Freescale Semiconductor
Application Note

Document Number: AN3514
Rev. 0, 09/2007

ColdFire Serial Boot Facility

by: Juan Mendoza
Microcontroller Division

Contents

1 Introduction 1
1.1 Serial Boot Facility. 2
1.2 External Interface 3
1.3 Memory Map 3
2 Functional Description. 4
2.1 Serial Initialization and Shift Clock Frequency Adjustment. 4
2.2 Reset Configuration and Optional Boot Load 5
2.3 Execution Transfer 6
2.4 FAST_READ Feature Initialization 6
3 Example. 7

1 Introduction

Today’s complex, highly-integrated processors make it impractical to dedicate or to share pins for the numerous available power-up options. The serial boot facility (SBF) solves this problem by giving you the ability to store and load all device reset configuration data and user code from an external SPI memory.

ColdFire® microprocessors offer different boot modes determined by configuration pin(s). When the processor uses the SBF, it can read data from a broad array of SPI-compatible EEPROMs, flashes, and FRAMs. The memory devices have a simple command set consisting of one byte opcodes that indicate the operation to be performed.

1.1 Serial Boot Facility

The SBF interfaces to an external SPI memory to read configuration data and boot code during the processor reset sequence requiring only minimal I/O pins. By reading data stored in the SPI memory, the SBF configures the SPI memory clock frequency setting, sets all configurable power-up options for the processor, and optionally loads code into the processor memory space. Through interaction with the reset controller, the SBF accomplishes all of this before the device’s reset negates, ensuring the chip is properly configured when exiting the reset state.

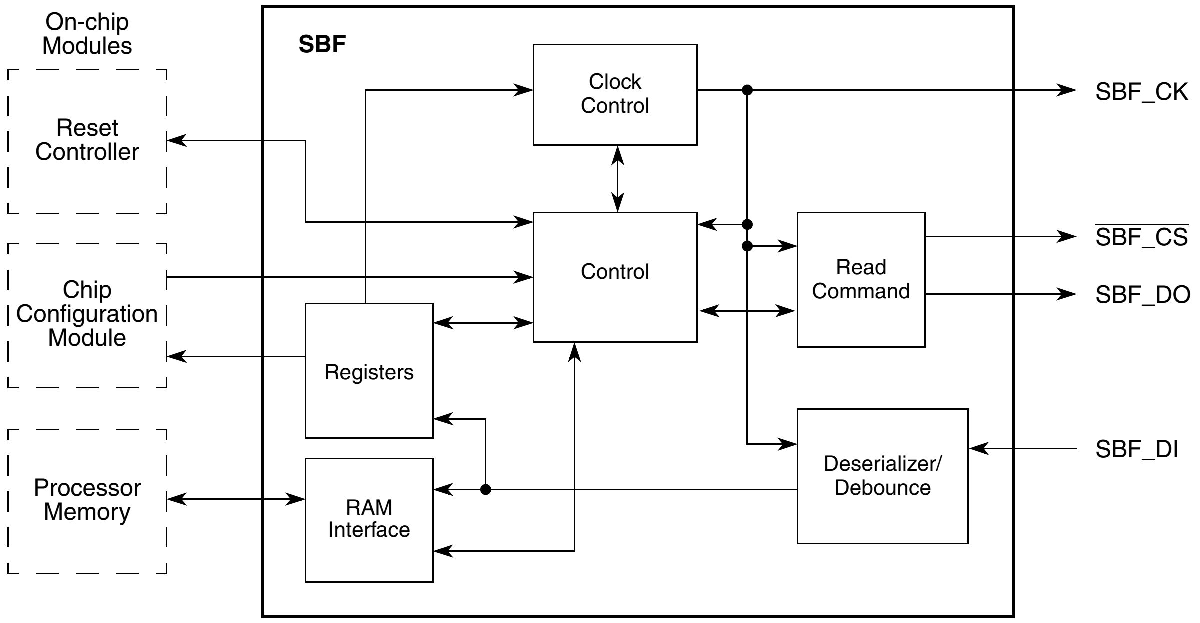

Figure 1. SBF Block Diagram

1.1.1 SBF Features

The SBF includes these distinctive features.

- Support for various different SPI memory devices
  - EEPROM
  - Flash
  - FRAM
  - Embedded FPGA memory
- External interface maps directly to and can be multiplexed with the DMA serial peripheral interface (DSPI) pins
- Self-adjusting shift clock frequency for maximum throughput supported by SPI memory
- Optionally load boot code into the processor’s memory space

1.2 External Interface

The SBF uses a four-wire interface.

Table 1. Signal Descriptions

Signal	I/O	Description	Reset	Pull Up
SBF_CK	O	Shift clock. Alternate edges of this signal cause the SPI memory to accept data from and drive data to the processor	—	—
$\overline{\text{SBF\_CS}}$	O	Chip select. This signal enables the SPI memory and places it into an active state, ready to accept commands.	—	—
SBF_DI	I	Data in. The SPI memory drives and the processor accepts read data on this signal.	—	Active[1]
SBF_DO	O	Data out. The SBF drives the read command and address on this signal. **Note:** The SBF does not write data to the SPI memory.	—	—

[1] Disabled by the SBF when the SPI memory begins shifting out data.

1.3 Memory Map

The serial boot facility programming model consists of two registers: the SBF status register (SBFSR) and the SBF control register (SBFCR).

1.3.1 SBFSR

The SBFSR is read-only and reflects the amount of boot code loaded through the external SPI memory.

Address: 0xFC0A_0018 (SBFSR) Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BLL															
W																
Reset	See Note															

Note: Reset value is user-defined (loaded from SPI memory during serial boot following any reset type)

Figure 2. Serial Boot Facility Status Register (SBFSR)

1.3.2 SBFCR

The SBFCR is read-always/write-once and controls the SBF operation. SBFCR[15:5] are reserved bits and should be cleared. SBFCR[FR] determines whether standard READ command (cleared) or flash FAST_READ command (set) is used following any reset other than power-on reset. SBFCR[BLDIV] represents the boot loader clock divider, the SBF clock divisor that generates serial shift clock output on SBF_CK. This value is loaded during the serial boot sequence with the value read from the SPI memory. Because this register is write-once, FR and BLDIV values must be written at the same time.

Address: 0xFC0A_0020 (SBFCR) Access: User read/write-once

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	FR	BLDIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0[1]	See Note			

[1] Reset value is 0 and is reset only by power-on reset (remains unchanged for other reset types)

Note: Reset value is user-defined (loaded from SPI memory during serial boot following power-on reset, remains unchanged for other reset types)

Figure 3. Serial Boot Facility Control Register (SBFCR)

2 Functional Description

When enabled, the SBF inserts three additional steps into the normal system boot process:

- Serial initialization and shift clock frequency adjustment
- Reset configuration and optional boot load
- Execution transfer

2.1 Serial Initialization and Shift Clock Frequency Adjustment

The SBF operates during device reset. The following sequence is followed during serial boot:

1. The SBF is engaged with the release of a pending source of reset (power-on, software watchdog, $\overline{\text{RESET}}$ pin, etc).
2. Boot-up is paused.
3. The weak internal pull-up on SBF_DI is enabled. This allows a 1-to-0 transition to register when the SPI memory output switches from high-impedance to logic 0.
4. The SBF shifts the standard SPI memory read command (0x03) followed by repeated 0x00 address bytes to the SPI memory at $f_{REF} \div 67$.
5. After the SPI memory accepts however many shift clock edges are necessary to respond to the READ command, it turns on its previously tri-stated output and begins driving the msb of the byte at address 0.
   Bits [7:4] of this byte must be 0000 so that the required 1-to-0 transition can be detected on SBF_DI to synchronize the SBF state machine. If bits [7:4] of this byte are not 0000, bits[3:0] are ignored, another byte is clocked out of the SPI memory (SBF_DO remains at logic 0) and the SBF state machine again tests for a 1-to-0 transition followed by four consecutive zero bits.
6. After the necessary 1-to-0 transition and reception of a byte with bits [7:4] equal to 0000, the SBF pauses and bits [3:0] of the received byte select a new shift clock divider according to the below table.

   The values for the divisors were chosen because they represent divisor values that generate common SPI memory bit rates when the PLL input reference (SBF clock) frequency is 25, 33, 33.33, 50, or 66 MHz. The result is an optimal shift clock frequency for the attached SPI memory.

Table 2. SBF Divisor Bit Settings

BLDIV	Ideal Divisor	Shift Clock		BLDIV	Ideal Divisor	Shift Clock	
		High Time (f_{ref} Ticks)	Low Time (f_{ref} Ticks)			High Time (f_{ref} Ticks)	Low Time (f_{ref} Ticks)
0000	1	Bypass	Bypass	1000	14	7	7
0001	2	1	1	1001	17	9	8
0010	3	2	1	1010	25	13	12
0011	4	2	2	1011	33	17	16
0100	5	3	2	1100	34	17	17
0101	7	4	3	1101	50	25	25
0110	10	5	5	1110	67	34	33
0111	13	7	6	1111	Reserved		

7. The weak internal pull-up on SBF_DI is disabled.
8. The shift clock begins toggling at the new frequency, resuming the READ command already in progress.

NOTE

Shift clock frequency adjustment follows a power-on/hard reset only. After the new divisor is known, it is stored in the sticky SBFCR[BLDIV] field and used for subsequent soft resets. This speeds reboot for systems that do not benefit from the optional FAST_READ on soft reset feature (e.g., the SPI memory does not support FAST_READ, or the input reference clock does not exceed the maximum allowable frequency for the READ command).

2.2 Reset Configuration and Optional Boot Load

After the steps in Section 2, "Functional Description", are executed, the following is performed to load configuration data and optional boot code.

1. The SBF shifts two bytes (16 bits) out of the SPI memory that indicate how many longwords, if any, are read during the optional boot load sequence. These bytes are software-visible in the SBFSR[BLL] field.
2. The read operation continues reading the reset configuration data. The format and length of the data varies per device.
3. At this point, the SBF determines whether or not to read boot code. If SBFSR[BLL] is not zero, BLL plus one longwords (4 × (BLL + 1) bytes) are consecutively loaded into the SRAM.

NOTE

Although the SBF permits loading up to 65,536 longwords (262,144 bytes), the maximum practical number that can be read is limited by the size of the processor's internal SRAM.

2.3 Execution Transfer

After boot load is complete or if no boot load is requested (SBFSR[BLL] = 0), the following steps complete the serial boot process:

1. The acquired configuration data is driven to the appropriate modules.
2. The system is released from reset.
3. The ColdFire processor initiates its normal reset vector fetch at address 0.
4. The actual memory that responds to the reset vector fetch depends on whether serial boot load is requested:
   - If SBFSR[BLL] is cleared, the reset vector fetch is handled by the FlexBus module and whatever external memory is mapped at address 0, governed by the user-provided setting of CCR[FBCONFIG].
   - If SBFSR[BLL] is not zero, the reset vector and boot code are read from the on-chip SRAM (The SBF enables the SRAM and maps it to address 0 via the RAMBAR before control of the processor is restored to the ColdFire core). The reset vector (initial stack pointer and program counter) should point to locations in the on-chip SRAM, so that boot code can initialize the device and load the application software from the SPI memory or via some other mechanism (e.g. a hard disk drive connected to the ATAPI controller or a network server responding to a TFTP client).

2.4 FAST_READ Feature Initialization

Many SPI flash memories implement a FAST_READ command which allows for a substantially higher shift-clock frequency. The SBF always uses the normal READ command when coming out of a power-on/hard reset. However, when coming out of a soft reset, it is possible to use FAST_READ because the SBF machine state is not lost.

The SBFCR[FR] sticky bit can be set, causing the FAST_READ command to be issued instead of the READ command in the event of a soft reset. To enable the FAST_READ feature, set the SBFCR[FR] bit in the same write that sets the SBFCR[BLDIV] field. The value written to SBFCR[BLDIV] should correspond to the frequency the SPI memory supports in FAST_READ mode. After a soft reset, the SBFCR[BLDIV] field is not overwritten with the value read from the SPI memory. Instead, the SBF uses the value written by the user to SBFCR[BLDIV] prior to soft reset to generate the SPI memory clock.

NOTE

The ability to use the FAST_READ command is limited by the SBF electrical specifications. Specifically, delays present throughout the system (including those between the SBF, the pin multiplexing logic, and the actual I/O pads) limit the maximum frequency at which the SBF operates and can preclude use of the FAST_READ feature altogether. Even when the delays within the processor itself are minimized, the actual SPI memories may have similarly untenable electrical specifications (data input setup and output valid times).

3 Example

The following example uses the ColdFire MCF54455 microprocessor. The SBF requires that, prior to device power-up, the SPI memory is loaded with data organized according to Table 3. See section “Reset Configuration (BOOTMOD[1:0] = 11)” of the *MCF54455 Reference Manual* for the reset configuration (RCON) data definition.

Table 3. SPI Memory Organization

Byte Address	Data Contents
0x0	{0000,BLDIV[3:0]}
0x1	BLL[7:0]
0x2	BLL[15:8]
0x3	RCON[7:0]
0x4	RCON[15:8]
...	...
0x12	RCON[127:120]
0x13[1]	CODE_BYTE_0
0x14	CODE_BYTE_1
...	...
0x12 + 4 x (BLL + 1)	CODE_BYTE_[4x(BLL+1) -1]

[1] This assumes BLL does not equal zero. If BLL equals 0, the SBF does not access data at this or following addresses.

You can load the data into the SPI memory by generating the opcodes for the bootcode:

```
uint32 sbf_code[] =
{
    0x80001000, 0x80000008, 0x203C8000, 0x00000680,
    0x00000221, 0x4E7B0C05, 0x203C07FF, 0x000123C0,
    0xFC008004, 0x203C0800, 0x000023C0, 0xFC008018,
    0x203C0000, 0x014023C0, 0xFC008020, 0x203C0000,
    0x000123C0, 0xFC00801C, 0x4E71203C, 0x00200000,
    0x123C00FF, 0x13C10800, 0x00054E71, 0x538066FA,
    0x203C0020, 0x0000123C, 0x000013C1, 0x08000005,
    0x4E715380, 0x66FA60D0, };
```

and by defining a structure that contains the configuration information required by the microprocessor:

```
uint8 sbf_config[] =
{
        0x03,                      /* BLDIV = 3 (CLKIN / 4) */
        (((sizeof(sbf_code)/4)-1) & 0x00FF),        /* BLL[7:0] */
        (((sizeof(sbf_code)/4)-1) & 0xFF00) >> 8,   /* BLL[15:8] */
        0x34, 0x12,                /* RCON[15:0], PCI Subsystem Vendor ID = 0x1234 */
        0x78, 0x56,                /* RCON[31:16], PCI Subsystem ID = 0x5678 */
        0x00,                      /* RCON[39:32], PCI Revision ID = 0x0 */
        0x00, 0x80, 0x06,          /* RCON[63:40], PCI Class Code = 0x068000 */
        0x57, 0x19,       /* RCON[79:64], PCI Vendor ID */
        0x07, 0x58,       /* RCON[95:80], PCI Device ID */
        0xFF,             /* RCON[103:96], PCI configurations */
```

```
                0x00,               /* RCON[111:104], enable PLL */
                8-1,                /* RCON[119:112], PLL multiplier *(x8) */
                0x98                /* RCON[127:120], 8-bit FB, muxed, oscillator bypass, BME */
         };
```

The structure sbf_code[] is comprised of the opcodes generated by the compiler for the following application boot code:

```
/*********************************************************************
 *
 * Exception Vector Table
 */
VECTOR_TABLE:
_VECTOR_TABLE:
INITSP:     .long   __SRAM + 0x1000                   /* Initial SP           */
INITPC:     .long   asm_startmeup                     /* Initial PC           */

asm_startmeup:
_asm_startmeup:
    /* Initialize 32KByte SRAM */
    move.l  #__SRAM,d0
    add.l   #0x221,d0
    movec   D0,RAMBAR1
    /* Initialize the system */
    move.l  #0x07FF0001,d0
    move.l  d0,0xFC008004   /* CSMR0 */
    move.l  #0x08000000,d0
    move.l  d0,0xFC008018   /* CSAR3 */
    move.l  #0x00000140,d0
    move.l  d0,0xFC008020   /* CSCR3 */
    move.l  #0x00000001,d0
    move.l  d0,0xFC00801C   /* CSMR3 */
loop1:
    nop
    move.l  #0x00200000,d0
    move.b  #0xFF,d1
    move.b  d1,0x08000005
wait1:
    nop
    subq.l  #1,d0
    bne     wait1
loop2:
    move.l  #0x00200000,d0
    move.b  #0x00,d1
    move.b  d1,0x08000005
wait2:
    nop
    subq.l  #1,d0
    bne     wait2
    bra     loop1

/*********************************************************************/
.end
```

The compiled application code is linked using the following linker directive file:

```
#/*
# * File:       sram.lcf Linker file targeting on-chip SRAM
# */
MEMORY {sram    (RX) : ORIGIN = 0x80000000,  LENGTH = 0x8000}

SECTIONS
{
    ___FLASH1                = 0x00000000;
    ___FLASH1_SIZE           = (32 * 1024 * 1024);

    ___FLASH0                = 0x04000000;
    ___FLASH0_SIZE           = (512 * 1024);

    ___CPLD                  = 0x08000000;
    ___CPLD_SIZE             = (16 * 1024 * 1024);

    ___FPGA                  = 0x09000000;
    ___FPGA_SIZE             = (16 * 1024 * 1024);

    ___SDRAM                 = 0x40000000;
    ___SDRAM_SIZE            = (256 * 1024 * 1024);

    ___SRAM                  = 0x80000000;
    ___SRAM_SIZE             = (32 * 1024);

    ___VECTOR_RAM            = ___SRAM;

.text :{ *(.text) } > sram
```

The compiled code and configuration information is then written to the SPI memory:

```
SPI FLASH DATA:
03 1D 00 34 12 78 56 00 00 80 06 57 19 07 58 FF
00 07 98 80 00 10 00 80 00 00 08 20 3C 80 00 00
00 06 80 00 00 02 21 4E 7B 0C 05 20 3C 07 FF 00
01 23 C0 FC 00 80 04 20 3C 08 00 00 00 23 C0 FC
00 80 18 20 3C 00 00 01 40 23 C0 FC 00 80 20 20
3C 00 00 00 01 23 C0 FC 00 80 1C 4E 71 20 3C 00
20 00 00 12 3C 00 FF 13 C1 08 00 00 05 4E 71 53
80 66 FA 20 3C 00 20 00 00 12 3C 00 00 13 C1 08
00 00 05 4E 71 53 80 66 FA 60 D0
```

where the first 19 bytes represent {0000,BLDIV[3:0]}, BLL[15:0], and RCON[127:0]. The remaining bytes represent the boot code to be loaded of length 0x1D + 1 longwords (120 bytes), as reported by BLL[15:0].

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3514
Rev. 0
09/2007