SECTION 6
STATIC RAM

The on-chip static RAM and its functionality are described in this section.

6.1 SRAM FEATURES

- 4 KByte-SRAM, organized as 1024 x 32 bits
- Single-cycle access
- Physically located on processor's high-speed local bus
- Memory location programmable on any 32 KByte address
- Byte, word, longword address capabilities

6.2 SRAM OPERATION

The SRAM module provides a general-purpose memory block that the ColdFire processor can access in a single cycle. The location of the memory block can be specified to any 0-modulo-32K address within the 4-GByte address space. The memory is ideal for storing critical code or data structures or for use as the system stack. Because the SRAM module is physically connected to the processor's high-speed local bus, it can service processor-initiated access or memory-referencing commands from the debug module.

Depending on configuration information, instruction fetches and operand reads may be sent to both the unified cache and the SRAM block simultaneously. If the reference is mapped into the region defined by the SRAM, the SRAM provides the data back to the processor, and the cache data discarded. Accesses from the SRAM module are not cached.

Also note that the SRAM cannot be accessed by the on-chip DMAs of the MCF5307. Figure 1-1, the MCF5307 system block diagram, clarifies this concept. The on-chip system is configured to allow concurrent core and DMA execution, where the core can execute code out of internal SRAM and/or cache during DMA access.

6.3 SRAM PROGRAMMING MODEL

The SRAM programming model includes a description of the SRAM base address register (SRAMBAR), SRAM initialization, and power management.

6.3.1 SRAM Base Address Register (RAMBAR)

The configuration information in the SRAM Base Address Register (RAMBAR) controls the operation of the SRAM module.

- The RAMBAR is the register that holds the base address of the SRAM. The MOVEC instruction provides write-only access to this register.
- The RAMBAR register can be read or written from the Debug module in a similar manner.
- All undefined bits in the register are reserved. These bits are ignored during writes to the RAMBAR, and return zeroes when read from the debug module.
- The RAMBAR valid bit is cleared by reset, disabling the SRAM module. All other bits are unaffected.

The RAMBAR register contains four control fields. These fields are detailed in the following.

SRAM Base Address Register (RAMBAR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BA31	BA30	BA29	BA28	BA27	BA26	BA25	BA24	BA23	BA22	BA21	BA20	BA19	BA18	BA17	BA16
RESET: —	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA15	—	—	—	—	—	—	WP	—	—	C/I	SC	SD	UC	UD	V
RESET: —	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0

Figure 6-1. SRAM Base Address Register (RAMBAR)

BA[31:15] - Base Address

This field defines the 0-modulo-32K base address of the SRAM module. The SRAM memory occupies a 4KByte space defined by the contents of the Base Address field. By programming this field, the SRAM may be located on any 32KByte boundary within the processor's four gigabyte address space.

WP - Write Protect

This field allows only read accesses to the SRAM. When this bit is set, any attempted write access will generate an access error exception to the ColdFire processor core.

0 = Allows read and write accesses to the SRAM module
1 = Allows only read accesses to the SRAM module

C/I, SC, SD, UC, UD - Address Space Masks (ASn)

This five bit field allows certain types of accesses to be “masked,” or inhibited from accessing the SRAM module. The address space mask bits are:

- C/I = CPU space/interrupt acknowledge cycle mask
- SC = Supervisor code address space mask
- SD = Supervisor data address space mask
- UC = User code address space mask
- UD = User data address space mask

For each address space bit:

0 = An access to the SRAM module can occur for this address space
1 = Disable this address space from the SRAM module. If a reference using this address space is made, it is inhibited from accessing the SRAM module, and is processed like any other non-SRAM reference.

These bits are useful for power management as detailed in Section 6.3.4.

V - Valid

The valid bit (V-bit) is specified by RAMBAR[0]. A hardware reset clears this bit. When set, this bit enables the SRAM module; otherwise, the module is disabled.

0 = Contents of RAMBAR are not valid
1 = Contents of RAMBAR are valid

6.3.2 SRAM Initialization

After a hardware reset, the contents of the SRAM module are undefined. The valid bit of the RAMBAR is cleared, disabling the module. If the SRAM requires initialization with instructions or data, the following steps should be performed:

1. Load the RAMBAR mapping the SRAM module to the desired location within the address space.
2. Read the source data and write it to the SRAM. There are various instructions to support this function, including memory-to-memory move instructions, or the MOVEM opcode. The MOVEM instruction is optimized to generate line-sized burst fetches on 0-modulo-16 addresses, so this opcode generally provides maximum performance.
3. After the data has been loaded into the SRAM, it may be appropriate to load a revised value into the RAMBAR with a new set of attributes These attributes consist of the write-protect and address space mask fields.

The ColdFire processor or an external emulator using the debug module can perform these initialization functions.

6.3.3 SRAM Initialization Code

The code segment below describes how to initialize the SRAM. The code sets the base address of the SRAM at $20000000 and then initializes the RAM to zeros.

```
RAMBASE     EQU   $20000000         ;set this variable to $20000000
RAMVALID    EQU   $00000000
move.l      #RAMBASE+RAMVALID,D0    ;load RAMBASE + valid bit into D0.
movec.l     D0, RAMBAR              ;load RAMBAR and enable SRAM

The following loop initializes the entire SRAM to zero

lea.l       RAMBASE,A0              ;load pointer to SRAM
move.l      #1024,D0                ;load loop counter into D0

SRAM_INIT_LOOP:
clr.l       (A0)+                   ;clear 4 bytes of SRAM
subq.l      #1,D0                   ;decrement loop counter
bne.b       SRAM_INIT_LOOP          ;if done, then exit; else continue looping
```

6.3.4 Power Management

As noted previously, depending on the configuration defined by the RAMBAR, instruction fetch and operand read accesses may be sent to the SRAM and unified cache simultaneously. If the access is mapped to the SRAM module, it sources the read data, and the unified cache access is discarded. If the SRAM is used only for data operands, asserting the ASn bits associated with instruction fetches can decrease power dissipation. Additionally, if the SRAM contains only instructions, masking operand accesses can reduce power dissipation.

Consider the examples in Table 6-1 of typical RAMBAR settings:

Table 6-1. Examples of Typical RAMBAR Settings

DATA CONTAINED IN SRAM	RAMBAR[7:0]
Code Only	$2B
Data Only	$35
Both Code And Data	$21