MCF5373 Chip Errata

Silicon Revision: All

This document identifies implementation differences between the MCF537*x* processors and the description contained in the *MCF5373ColdFire® Reference Manual*. Refer to http://www.freescale.com/coldfire for the latest updates.

Early devices are marked as M29B mask set. The date code on the marking can be used to determine which errata have been corrected on a particular device as shown in Table 1. The datecode format is XXXYYWW, where YY represents the year and WW represents the work week. The three leading digits can be ignored. Newer devices are marked with a mask set number including a revision.

Table 1. Summary of MCF537x Errata

Errata	Module Affected	Date Errata Added	Revision Affected?						
			< XXX0617	≥ XXX0617	3M29B	4M29B < XXX0821	4M29B ≥ XXX0821	5M29B	6M29B
SECF022	Clock	11/9/05	Yes	Yes	No	No	No	No	No
SECF011	SRAM	11/9/05	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SECF047	USB	12/14/05	Yes	No	No	No	No	No	No
SECF010	FEC	1/23/06	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SECF017	FlexBus	1/23/06	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SECF013	Core	5/4/06	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SECF012	Core	5/4/06	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SECF045	SDRAMC	6/14/06	Yes	Yes	No	No	No	No	No
SECF008	FlexBus	6/14/06	Yes	Yes	No	No	No	No	No
SECF049	USB	6/14/06	Yes	Yes	No	No	No	No	No

Table continues on the next page...

Table 1. Summary of MCF537x Errata (continued)

Errata	Module Affected	Date Errata Added	Revision Affected?						
			< XXX0617	≥ XXX0617	3M29B	4M29B < XXX0821	4M29B ≥ XXX0821	5M29B	6M29B
SECF039	Reset	1/23/07	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SECF014	Debug	3/20/08	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SECF130	SDRAMC	7/18/08	Yes	Yes	Yes	Yes	No	No	No
SECF147	SSI	6/25/09	Yes	Yes	Yes	Yes	Yes	No	No
SECF149	SDRAM	8/19/09	Yes	Yes	Yes	Yes	Yes	No	No
SECF180	INTC	12/01/10	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SECF196	Pins	04/06/11	Yes	Yes	Yes	Yes	Yes	Yes	No

The chip identification register (CIR) can also be used to determine the silicon revision. Table 2 list the CIR[PRN] field values that correspond to given datecodes and mask sets.

Table 2. CIR[PRN] to mask

CIR[PRN] value	Mask set
0	< XXX0617
1	≥ XXX0617
2	3M29B and 4M29B
5	5M29B
6	6M29B

The table below provides a revision history for this document.

Table 3. Document Revision History

Rev. No.	Date	Substantive Changes
1	11/2005	Initial revision
1.1	12/2005	Added SECF047
1.2	2/2006	Corrected title in Table 1 Added "Possible Cache Corruption After Clearing Cache (Setting CACR[CINV])" and "Incorrect Operation of Cache Freeze (CACR[CFRZ])" which were later removed in a subsequent revision Added SECF010 and SECF017

Table continues on the next page...

Table 3. Document Revision History (continued)

Rev. No.	Date	Substantive Changes
1.3	6/2006	Added information for datecode XXX0618 and greater that includes a fix for errata #4. Added CIR[PRN] decode information for different silicon revisions in Table 2. Added SECF013, SECF012, , SECF045, SECF008, and SECF049 Removed errata which apply to V2 ColdFire core devices and should not have been placed in this document: "Possible Cache Corruption After Clearing Cache (Setting CACR[CINV])" and "Incorrect operation of Cache Freeze (CACR[CFRZ])"
1.4	10/2006	Updated errata to reflect fixed bugs for the 3M29B and 4M29B mask sets.
1.5	1/2007	Added SECF039
2	2/2007	Thorough editing sweep of document for grammar, punctuation, and stylistic changes.
3	11/2007	Added MCF53721 to list of affected devices.
4	3/2008	Added SECF014
5	7/2008	Added SECF130
6	6/2009	Added SECF147
7	8/2009	Added SECF149
8	8/2010	Added SECF180
9	12/2010	Added 5M29B mask set
10	04/2011	Added SECF196
11	09/2011	Added 6M29B mask set

SECF022: PLL Behavior in Stop Mode

Errata type: Silicon
Affects: PLL
Description: If the PLL is disabled in stop mode, the device exits stop mode with its booted frequency regardless of the operating frequency prior to entering stop mode. Entering stop mode with an LPCR[STPMD] setting of 0b10 or 0b11 causes a hard reset to the PLL module. When stop mode is exited, the PLL re-latches the reset configuration setting for the PLL. For example:

1. The device is booted with a 240/80 MHz frequency selected.
2. The device is placed into limp mode to change the operating frequency to 180/60 MHz and limp mode is exited.
3. The LPCR is programmed to disable the PLL in stop mode (LPCR = 0xD0 or 0xD8).
4. The stop instruction is executed.
5. An interrupt is generated to exit the processor from stop mode.
6. The device exits stop mode at an operating frequency of 240/80 MHz instead of 240/80180/60 MHz.

Workaround: If the PLL frequency has not been modified since the last reset or if limp mode is being used, no workaround is needed.

If the PLL was reprogrammed after reset, the PLL needs to be reprogrammed after exiting stop mode. The following steps show how this can be done:

1. Program the LPCR register to disable the PLL in stop mode (LPCR = 0xD0 or 0xD8).
2. Save the values of the PLL registers.

3. Place the device into limp mode by clearing the MISCCR[LIMP] bit.
4. Execute the STOP instruction.
5. After exiting stop mode, reload the PLL with the register values saved in step 2.
6. Exit limp mode by clearing the MISCCR[LIMP] bit.

Fix plan: Fixed starting with 3M29B mask set.

SECF044: SDRAMC Incorrect Operation After Exiting Limp Mode

Errata type: Silicon
Affects: SDRAM controller
Description: The circuitry that controls the SDRAMC's SD_DQS signals attempts to lock to the clock when the device is in limp mode. The large difference in clock speeds between limp mode and normal operation causes the SD_DQS logic to become unsynchronized when limp mode is exited.

Workaround: After exiting limp mode, the value of 0x4000_0000 should be written to address 0xFC0B_8080 before attempting to initialize the SDRAMC or exit the SDRAM from self-refresh mode.

Fix plan: Fixed starting with 3M29B mask set.

SECF011: RAMBAR Cannot be read Through BDM

Errata type: Silicon
Affects: Core
Description: There is an error in the read path for the RAMBAR register that prevents the register from being read through the BDM module. When the RAMBAR is read, it reads back 0x00000_0000. The write path for the register works correctly; therefore, the RAMBAR can be written via BDM or the MOVEC instruction.

Workaround: No workaround.

Fix plan: Currently, there are no plans to fix this.

SECF047: USB On-Chip Transceivers do not Operate Correctly in Full-Speed Mode

Errata type: Silicon
Affects: USB
Description: There is an error in the integration of the USB on-chip full-speed (FS)/low-speed (LS) transceivers that prevents them from operating correctly in FS mode. The transceivers can be used in LS mode.

Workaround:
- For the USB host module, use the on-chip transceiver in LS mode.
- For the USB on-the-go module, use an external ULPI transceiver instead of the on-chip transceiver.

Fix plan: Fixed on datecodes XXX0618 and later.

SECF010: FEC Interrupts will not Trigger on Consecutive Transmit Frames

Errata type: Silicon
Affects: FEC
Description: The late collision (LC), retry limit (RL), and underrun (UN) interrupts do not trigger on consecutive transmit frames. For example, if back-to-back frames cause a transmit underrun, only the first frame generates an underrun interrupt. No other underrun interrupts are generated until a frame is transmitted that does not underrun or the FEC is reset.

Workaround: Because late collision, retry limit, and underrun errors are not directly correlated to a specific transmit frame, in most cases a workaround for this problem is not needed. If a workaround is required, there are two independent workarounds:

- Ensure that a correct frame is transmitted after a late collision, retry limit, or underrun errors are detected.
- Perform a soft reset of the FEC by setting ECR[RESET] when a late collision, retry limit, or underrun errors are detected.

Fix plan: Currently, there are no plans to fix this.

SECF017: No Bus Monitor or Watchdog Recovery for Hung FlexBus Accesses

Errata type: Silicon
Affects: FlexBus
Description: This device does not include a bus monitor or watchdog timer capable of forcing the termination of a hung FlexBus access. There are two possible cases that could lead to a hung FlexBus access:

- If an access to one of the FlexBus memory areas (0x0000_0000–0x3FFF_FFFF or 0xC000_0000–0xDFFF_FFFF) does not hit in the valid range for one of the chip selects, a 32-bit access with no chip select asserted and no internal acknowledge is generated. The resulting bus cycle continues indefinitely waiting for the assertion of the TA signal (external acknowledge).
- If a chip select is configured for external termination, but no termination is received or a timing problem prevents TA from being recognized, the bus cycle is hung and continues indefinitely.

There is no on-chip fail-safe to exit from the hung bus state. The system needs to assert TA to terminate the hung cycle or reset the entire device.

Workaround: Avoid issuing non-terminated FlexBus accesses. Ideally, a complete system does not access unused memory areas. This is harder to achieve while debugging, but as long as the debugger has access to the TA signal on the BDM header, it should be able to force termination of any hung bus cycles.

Workaround: Use an external bus monitor circuit that can detect hung bus cycles and generate a TA to terminate the access. The bus monitor circuit can optionally assert an external interrupt signal along with TA to indicate the error condition back to the CPU.

Workaround: If there are unused chip selects in the system, they can be mapped over unused memory areas. The only purpose of these chip selects would be to terminate accesses to addresses that are not used. BDM address breakpoints can then be used to generate an interrupt error.

Fix plan: Currenty, there are no plans to fix this.

SECF013: Incorrect operation of STLDSR instruction

Errata type: Silicon
Affects: Core
Description: This device supports the ISA_A+ instruction set. ISA_A+ includes the STLDSR (store load status register) instruction, but this instruction does not work correctly in the V3 core.

Workaround: Do not use the STLDSR instruction.

Fix plan: Currently, there are no plans to fix this.

SECF012: Possible Stack Pointer Corruption when Using Supervisor and User Stack Pointers

Errata type: Silicon
Affects: Core
Description: The ColdFire V3 core architecture includes support for two independent stack pointer registers: a user stack pointer and a supervisor stack pointer. The switching of the stack pointers during the execution af an RTE instruction may cause a return to an incorrect address.

Workaround: Do not enable the user stack pointer (keep CACR[EUSP] cleared).

Workaround: Where the target of the RTE is cacheable or in internal SRAM, the following specific code sequence touches the cache line that is the destination of the RTE instruction. This allows for correct timing in the core logic to RTE correctly.

```
mov.l 4(%a7), %an   ## where n is 0-6
                    tst.l %a0           ## touches cache line
                    movm  to restore registers
                    rte
```

Fix plan: Currently, there are no plans to fix this.

SECF045: Potential Boot Failure Using 32-Bit Wide SDRAM

Errata type: Silicon
Affects: SDRAM controller
Description: If the SD_CKE signal to the SDRAM deasserts while one or more banks are active, the SDRAM enters a clock suspend state. If the SDRAM was driving the data lines before entering the clock suspend state, the buffers continue to drive.

During a reset, the processor deasserts SD_CKE without any graceful stop period to ensure that the SDRAM banks are all in an IDLE state. In some cases, the SDRAM could be driving the data bus during and immediately after reset. This can lead to possible bus contention while latching reset configuration (RCON) values and/or while reading boot code from flash, and that could cause the processor to enter an undefined state.

Workaround: Use a split bus mode configuration with DDR SDRAM or SDR SDRAM. This creates a dedicated 16-bit port for the SDRAM on D[31:16]. In this configuration, the SDRAM does not share data lines with other devices, so bus contention is not an issue.

Workaround: Additional workarounds TBD. We are investigating workarounds that can be used for 32-bit wide SDRAM configurations.

Fix plan: Fixed starting with 3M29B mask set.

SECF008: Programmable Address Hold Does Not Function Correctly

Errata type: Silicon
Affects: FlexBus
Description: The programmable address hold feature for the FlexBus chip selects does not function correctly. The address is held for one clock after the chip select deasserts regardless of the address hold value programmed in the CSCR[WRAH] or CSCR[RDAH] fields. The address hold fields creates a delay at the end of the bus cycle. They can continue to be used to ensure a minimum delay between bus cycles.

Workaround: No workarounds.

Fix plan: Fixed starting with 3M29B mask set.

SECF049: Change to Operation of USBHOST_VBUS_OC Signal

Errata type: Silicon
Affects: USB
Description: The USBHOST_VBUS_OC is an active high input used to indicate to the processor that a short has occurred on the USB data bus. To allow for more flexibility, a control bit is added to future silicon revisions that allows programmable polarity of the USBHOST_VBUS_OC signal.

The control bit is added to the burst configuration register (BCR) of the system control module.

The upper two bytes of the BCR register is shown below:

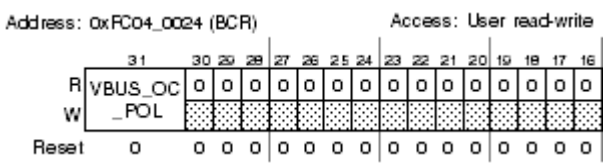

Figure 1. Burst Configuration Register (BCR)

The VBUS_OC_POL field description is given below. The rest of the bits in the BCR remain unchanged.

Table 4. BCR Field Definitions

Field	Description
31 VBUS_OC_POL	USBHOST_VBUS_OC polarity. Allows for programmable polarity of the USBHOST_VBUS_OC signal. 0 Active high 1 Active low

Workaround: No workaround is needed. Designs using existing silicon are not adversely impacted by this change.

Fix plan: Fixed starting with 3M29B mask set.

SECF039: Potential Reset Hang Booting into Limp Mode

Errata type: Silicon
Affects: Reset controller
Description: When running in limp mode with a non-default clock divide value (CDR[LPDIV] > 0), if an external reset occurs, the reset might hang the processsor if RCON is used to boot directly into limp mode. The reason for this is that the internal clock logic that generates the 3:1 clock ratio becomes out of sync when the asychronous reset occurs.

This problem does not occur if the default limp mode divider is used (CDR[LPDIV] = 0).

Workaround: Do not boot directly into limp mode.

Workaround: If using RCON to boot directly into limp mode, use only a low power clock divider of 1 (CDR[LPDIV] = 0).

Fix plan: Currently, there are no plans to fix this.

SECF014: Level 2 Trigger Operation Controlled by TDR[31]

Errata type: Silicon
Affects: BDM
Description: The TDR[L2T] bit (TDR bit 15) has no effect on the level 2 trigger. Bit 31 of the TDR register provides both trigger response control and logical operation of the level 2 trigger.

Workaround: Use the TDR[31] bit to control the logical operation for the level 2 trigger as follows:

- 0 -- Level 2 trigger = PC_condition & Address_range & Data_condition
- 1 -- Level 2 trigger = PC_condition | (Address_range & Data_condition)

Since TDR[31] is also part of the trigger response control, only certain combinations of trigger responses and logical operations are available as shown below:

Table 5. TDR[31:30] Definitions

TDR[31:30]	Level 2 Trigger	Trigger Response
00	PC_cond & (Add_range & Data_cond)	Display on DDATA
01		Processor halt
10	PC_cond | (Add_range & Data_cond)	Debug interrupt
11		Reserved

Fix plan: Currently, there are no plans to fix this.

SECF130: Non-Matching SDVDD and EVDD Voltages

Errata type: Silicon
Affects: SDRAMC
Description: Due to a packaging error, there is an internal connection between the EVDD and SDVDD voltage domains. EVDD always has a nominal value of 3.3V, but SDVDD can be 3.3V, 2.5V, or 1.8V. If a 2.5V or 1.8V nominal SDVDD is used, then the connection to the 3.3V EVDD domain raises the SDVDD domain above the applied voltage.

Workaround: Use the same 3.3V nominal voltage supply for the EVDD and SDVDD domains.

Fix plan: Fixed on datecodes XXX0821 and higher.

SECF147: Incorrect Operation of SSI Clock Divider

Errata type: Silicon
Affects: CCM and SSI
Description: When the CCM's MISCCR[SSISRC] bit is set, the SSI oversampling clock (SSI_MCLK) is generated by passing the system clock (f_{sys}) through a fractional divider (CDR[SSIDIV]). This divider does not work correctly with a high-speed system clock. The clock glitches and causes the SSI to see extra clock edges. The clock behavior can vary from device to device and can also vary across temperature and voltage on a given part.

Workaround: Use an external clock source for the SSI on the SSI_CLKIN pin.

Workaround: Use the SSI in slave mode. In this case, the SSI_MCLK is supplied externally and the output of the SSI clock divider is not used.

Workaround: Run the system clock (f_{sys}) at 180MHz. At this frequency the SSI clock divider works correctly.

Fix plan: Fixed starting with 5M29B mask set.

SECF149: SD_CLK delay when using FlexBus

Errata type: Silicon
Affects: SDRAMC
Description: When performing FlexBus cycles, SD_CLK and $\overline{\text{SD\_CLK}}$ are delayed by up to 2.34 ns in the worst case scenario, resulting in duty cycle excursions that may exceed DDR memory specifications for clock jitter. Though no system problems have been found while using DDR at this time, a potential problem may exist in some systems due to this issue.

Workaround: Use low-power DDR or SDR SDRAM.

Fix plan: Fixed starting with 5M29B mask set.

SECF180: Spurious Interrupts Can Cause Incorrect Vector Fetch

Errata type: Silicon
Affects: INTC

Description: In rare cases the interrupt controller's spurious detection logic can cause a fetch to an incorrect vector number. This can occur when the core is starting the IACK for a spurious interrupt. During this small window of time, if a second interrupt at a different level arrives, the second interrupt causes the interrupt controller logic to clear the spurious request. Therefore, the interrupt controller sees no valid interrupt pending at the requested level and returns vector number 0 for INTC0 or vector number 64 for INTC1.

The second interrupt can be at any level other than the level that caused the spurious interrupt (it can even be a lower priority than the spurious interrupt). If the second interrupt is at the same level as the spurious interrupt, then the correct vector number for the second interrupt is returned.

Workaround: In many systems spurious interrupts represent error conditions in and of themselves. So, it is always a good design practice to eliminate potential causes of spurious interrupts during product development. Proper interrupt management can help to prevent or reduce the possibility of spurious interrupts (and the potential occurrence of this errata). The correct procedure for masking an interrupt in the INTC or inside the module is:

1. Write the interrupt level mask in the core's status register (SR[I]) to a value higher than the priority level of the interrupt you want to mask.
2. Mask the interrupt using the INTC's IMR and/or an interrupt mask register inside the module.
3. Write the original value back to the core's status register.

Even when steps are taken to remove spurious interrupts, it is still desirable to have a spurious interrupt handler to help manage unexpected events and glitches in a system. A workaround to allow for correct spurious interrupt handling is to:

1. After boot, copy the vector table to RAM
2. Modify the vector 0 and vector 64 entries so that they point to the spurious interrupt handler.

This way the system performs the same for any potential spurious interrupt vectors. Vectors 0, 64, and 24 (the correct spurious interrupt vector) should point to the same handler.

Fix plan: No plans to fix.

SECF196: Pins: Undefined Pin States Caused by Power-On Reset Sequence

Errata type: Silicon
Affects: FlexBus, SDRAMC, and BDM
Description: Some power ramp and clock startup sequences can cause FlexBus, SDRAMC, and BDM pins to drive undefined values for a period of time during the reset sequence.

Initially the pins are tri-stated while IVDD and EVDD/SDVDD are not fully powered. As the voltage rails ramp to the operating levels, the processor releases the pins from tri-state and the pin states are determined by digital control logic. However, the digital logic needs several clock cycles to initialize, so between tri-state release and clock startup the pins can be driven by uninitialized logic.

Depending on how the processor signals are used in the rest of the system, an undefined pin state could cause unexpected behavior for an external memory device. Since the undefined pin states are temporary (pins will enter a defined state well before the processor exits reset) in many cases the undefined pin states won't cause a functional issue for the system.

Workaround: The undefined pin states occur only during a power-on reset (POR), the $\overline{\text{RESET}}$ and/or $\overline{\text{RSTOUT}}$ signals can be used to qualify FlexBus and SDRAMC signals. In most cases, the only signals that will need to be qualified are the FlexBus chip select and SDRAM chip select signals. By using qualified versions of the chip selects to connect to external devices, you can prevent those external devices from sampling the undefined pin states on address, data, and other control signals.

The first step to the workaround is to generate a signal that can be used to qualify or mask the undefined signal states on the chip select signals (or other control signals as needed). Determining the qualifier to use can require some characterization of power and reset signal operation in the system. The idea is to find a signal or combination of signals that remain high around the time when the undefined signal states can occur.

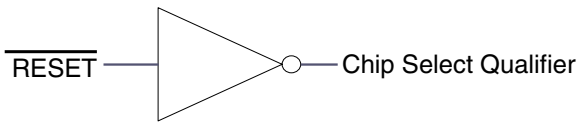

Figure 2. Generating a qualifier signal that can be used if $\overline{\text{RESET}}$ remains asserted until internal clocks start

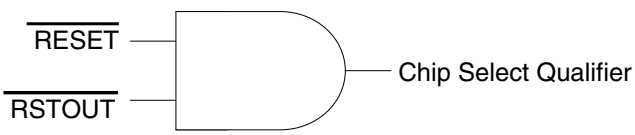

Figure 3. Generating a qualifier signal if $\overline{\text{RESET}}$ only remains asserted until $\overline{\text{RSTOUT}}$ drives low

The second step of the workaround is to use the qualifier signal generated to qualify or mask chip select signals to external devices as needed.

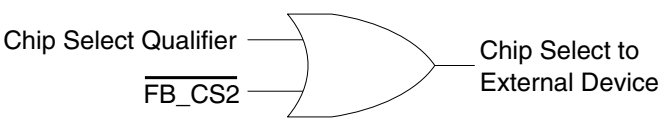

Figure 4. Qualifying a single chip select using an OR gate

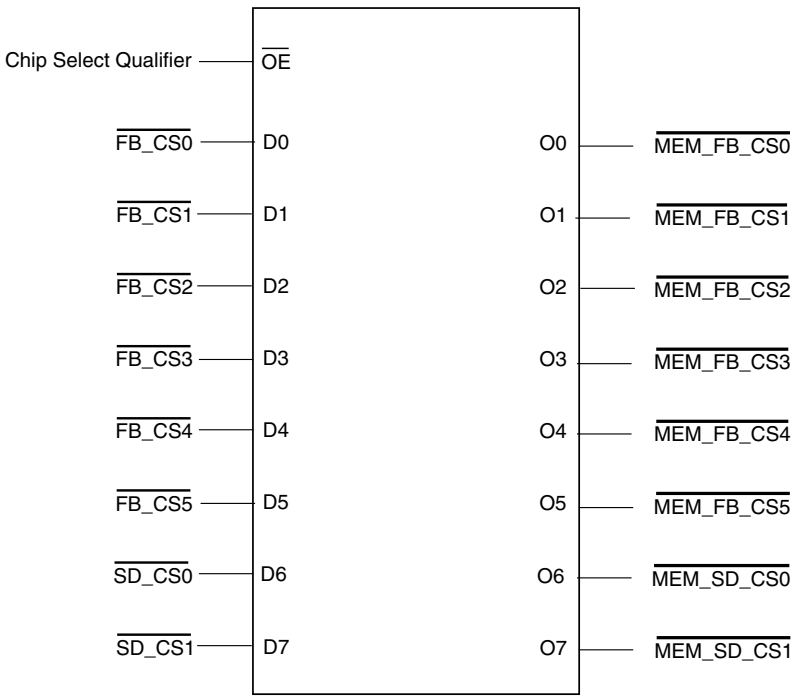

Figure 5. Qualifying multiple chip selects using a buffer

In the diagram above “MEM_” is used to differentiate the chip select signals from the processor and the qualified versions that would be connected to external memories. Each of the “MEM_” signals should include a pullup resistor to force the signal high while the buffer is disabled.

Fix plan: This issue has been fixed starting with the 6M29B mask set. The fix that has been implemented ensures that $\overline{\text{SD\_CS}}$[1:0] and $\overline{\text{FB\_CS}}$[2:0] will either drive high or float during reset. These pins will never driver low during the reset sequence. The operation of $\overline{\text{FB\_CS}}$[5:3] is unchanged. The state of $\overline{\text{FB\_CS}}$[5:3] can still be undefined during the reset sequence.

How to Reach Us:

Home Page:
www.freescale.com

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

USA/Europe or Locations Not Listed:
Freescale Semiconductor
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 China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: MCF5373DE
Rev. 11, 09/2011

freescale semiconductor