MCF5208 Chip Errata

Silicon Revision: All

Supports: MCF5207 and MCF5208

Summary of MCF520*x* Errata

Early MCF5208 devices are marked as M28B mask set. The latest version of the MCF520*x* is marked as 2M28B.

Table 1. Summary of MCF520*x* Errata

Errata	Module Affected	Date Errata Added	Revision Affected?	
			M28B	2M28B
SECF022	Clock	8/11/05	Yes	No
SECF044	SDRAMC	10/4/05	Yes	No
SECF005	Cache	1/23/06	Yes	Yes
SECF001	Cache	1/23/06	Yes	Yes
SECF010	FEC	1/23/06	Yes	Yes
SECF017	FlexBus	1/23/06	Yes	Yes
SECF045	SDRAMC	6/14/06	Yes	No
SECF008	FlexBus	6/14/06	Yes	No
SECF014	Debug	3/20/08	Yes	Yes
SECF180	Interrupt Controller	8/12/10	Yes	Yes

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 mask set.

Table 2. CIR[PRN] to mask

CIR[PRN] value	Mask set
0	M28B
2	2M28B

Revision History

The table below provides a revision history for this document.

Table 3. Document Revision History

Rev. No.	Date	Substantive Changes
1	8/2005	Initial revision
1.1	10/2005	Added SECF044
1.2	2/2006	Added SECF005, SECF001, SECF010, and SECF017
1.3	6/2006	Added SECF045 and SECF008
1.4	10/2006	Updated errata to reflect fixed bugs for the 2M28B mask set.
2	3/2008	Added SECF014
3	8/2010	Added SECF180

SECF001: Incorrect Operation of Cache Freeze (CACR[CFRZ])

Errata type: Silicon

Affected component: Version 2 ColdFire Cache

Description: The cache on the V2 ColdFire core is controlled by the cache control register (CACR). When the CACR[CFRZ] bit is set, the cache freeze function is enabled and no valid cache array entry is displaced. However, this feature does not always work as specified, sometimes allowing valid lines to be displaced when CACR[CFRZ] is enabled.

This does not cause any corrupted accesses. However, there could be cache misses for data that was originally loaded into the cache but was subsequently deallocated, even though the CACR[CFRZ] bit was set.

Also, incoherent cache states are possible when a frozen cache is cleared via the CACR[CINV] bit.

Workaround: Unfreeze the cache by clearing CACR[CFRZ] when invalidating the cache using the CACR[CINV] bit

Workaround: Use the internal SRAM to store critical code/data if the system cannot handle a potential cache miss

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

SECF005: Possible Cache Corruption After Clearing Cache (Setting CACR[CINV])

Errata type: Silicon

Affected component: Version 2 ColdFire Cache

Description: The cache on the V2 ColdFire core may function as either:

- a unified data and instruction cache

- an instruction cache
- a data cache

The cache function and organization is controlled by the cache control register (CACR). The CACR[CINV] bit causes a cache clear. If the cache is configured as a unified cache and the CINV bit is set, the scope of the cache clear is controlled by two other bits in the CACR:

- CACR[INVI] invalidates instruction cache only
- CACR[INVD] invalidates data cache only

If a write to the CACR is performed to clear the cache (CACR[CINV] = 1) and only a partial clear is done (CACR[INVI] or CACR[INVD] set), then cache corruption may occur.

Workaround: All loads of the CACR that perform a cache clear operation (CACR[CINV] set) should be followed immediately by a NOP instruction. This avoids the cache corruption problem.

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

SECF008: Programmable Address Hold Does Not Function Correctly

Errata type: Silicon

Affected component: 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 2M28B mask set.

SECF010: FEC Interrupts will not Trigger on Consecutive Transmit Frames

Errata type: Silicon

Affected component: 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.

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

Errata type: Silicon

Affected component: 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 4. 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.

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

Errata type: Silicon

Affected component: 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.

SECF022: PLL Behavior in Stop Mode

Errata type: Silicon

Affected component: 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 166.67/83.33 MHz frequency selected.
2. The device is placed into limp mode to change the operating frequency to 144/72 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 166.67/83.33144/72 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 2M28B mask set.

SECF044: SDRAMC Incorrect Operation After Exiting Limp Mode

Errata type: Silicon

Affected component: 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 2M28B mask set.

SECF045: Potential Boot Failure Using 32-Bit Wide SDRAM

Errata type: Silicon

Affected component: 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 2M28B mask set.

SECF180: Spurious Interrupts Can Cause Incorrect Vector Fetch

Errata type: Silicon

Affected component: 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.

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 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 and 24 (the correct spurious interrupt vector) should point to the same handler.

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

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: MCF5208DE
Rev. 3, 8/2010