-1100 executes until the instruction at this address is fetched and the fetch address equals the program counter (ignoring bits 0 and 1 of the address). At this point, the processor takes a prefetch abort exception. The interrupt routine must examine R14 (the saved program counter) to determine if the exception was caused by the breakpoint.

The IBCR is loaded by way of coprocessor 15, register 14. Access to this register is privileged. See the Section 5.1, “Internal Coprocessor Instructions” on page 5-1 for details on the format of the instruction used to access the IBCR.

15.2 Data Breakpoint

The data breakpoint allows the user to stop the processor execution after a load or store operation to a particular address. The data breakpoint address is programmed into the data breakpoint address register (DBAR) and is a full 32-bit value (to permit breakpoints on byte accesses).

For stores, the breakpoint condition may also be programmed to include a particular data pattern as well as the reference address. The data value is programmed by way of the data breakpoint value register (DBVR) and the data breakpoint mask register (DBMR). The DBVR is a 32-bit register containing the value against which the store data is compared. The data value can be further qualified through the data breakpoint mask register (DBMR). The DBMR is a 32-bit register containing mask information indicating which bits in the store data should be compared against the DBMR. A 1 in a particular bit position in the DBMR indicates that bit in the DBVR should be compared against the store data to qualify the breakpoint. To cause a breakpoint on a store data value, the address breakpoint must also be enabled, otherwise, no breakpoint will occur.

Breakpoints on loads are permitted only through an address match. Breakpoints on load address, store address, and store data are enabled and disabled through the data breakpoint control register (DBCR). A single bit is defined for each action. When a breakpoint is taken, the processor takes a data abort exception and sets bit 9 in the fault status register (FSR).

The DBAR, DBVR, and DBMR are loaded by way of coprocessor 15, register 14. Access to this register is privileged.

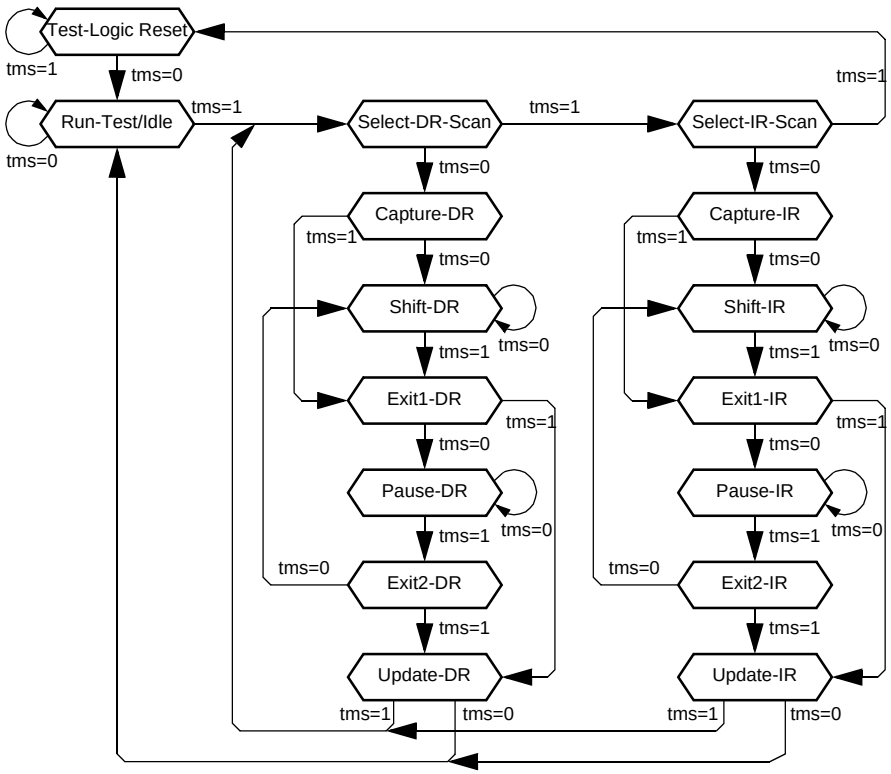
Boundary-Scan Test Interface 16

The boundary-scan interface conforms to the IEEE Std. 1149.1 – 1990, *Standard Test Access Port and Boundary-Scan Architecture.* (Refer to this standard for an explanation of the terms used in this section and for a description of the TAP controller states.) The Intel® StrongARM® SA-1100 Microprocessor (SA-1100) supports only JTAG continuity testing.

16.1 Overview

The boundary-scan interface provides a means of driving and sampling all the external pins of the device irrespective of the core state. This function permits testing of both the device's electrical connections to the circuit board and (in conjunction with other devices on the circuit board having a similar interface) testing the integrity of the circuit board connections between devices. The interface intercepts all external connections within the device, and each such "cell" is then connected together to form a serial shift register (the boundary-scan register). The whole interface is controlled via five dedicated pins: TDI, TMS, TCK, nTRST, and TDO. Figure 16-1 shows the state transitions that occur in the TAP controller. Note that all SA-1100 signals participate in the boundary scan. However, in the case of the PWR_EN pin, the contents of the scan latches are not placed on the pin. This is to prevent a scan operation from turning off power to the SA-1100.

Figure 16-1. Test Access Port (TAP) Controller State Transitions



16.2 Reset

The boundary-scan interface includes a state-machine controller (the TAP controller). In order to force the TAP controller into the correct state after power-up of the device, a reset pulse must be applied to the nTRST pin. If the boundary-scan interface is to be used, then nTRST must be driven low, and then high again. If the boundary-scan interface is not to be used, then the nTRST pin may be tied permanently low. Note that a clock on TCK is not necessary to reset the device.

The action of reset (either a pulse or a dc level) is as follows:

- System mode is selected (the boundary-scan chain does NOT intercept any of the signals passing between the pads and the core).
- IDcode mode is selected. If TCK is pulsed, the contents of the ID register will be clocked out of TDO.

16.3 Pull-Up Resistors

The IEEE 1149.1 standard effectively requires that TDI, nTRST, and TMS should have internal pull-up resistors. To minimize static current draw, nTRST has an internal pull-down resistor. These pins can be left unconnected for normal operation and overdriven to use the JTAG features.

16.4 Instruction Register

The instruction register is 5 bits in length. There is no parity bit. The fixed value loaded into the instruction register during the CAPTURE-IR controller state is: 00001.

16.5 Public Instructions

The following public instructions are supported:

Instruction	Binary Code
EXTEST	00000
SAMPLE/PRELOAD	00001
CLAMP	00100
HIGHZ	00101
IDCODE	00110
BYPASS	11111
Private	00010, 00011, 00111, 01000-01111, 10000-11110

In the descriptions that follow, TDI and TMS are sampled on the rising edge of TCK, and all output transitions on TDO occur as a result of the falling edge of TCK.

16.5.1 EXTEST (00000)

The boundary-scan (BS) register is placed in test mode by the EXTEST instruction. The EXTEST instruction connects the BS register between TDI and TDO. When the instruction register is loaded with the EXTEST instruction, all the boundary-scan cells are placed in their test mode of operation.

In the CAPTURE-DR state, inputs from the system pins and outputs from the boundary-scan output cells to the system pins are captured by the boundary-scan cells. In the SHIFT-DR state, the previously captured test data is shifted out of the BS register via the TDO pin, while new test data is shifted in via the TDI pin to the BS register parallel input latch. In the UPDATE-DR state, the new test data is transferred into the BS register parallel output latch. Note that this data is applied immediately to the system logic and system pins.

16.5.2 SAMPLE/PRELOAD (00001)

The BS register is placed in normal (system) mode by the SAMPLE/PRELOAD instruction. The SAMPLE/PRELOAD instruction connects the BS register between TDI and TDO. When the instruction register is loaded with the SAMPLE/PRELOAD instruction, all the boundary-scan cells are placed in their normal system mode of operation.

In the CAPTURE-DR state, a snapshot of the signals at the boundary-scan cells is taken on the rising edge of TCK. Normal system operation is unaffected. In the SHIFT-DR state, the sampled test data is shifted out of the BS register via the TDO pin, while new data is shifted in via the TDI pin to preload the BS register parallel input latch. In the UPDATE-DR state, the preloaded data is transferred into the BS register parallel output latch. Note that this data is not applied to the system logic or system pins while the SAMPLE/PRELOAD instruction is active. This instruction should be used to preload the boundary-scan register with known data prior to selecting EXTEST instructions.

16.5.3 CLAMP (00100)

The CLAMP instruction connects a 1-bit shift register (the BYPASS register) between TDI and TDO.

When the CLAMP instruction is loaded into the instruction register, the state of all output signals is defined by the values previously loaded into the boundary-scan register. A guarding pattern (specified for this device at the end of this section) should be preloaded into the boundary-scan register using the SAMPLE/PRELOAD instruction prior to selecting the CLAMP instruction.

In the CAPTURE-DR state, a logic 0 is captured by the bypass register. In the SHIFT-DR state, test data is shifted into the bypass register via TDI and out via TDO after a delay of one TCK cycle. Note that the first bit shifted out will be a zero. The bypass register is not affected in the UPDATE-DR state.

16.5.4 HIGHZ (00101)

The HIGHZ instruction connects a 1-bit shift register (the BYPASS register) between TDI and TDO. When the HIGHZ instruction is loaded into the instruction register, all outputs are placed in an inactive drive state.

In the CAPTURE-DR state, a logic 0 is captured by the bypass register. In the SHIFT-DR state, test data is shifted into the bypass register via TDI and out via TDO after a delay of one TCK cycle. Note that the first bit shifted out will be a zero. The bypass register is not affected in the UPDATE-DR state.

16.5.5 IDCODE (00110)

The IDCODE instruction connects the device identification register (or ID register) between TDI and TDO. The ID register is a 32-bit register that allows the manufacturer, part number and version of a component to be determined through the TAP. When the instruction register is loaded with the IDCODE instruction, all the boundary-scan cells are placed in their normal (system) mode of operation.

In the CAPTURE-DR state, the device identification code (specified at the end of this section) is captured by the ID register. In the SHIFT-DR state, the previously captured device identification code is shifted out of the ID register via the TDO pin, while data is shifted in via the TDI pin into the ID register. In the UPDATE-DR state, the ID register is unaffected.

16.5.6 BYPASS (11111)

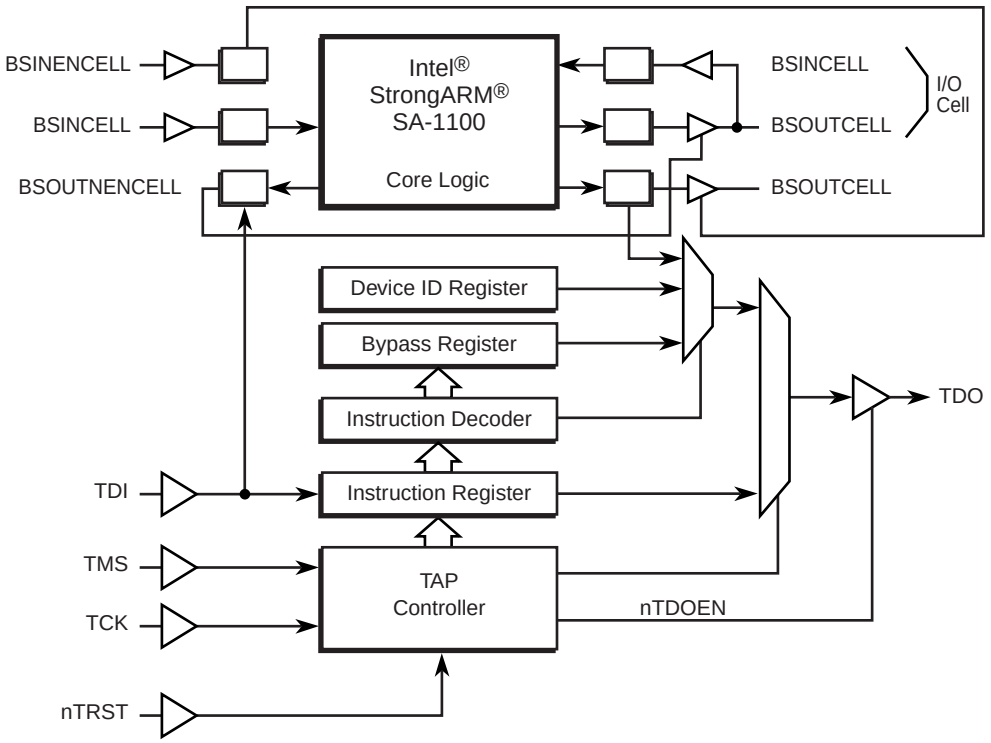
The BYPASS instruction connects a 1-bit shift register (the BYPASS register) between TDI and TDO. When the BYPASS instruction is loaded into the instruction register, all the boundary-scan cells are placed in their normal (system) mode of operation. This instruction has no effect on the system pins.

In the CAPTURE-DR state, a logic 0 is captured by the bypass register. In the SHIFT-DR state, test data is shifted into the bypass register via TDI and out via TDO after a delay of one TCK cycle. Note that the first bit shifted out will be a zero. The bypass register is not affected in the UPDATE-DR state.

16.6 Test Data Registers

Figure 16-2 illustrates the structure of the boundary-scan logic.

Figure 16-2. Boundary-Scan Block Diagram



* StrongARM is a registered trademark of ARM Limited.

A6839-01

16.6.1 Bypass Register

Purpose: This is a single-bit register that can be selected as the path between TDI and TDO to allow the device to be bypassed during boundary-scan testing.

Length: 1 bit

Operating Mode: When the BYPASS instruction is the current instruction in the instruction register, serial data is transferred from TDI to TDO in the SHIFT-DR state with a delay of one TCK cycle.

There is no parallel output from the bypass register.

A logic 0 is loaded from the parallel input of the bypass register in the CAPTURE-DR state.

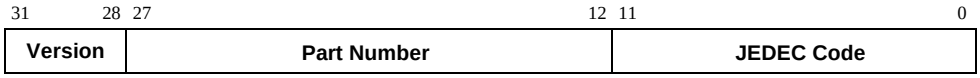
16.6.2 SA-1100 Device Identification (ID) Code Register

Purpose: This register is used to read the 32-bit device identification code. No programmable supplementary identification code is provided.

Length: 32 bits

Operating Mode: When the IDCODE instruction is current, the ID register is selected as the serial path between TDI and TDO.

The format of the ID register is as follows:



The high-order 4 bits of the ID register contains the version number of the silicon and changes with each new revision.

There is no parallel output from the ID register.

The 32-bit device identification code is loaded into the ID register from its parallel inputs during the CAPTURE-DR state.

16.6.3 SA-1100 Boundary-Scan (BS) Register

Purpose: The BS register consists of a serially connected set of cells around the periphery of the device, at the interface between the core logic and the system input/output pads. This register can be used to isolate the pins from the core logic and then drive or monitor the system pins.

Operating Modes: The BS register is selected as the register to be connected between TDI and TDO only during the SAMPLE/PRELOAD and EXTEST instructions. Values in the BS register are used, but are not changed, during the CLAMP instruction.

In the normal (system) mode of operation, straight-through connections between the core logic and pins are maintained, and normal system operation is unaffected.

In TEST mode (when EXTEST is the currently selected instruction), values can be applied to the output pins independently of the actual values on the input pins and core logic outputs. On the SA-1100, all of the boundary-scan cells include update registers; thus, all of the pins can be controlled in the above manner. An additional boundary-scan cell is interposed in the scan chain to control the enabling of the data bus.

The EXTEST guard values should be clocked into the boundary-scan register (using the SAMPLE/PRELOAD instruction) before the EXTEST instruction is selected, to ensure that known data is applied to the core logic during the test. These guard values should also be used when new EXTEST vectors are clocked into the boundary-scan register.

The values stored in the BS register after power-up are not defined. Similarly, the values previously clocked into the BS register are not guaranteed to be maintained across a boundary-scan reset (from forcing nTRST low or entering the Test Logic Reset state).

Figure 16-3, Figure 16-4, and Figure 16-5 show the typical timing for the BS register.

16.7 Boundary-Scan Interface Signals

Figure 16-3. Boundary-Scan General Timing

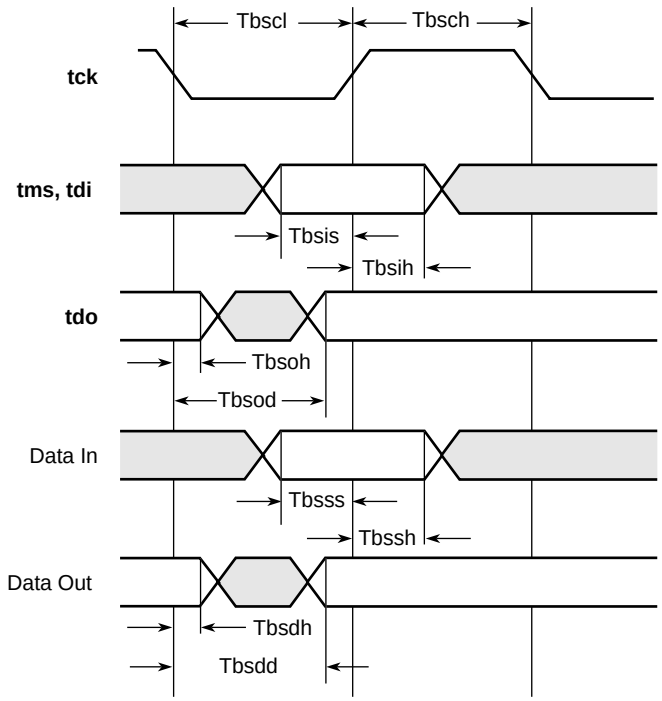
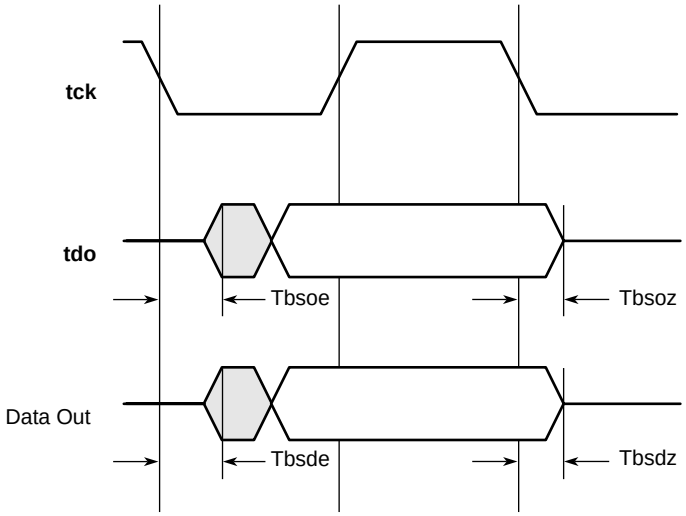
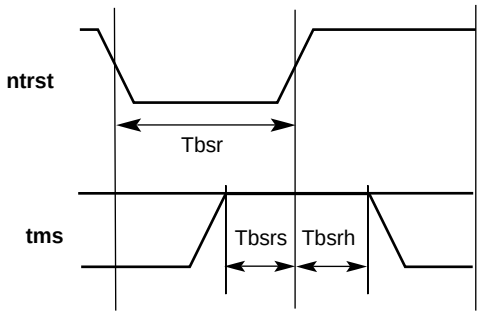


Figure 16-4. Boundary-Scan Tristate Timing



A4773-01

Figure 16-5. Boundary-Scan Reset Timing



A4771-01

Table 16-1 shows the SA-1100 boundary-scan interface timing specifications.

Table 16-1. SA-1100 Boundary-Scan Interface Timing

Symbol	Parameter	Minimum	Typical	Maximum	Units	Notes
Tbscl	**TCK** low period	50	–	–	ns	~~8~~
Tbsch	**TCK** high period	50	–	–	ns	~~8~~
Tbsis	**TDI**,**TMS** setup to [TCr]	10	–	–	ns	–
Tbsih	**TDI**,**TMS** hold from [TCr]	10	–	–	ns	–
Tbsoh	**TDO** hold time	5	–	–	ns	1
Tbsod	TCf to **TDO** valid	–	–	40	ns	1
Tbsss	I/O signal setup to [TCr]	5	–	–	ns	4
Tbssh	I/O signal hold from [TCr]	20	–	–	ns	4
Tbsdh	Data output hold time	5	–	–	ns	5
Tbsdd	TCf to data output valid	–	–	40	ns	–
Tbsoe	**TDO** enable time	5	–	–	ns	1,2
Tbsoz	**TDO** disable time	–	–	40	ns	1,3
Tbsde	Data output enable time	5	–	–	ns	5,6
Tbsdz	Data output disable time	–	–	40	ns	5,7
Tbsr	Reset period	30	—	—	ns	—
Tbsrs	**TMS** setup to [TRr]	10	—	—	ns	8
Tbsrh	**TMS** hold from [TRr]	10	—	—	ns	8

NOTES:
1. Assumes a 25-pF load on **TDO**. Output timing derates at 0.072 ns/pF of extra load applied.
2. TDO enable time applies when the TAP controller enters the Shift-DR or Shift-IR states.
3. TDO disable time applies when the TAP controller leaves the Shift-DR or Shift-IR states.
4. For correct data latching, the I/O signals (from the core and the pads) must be set up and held with respect to the rising edge of TCK in the CAPTURE-DR state of the SAMPLE/PRELOAD and EXTEST instructions.
5. Assumes that the data outputs are loaded with the ac test loads.
6. Data output enable time applies when the boundary-scan logic is used to enable the output drivers.
7. Data output disable time applies when the boundary scan is used to disable the output drivers.
8. TCK may be stopped indefinitely in either the low or high phase.

16.8 Memory Bus Alternate Master

Control is provided to allow an external device to request and become master of the memory bus by setting up a pair of GPIO pins to become MBREQ and MBGNT pins. This capability is intended for use by specialized systems (development platforms). The external device should not be master of the memory bus for longer than a DRAM refresh period because the SA-1100 will not perform refreshes while an external device is master of the bus.

To enter alternate master mode, set GPIO[21] to an output and GPIO[22] to an input through the GPIO pin direction register (GPDR). Then, set the corresponding bits in the GPIO alternate function register (GAFR) to select them for the non-GPIO function. Finally, set bit MR [bit 3 of the test unit control register, (TUCR)]. This configures GPIO[21] to act as MBREQ and GPIO[22] to act as MBGNT.

The protocol for these pins is straightforward. When an alternate master wants to take control of the memory bus, it asserts MBREQ. When the SA-1100 has stopped driving and tristates the memory bus address, data, and control pins, it asserts MBGNT. The SA-1100 keeps the pins in this state until MBREQ is negated. The SA-1100 then negates MBGNT, and begins to access and refresh the DRAM again.

Register Summary A

This appendix describes all of the Intel® StrongARM® SA-1100 Microprocessor (SA-1100) internal registers.

Physical Address	Symbol	Register Name
GPIO Registers		
0h 9004 0000	GPLR	GPIO pin level register.
0h 9004 0004	GPDR	GPIO pin direction register.
0h 9004 0008	GPSR	GPIO pin output set register.
0h 9004 000C	GPCR	GPIO pin output clear register.
0h 9004 0010	GRER	GPIO rising-edge register.
0h 9004 0014	GFER	GPIO falling-edge register.
0h 9004 0018	GEDR	GPIO edge detect status register.
0h 9004 001C	GAFR	GPIO alternate function register.
Interrupt Controller Registers		
0h 9005 0000	ICIP	Interrupt controller irq pending register.
0h 9005 0004	ICMR	Interrupt controller mask register.
0h 9005 0008	ICLR	Interrupt controller FIQ level register.
0h 9005 0010	ICFP	Interrupt controller FIQ pending register.
0h 9005 0020	ICPR	Interrupt controller pending register.
0h 9005 000c	ICPR	Interrupt controller control register.
Real-Time Clock Registers		
0h 9001 0004	RCNR	Real-time clock count register.
0h 9001 0000	RTAR	Real-time clock alarm register.
0h 9001 0010	RTSR	Real-time clock status register.
0h 9001 0008	RTTR	Real-time clock trim register.
OS Timer Registers		
0h 9000 0000	OSMR[0]	OS timer match registers[3:0].
0h 9000 0004	OSMR[1]	
0h 9000 0008	OSMR[2]	
0h 9000 000C	OSMR[3]	
0h 9000 0010	OSCR	OS timer counter register.
0h 9000 0014	OSSR	OS timer status register.
0h 9000 0018	OWER	OS timer watchdog enable register.
0h 9000 001C	OIER	OS timer interrupt enable register.

Physical Address	Symbol	Register Name
Power Manager Registers		
0h 9002 0000	PMCR	Power manager control register.
0h 9002 0004	PSSR	Power manager sleep status register.
0h 9002 0008	PSPR	Power manager scratchpad register.
0h 9002 000C	PWER	Power manager wakeup enable register.
0h 9002 0010	PCFR	Power manager configuration register.
0h 9002 0014	PPCR	Power manager PLL configuration register.
0h 9002 0018	PGSR	Power manager GPIO sleep state register.
0h 9002 001C	POSR	Power manager oscillator status register.
Reset Controller Registers		
0h 9003 0000	RSRR	Reset controller software reset register.
0h 9003 0004	RCSR	Reset controller status register.
0h 9003 0008	TUCR	Reserved for test.
Memory Controller Registers		
0xA000 0000	MDCNFG	DRAM configuration register.
0xA000 0004	MDCAS0	DRAM CAS waveform shift register 0.
0xA000 0008	MDCAS1	DRAM CAS waveform shift register 1.
0xA000 000C	MDCAS2	DRAM CAS waveform shift register 2.
0xA000 0010	MSC0	Static memory control register 0.
0xA000 0014	MSC1	Static memory control register 1.
0xA000 0018	MECR	Expansion bus configuration register.
DMA Controller Registers		
0h B000 0000	DDAR0	DMA device address register.
0h B000 0004	DCSR0	DMA control/status register 0 – write ones to set.
0h B000 0008		Write ones to clear.
0h B000 000C		Read only.
0h B000 0010	DBSA0	DMA buffer A start address 0.
0h B000 0014	DBTA0	DMA buffer A transfer count 0.
0h B000 0018	DBSB0	DMA buffer B start address 0.
0h B000 001C	DBTB0	DMA buffer B transfer count 0.
0h B000 0020	DDAR1	DMA device address register 1.
0h B000 0024	DCSR1	DMA control/status register 1 – write ones to set.
0h B000 0028		Write ones to clear.
0h B000 002C		Read only.
0h B000 0030	DBSA1	DMA buffer A start address 1.
0h B000 0034	DBTA1	DMA buffer A transfer count 1.
0h B000 0038	DBSB1	DMA buffer B start address 1.
0h B000 003C	DBTB1	DMA buffer B transfer count 1.
0h B000 0040	DDAR2	DMA device address register 2.

Physical Address	Symbol	Register Name
0h B000 0044	DCSR2	DMA control/status register 2 – write ones to set.
0h B000 0048		Write ones to clear.
0h B000 004C		Read only.
0h B000 0050	DBSA2	DMA buffer A start address 2.
0h B000 0054	DBTA2	DMA buffer A transfer count 2.
0h B000 0058	DBSB2	DMA buffer B start address 2.
0h B000 005C	DBTB2	DMA buffer B transfer count 2.
0h B000 0060	DDAR3	DMA device address register 3.
0h B000 0064	DCSR3	DMA control/status register 3 – write ones to set.
0h B000 0068		Write ones to clear.
0h B000 006C		Read only.
0h B000 0070	DBSA3	DMA buffer A start address 3.
0h B000 0074	DBTA3	DMA buffer A transfer count 3.
0h B000 0078	DBSB3	DMA buffer B start address 3.
0h B000 007C	DBTB3	DMA buffer B transfer count 3.
0h B000 0080	DDAR4	DMA device address register 4.
0h B000 0084	DCSR4	DMA control/status register 4 – write ones to set.
0h B000 0088		Write ones to clear.
0h B000 008C		Read only.
0h B000 0090	DBSA4	DMA buffer A start address 4.
0h B000 0094	DBTA4	DMA buffer A transfer count 4.
0h B000 0098	DBSB4	DMA buffer B start address 4.
0h B000 009C	DBTB4	DMA buffer B transfer count 4.
0h B000 00A0	DDAR5	DMA device address register 5.
0h B000 00A4	DCSR5	DMA control/status register 5 – write ones to set.
0h B000 00A8		Write ones to clear.
0h B000 00AC		Read only.
0h B000 00B0	DBSA5	DMA buffer A start address 5.
0h B000 00B4	DBTA5	DMA buffer A transfer count 5.
0h B000 00B8	DBSB5	DMA buffer B start address 5.
0h B000 00BC	DBTB5	DMA buffer B transfer count 5.

Physical Address	Symbol	Register Name
LCD Controller Registers		
0hB010 0000	LCCR0	LCD controller control register 0.
0hB010 0004	LCSR	LCD controller status register.
0hB010 0008 – 0hB010 000C	—	Reserved.
0hB010 0010	DBAR1	DMA channel 1 base address register.
0hB010 0014	DCAR1	DMA channel 1 current address register.
0hB010 0018	DBAR2	DMA channel 2 base address register.
0hB010 001C	DCAR2	DMA channel 2 current address register.
0hB010 0020	LCCR1	LCD controller control register 1.
0hB010 0024	LCCR2	LCD controller control register 2.
0hB010 0028	LCCR3	LCD controller control register 3.
0hB010 002C – 0hB010 FFFF	—	Reserved.
UDC Registers (Serial Port 0)		
0h8000 0000	UDCCR	UDC control register.
0h8000 0004	UDCAR	UDC address register.
0h8000 0008	UCDOMP	UDC OUT max packet register.
0h8000 000C	UDCIMP	UDC IN max packet register.
0h8000 0010	UDCCS0	UDC endpoint 0 control/status register.
0h8000 0014	UDCCS1	UDC endpoint 1 (out) control/status register.
0h8000 0018	UDCCS2	UDC endpoint 2 (in) control/status register.
0h8000 001C	UDCD0	UDC endpoint 0 data register.
0h8000 0020	UDCWC	UDC endpoint 0 write count register.
0h8000 0024	—	Reserved.
0h8000 0028	UDCDR	UDC transmit/receive data register (FIFOs).
0h8000 002C	—	Reserved.
0h8000 0030	UDCSR	UDC status/interrupt register.
UART Registers (Serial Port 1)		
0h 8001 0000	UTCR0	UART control register 0.
0h 8001 0004	UTCR1	UART control register 1.
0h 8001 0008	UTCR2	UART control register 2.
0h 8001 000C	UTCR3	UART control register 3.
0h 8001 0010	—	Reserved.
0h 8001 0014	UTDR	UART data register.
0h 8001 0018	—	Reserved.
0h 8001 001C	UTSR0	UART status register 0.
0h 8001 0020	UTSR1	UART status register 1.
0h 8001 0024 – 0h 8001 FFFF	—	Reserved.

Physical Address	Symbol	Register Name
SDLC Registers (Serial Port 1)		
0h 8002 0060	SDCR0	SDLC control register 0.
0h 8002 0064	SDCR1	SDLC control register 1.
0h 8002 0068	SDCR2	SDLC control register 2.
0h 8002 006C	SDCR3	SDLC control register 3.
0h 8002 0070	SDCR4	SDLC control register 4.
0h 8002 0074	—	Reserved.
0h 8002 0078	SDDR	SDLC data register.
0h 8002 007C	—	Reserved.
0h 8002 0080	SDSR0	SDLC status register 0.
0h 8002 0084	SDSR1	SDLC status register 1.
0h 8002 0088 – 0h 8002 FFFF	—	Reserved.
ICP – UART Registers (Serial Port 2)		
0h 8003 0000	UTCR0	UART control register 0.
0h 8003 0004	UTCR1	UART control register 1.
0h 8031 0008	UTCR2	UART control register 2.
0h 8003 000C	UTCR3	UART control register 3.
0h 8003 0010	UTCR4	UART control register 4.
0h 8003 0014	UTDR	UART data register.
0h 8003 0018	—	Reserved.
0h 8003 001C	UTSR0	UART status register 0.
0h 8003 0020	UTSR1	UART status register 1.
0h 8003 0024 – 0h 8003 FFFF	—	Reserved.
ICP – HSSP Registers (Serial Port 2)		
0h 8004 0060	HSCR0	HSSP control register 0.
0h 8004 0064	HSCR1	HSSP control register 1.
0h 8004 0068	—	Reserved.
0h 8004 006C	HSDR	HSSP data register.
0h 8004 0070	—	Reserved.
0h 8004 0074	HSSR0	HSSP status register 0.
0h 8004 0078	HSSR1	HSSP status register 1.
0h 8004 007C – 0h 8004 FFFF	—	Reserved.

Physical Address	Symbol	Register Name
UART Registers (Serial Port 3)		
0h 8005 0000	UTCR0	UART control register 0.
0h 8005 0004	UTCR1	UART control register 1.
0h 8005 0008	UTCR2	UART control register 2.
0h 8005 000C	UTCR3	UART control register 3.
0h 8005 0010	—	Reserved.
0h 8005 0014	UTDR	UART data register.
0h 8005 0018	—	Reserved.
0h 8005 001C	UTSR0	UART status register 0.
0h 8005 0020	UTSR1	UART status register 1.
0h 8005 0024 – 0h 8005 FFFF	—	Reserved.
MCP Registers (Serial Port 4)		
0h 8006 0000	MCCR0	MCP control register 0.
0h 8006 0004	—	Reserved.
0h 8006 0008	MCDR0	MCP data register 0.
0h 8006 000C	MCDR1	MCP data register 1.
0h 8006 0010	MCDR2	MCP data register 2.
0h 8006 0014	—	Reserved.
0h 8006 0018	MCSR	MCP status register.
0h 8006 001C – 0h 8006 005C	—	Reserved.
SSP Registers (Serial Port 4)		
0h 8007 0060	SSCR0	SSP control register 0.
0h 8007 0064	SSCR1	SSP control register 1.
0h 8007 0068	—	Reserved.
0h 8007 006C	SSDR	SSP data register.
0h 8007 0070	—	Reserved.
0h 8007 0074	SSSR	SSP status register.
0h 8007 0078 – 0h 8007 FFFF	—	Reserved.
PPC Registers		
0h 9006 0000	PPDR	PPC pin direction register.
0h 9006 0004	PPSR	PPC pin state register.
0h 9006 0008	PPAR	PPC pin assignment register.
0h 9006 000C	PSDR	PPC sleep mode direction register.
0h 9006 0010	PPFR	PPC pin flag register.
0h 9006 0030	MCCR1	MCP control register 1.
0h 9006 0034 – 0h 9006 FFFF	—	Reserved.

3.6864–MHz Oscillator Specifications B

A 3.6864-MHz crystal oscillator is integrated on the Intel® StrongARM® SA-1100 Microprocessor (SA-1100) for use as a reference frequency for the PLLs that generate the internal clocks to the processor. The phase noise of this reference frequency should be minimized because it could be amplified by the PLLs, resulting in PLL output frequency jitter. For this application, the long-term stability and the temperature effect on the frequency are not important because they affect the frequency by less than 1%. The oscillator circuit is designed to work across a range of crystal parameters so that the system designer can choose from several 3.6864-MHz crystals available on the market. In normal operation, the pins of the crystal, Q1 and Q2, are connected to the SA-1100 pins, PXTAL and PEXTAL. Note that a 3.5795-MHz crystal can also be used, but in order to meet the frequency specifications of several of the integrated I/O ports, a 3.6864-MHz crystal is required.

In some applications, it may be desirable to provide the 3.6864-MHz reference from an external signal source. This option is supported by the SA-1100. See Chapter 8, “Clocks”.

B.1 Specifications

This section includes specifications for the oscillator circuit and the quartz crystal.

B.1.1 System Specifications

This section includes the specifications of the oscillator circuit. It assumes that the crystal used meets the specifications given in the following sections.

Temperature Range

This is the junction temperature range for the oscillator circuit on the SA-1100. The crystal itself may be at the ambient temperature; the oscillator circuit integrated on the SA-1100 is most likely operating at a higher temperature that is dependent on the activity of the SA-1100.

Current Consumption

Because this oscillator might run during the sleep mode of the processor, the power consumption is critical. The specified current consumption is for the oscillator only. The power associated with the oscillator output buffer is not included because this buffer is powered down in sleep.

Startup Time

This specification depends on the crystal characteristics and the layout of the printed circuit board (PCB). The value given assumes that the crystal and board layout conform to the values given in the remainder of this document. The critical parameters in the crystal specification are the shunt capacitance (Co) and the motional resistance (Rm), which must be no greater than the maximums specified. The critical parameters in the PCB layout are the parasitic capacitances between PXTAL and PEXTAL, and between either of these nodes and VSS. Note that in some applications, such as a system that includes a socketed SA-1100, it may be difficult to meet the parasitic capacitances specified. While the 3.6864-MHz oscillator will start with parasitic capacitances, which are

approximately twice the values given, the startup time in this situation will be about double the specified startup time and the current consumption will increase. Capacitances larger than twice the specified values may prevent the oscillator from starting.

B.1.1.1. Parasitic Capacitance Off-chip Between PXTAL and PEXTAL

The parasitic capacitance off-chip between PXTAL and PEXTAL is the board capacitance between the PXTAL and PEXTAL pins.

B.1.1.2. Parasitic Capacitance Off-chip Between PXTAL or PEXTAL and VSS

The parasitic capacitance off-chip between PXTAL or PEXTAL and VSS is the parasitic board capacitance between the PXTAL or PEXTAL pins and the VSS wire surrounding the crystal connections.

B.1.1.3. Parasitic Resistance Between PXTAL and PEXTAL

The parasitic resistance between PXTAL and PEXTAL is the parasitic resistance between the PXTAL and PEXTAL pins due to moisture and other effects.

B.1.1.4. Parasitic Resistance Between PXTAL or PEXTAL and VSS

The parasitic resistance between PXTAL or PEXTAL and VSS is the parasitic resistance between the PXTAL or PEXTAL pins to VSS due to moisture and other effects.

The following table describes the system specifications of the oscillator circuit.

Specification	Minimum	Typical	Maximum	Unit
Temperature range	0		100	^{o}C
Supply voltage	3	3.3	3.6	V
Ripple voltage on the supply	—	—	0.3	V
Current consumption	—	15	40	µA
Startup time	—	15	150	ms
Parasitic capacitance off-chip between PXTAL and PEXTAL	—	—	1	pF
Parasitic capacitance off-chip between PXTAL or PEXTAL and VSS	—	—	2	pF
Parasitic resistance between PXTAL or PEXTAL to VSS	1	—	—	MΩ
Parasitic resistance between PXTAL and PEXTAL	1	—	—	MΩ

B.1.2 Quartz Crystal Specification

The following specifications for the quartz crystal are shown in the figure and table below.

Resonance frequency (**fs**): Resonance frequency of the crystal.

Motional capacitance (**Cm**): Equivalent serial capacitance in the crystal model.

Motional inductance (**Lm**): Not generally given in supplier specification.

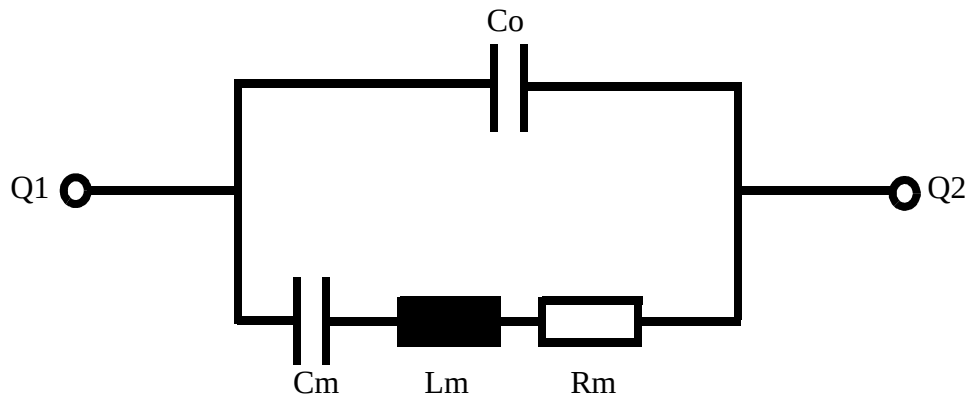
Motional resistance (**Rm**): Equivalent serial resistance in the crystal model. Some crystal providers refer to this resistance as the Equivalent Series Resistance (ESR) or simply Series Resistance.

Shunt capacitance (**Co**): Parasitic capacitance between Q1 and Q2.

Load capacitance (**CL**): Needed load capacitance viewed by the crystal to oscillate at fs.

Drive level: Power dissipated in the equivalent serial resistance (Rm).

Aging: Resonance frequency shift due to aging.



Specification	Minimum	Typical	Maximum	Unit
Resonance frequency (fs)	3.5795	3.6864	—	MHz
Motional resistance (Rm)	40	180	300	W
Shunt capacitance (Co)	—	—	7	pF
Drive level	—	—	10	μW
Crystal type	AT cut crystal			

32.768–kHz Oscillator Specifications C

A 32.768-kHz crystal oscillator is integrated on the Intel® StrongARM® SA-1100 Microprocessor (SA-1100) for use as a time base for the real-time clock (RTC). The output frequency of the crystal oscillator is divided by 32768 (2^{15}) to deliver a 1-Hz signal to the RTC. A digital tuning circuit is included on the SA-1100 in order to calibrate the 1-Hz output for each crystal and circuit based on a set of values stored in an external EEPROM. The oscillator circuit is designed to work across a range of crystal parameters so that the system designer can choose from several 32.768-kHz crystals available on the market. In normal operation, the pins of the crystal, Q1 and Q2, are connected to the SA-1100 pins, TXTAL and TEXTAL.

In some applications, it may be desirable to provide the 32.768-kHz reference from an external signal source. This option is supported by the SA-1100. See the Chapter 8, “Clocks”.

C.1 Specifications

This section includes specifications for the oscillator circuit and the quartz crystal.

C.1.1 System Specifications

This section includes the specifications of the oscillator circuit. It assumes that the crystal used meets the specifications given in the following sections.

C.1.1.1. Temperature Range

This is the junction temperature range for the oscillator circuit on the SA-1100. The crystal itself may be at the ambient temperature; the oscillator circuit integrated on the SA-1100 is most likely operating at a higher temperature that is dependent on the activity of the SA-1100.

C.1.1.2. Current Consumption

Because this oscillator runs during the sleep mode of the processor, the power consumption is critical. The specified current consumption is for the oscillator and its output buffer only. The power of the tuning circuit and RTC is not included in the value specified.

C.1.1.3. Startup Time

This specification depends on the crystal characteristics and the layout of the printed circuit board (PCB). The value given assumes that the crystal and board layout conform to the values given in the remainder of this document. The critical parameters in the crystal specification are the shunt capacitance (Co) and the motional resistance (Rm), which must be no greater than the maximums specified. The critical parameters in the PCB layout are the parasitic capacitances between TXTAL and TEXTAL, and between either of these nodes and VSS. Note that in some applications, such as a system that includes a socketed SA-1100, it may be difficult to meet the parasitic capacitances specified. While the 32.768-kHz oscillator will start with parasitic capacitances which are

approximately twice the values given; the startup time in this situation will be about double the specified startup time and the current consumption will increase. Capacitances larger than twice the specified values may prevent the oscillator from starting.

C.1.1.4. Frequency Shift Due to Temperature Effect on the Circuit

The frequency shift due to temperature effect on the circuit is the influence of the oscillator circuit on the frequency of oscillation due to temperature effect. The appropriate temperature range is the junction temperature on the SA-1100, not the ambient temperature. Note that this specification *does not include either* the temperature effects on the quartz ***or*** the aging of the crystal. It includes the temperature effect of the circuit only. The frequency shift of the crystal itself due to temperature may be significantly larger than that of the oscillator circuit. However, for a long-term stability calculation, it may be appropriate to consider the average temperature of the crystal rather than the extreme values of temperature.

C.1.1.5. Parasitic Capacitance Off-chip Between TXTAL and TEXTAL

The parasitic capacitance off-chip between TXTAL and TEXTAL is the board capacitance between the TXTAL and TEXTAL pins.

C.1.1.6. Parasitic Capacitance Off-chip Between TXTAL or TEXTAL and VSS

The parasitic capacitance off-chip between TXTAL or TEXTAL and VSS is the parasitic board capacitance between the TXTAL or TEXTAL pins and the VSS wire surrounding the crystal connections.

C.1.1.7. Parasitic Resistance Between TXTAL and TEXTAL

The parasitic resistance between TXTAL and TEXTAL is the parasitic resistance between the TXTAL and TEXTAL pins due to moisture and other effects.

C.1.1.8. Parasitic Resistance Between TXTAL or TEXTAL and VSS

The parasitic resistance between TXTAL or TEXTAL and VSS is the parasitic resistance between the TXTAL or TEXTAL pins to VSS due to moisture and other effects.

The following table describes the specifications of the oscillator circuit.

Specification	Minimum	Typical	Maximum	Unit
Temperature range	0		100	^{0}C
Supply voltage	3	3.3	3.6	V
Ripple voltage on the supply	—	—	0.3	V
Current consumption	—	1	2	µA
Startup time	—	—	2	s
Frequency shift due to temperature effect on the circuit	—	—	+/-3	ppm
Parasitic capacitance off-chip between TXTAL and TEXTAL	—	—	1	pF

Parasitic capacitance off-chip between TXTAL or TEXTAL and VSS	—	—	2	pF
Parasitic resistance between TXTAL or TEXTAL to VSS	10	—	—	MΩ
Parasitic resistance between TXTAL and TEXTAL	10	—	—	MΩ

C.1.2 Quartz Crystal Specification

The following specifications for the quartz crystal are shown in the figure and table below.

Resonance frequency (fs): Resonance frequency of the crystal.

Motional capacitance (Cm): Equivalent serial capacitance in the crystal model.

Motional inductance (Lm): Not generally given in supplier specification.

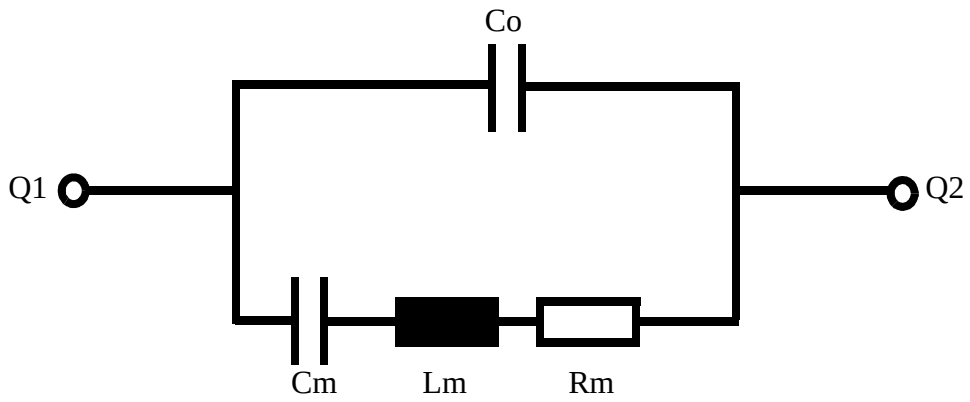
Motional resistance (Rm): Equivalent serial resistance in the crystal model. Some crystal providers refer to this resistance as the Equivalent Series Resistance (ESR) or simply Series Resistance. Other providers supply a Quality Factor, Q, instead of Rm; therefore, the values for Q corresponding to specified range of Rm are supplied in the following table.

Shunt capacitance (Co): Parasitic capacitance between Q1 and Q2.

Load capacitance (CL): Needed load capacitance viewed by the crystal to oscillate at fs.

Drive level: Power dissipated in the equivalent serial resistance (Rm).

Aging: Resonance frequency shift due to aging.



Specification	Minimum	Typical	Maximum	Unit
Resonance frequency (fs)	—	32768	—	Hz
Quality factor (Q)	40K	80K	200K	—
Motional capacitance (Cm)	2	3	4	fF
Motional resistance (Rm)	—	—	50K	W
Shunt capacitance (Co)	0.9	—	2	pF

Load capacitance (CL)	10	12.5	25	pF
Drive level	—	—	1	μW
Crystal type	Tuning fork (X+5^{o} or X+2^{o} cut)			

The following values are not required for the crystal oscillator to function, but they directly affect the performance of the oscillator in the system because they determine the accuracy of the crystal itself. The values given represent those seen on typical crystals used for timekeeping, and are provided for information only.

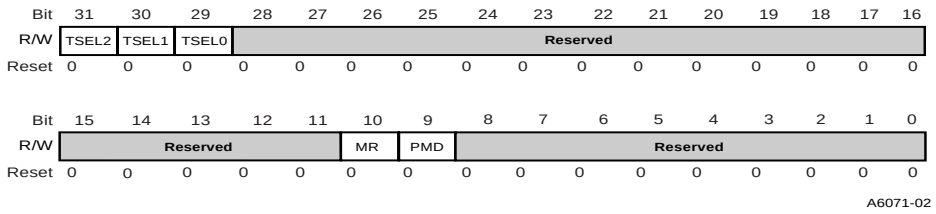
Specification	**Minimum**	**Typical**	**Maximum**	**Unit**
Frequency tolerance	+/-5	+/-20	+/-30	ppm
Parabolic curvature	—	-0.042	-0.05	ppm/ ^{o}C
Turnover temperature	20	25	30	^{o}C
Temperature range	0	—	60	^{o}C
Aging	—	+/-3	+/-5	ppm/year

Internal Test D

The Test Unit contains a register that enables certain test modes. Some of these test modes are reserved for manufacturing test and should not be invoked by an end user.

D.1 Test Unit Control Register (TUCR)

The Test Unit Control Register (TUCR) contains control bits that put the Intel® StrongARM® SA-1100 Microprocessor (SA-1100) in various test modes. It is recommended that the operating system write protect these registers under normal conditions to prevent them from being inadvertently written. The following figure shows the format of this register. At reset reserved bits are zero. Writing reserved bits to one can lead to UNPREDICTABLE results.



Bit	Name	Description
0..5	Reserved	—
6	Reserved	—
7	Reserved	—
8	Reserved	—
9	PMD	Power management disable. When PMD is set, sleep mode is disabled and the SA-1100 ignores the ForceSleep bit, as well as the BATT_FAULT and VDD_Fault pins. This bit is cleared on hard reset.
10	MR	Memory request mode. Controls two GPIO pins used for external arbitration and for the memory bus. 0 – GP<21> and GP<22> are not used for an alternate function. 1 – GP<21> and GP<22> are reserved for use as MBREQ and MBGNT.
11..19	Reserved	—
20	Reserved	—
21	Reserved	—
22	Reserved	—
23	Reserved	—
24	Reserved	—
25	Reserved	—
26	Reserved	—

Bit	Name	Description
27..28	Reserved	—
29..31	TSEL2-0	Test selects. Routes internal signals out onto GPIO<27> for observing internal clock signals. To observe these clocks, set bit 27 to one in the GAFR and GPDR registers and set the TSEL bits to the following settings to select which clock is driven onto GP<27>:

TSEL2	TSEL1	TSEL0	GP<27>(alternate function)
0	0	0	32-kHz oscillator
0	0	1	3.6864-MHz oscillator
0	1	0	VDD ring oscillator/16
0	1	1	96-MHz PLL/4
1	0	0	32-kHz oscillator (also enable rclk on GP<26>
1	0	1	3.6864-MHz oscillator
1	1	0	Main PLL/16
1	1	1	VDDL ring oscillator/4

Support, Products, and Documentation

If you need general information or support, call **1-800-628-8686** or visit Intel's website at:

http://www.intel.com

Copies of documents that have an ordering number and are referenced in this document, a product catalog, or other Intel literature may be obtained by calling **1-800-548-4725** or by visiting Intel's website for developers at:

http://developer.intel.com