MCF52277 Chip Errata

Silicon Revision: All

Supports: MCF52274, MCF52277

Introduction

This document identifies implementation differences between the MCF5227*x* processors and the description contained in the *MCF52277 ColdFire® Reference Manual*. Refer to http://www.freescale.com/coldfire for the latest updates.

Summary of MCF5227*x* Errata

The latest version of the MCF5227x family is revision 1.3.

Table 1. Summary of MCF5227*x* Errata

Errata	Module Affected	Date Errata Added	Revision Affected?		
			Rev 1.1	Rev 1.2	Rev 1.3
SECF001	Cache	11/02/07	Yes	Yes	Yes
SECF005	Cache	11/02/07	Yes	Yes	Yes
SECF032	LCD	11/02/07	Yes	No	No
SECF046	Touchscreen	11/02/07	Yes	No	No
SECF040	RTC	11/02/07	Yes	No	No
SECF033	PLL	1/22/08	Yes	No	No
SECF014	BDM	3/18/08	Yes	Yes	Yes
SECF121	LCD	11/17/08	Yes	Yes	No
SECF122	SBF	4/21/09	Yes	Yes	Yes
SECF126	LCD	4/21/09	Yes	Yes	Yes
SECF150	SDRAM	8/27/09	Yes	Yes	Yes
SECF162	SBF and reset	4/28/10	Yes	Yes	Yes
SECF178	GPIO and LCD	6/24/10	Yes	Yes	Yes

Errata	Module Affected	Date Errata Added	Revision Affected?		
			Rev 1.1	Rev 1.2	Rev 1.3
SECF180	Interrupt Controller	8/12/10	Yes	Yes	Yes

Use the table below to determine the revision using the part number and mask set markings on the device.

Table 2. Part Number to Revision

Orderable Part Number	Mask Set	Revision
PCF52274CLU120	M26H	Rev 1.1
MCF52274CLU120	2M26H	Rev 1.2
MCF52274CLU120	3M26H	Rev 1.3
PCF52277CVM160	M26H	Rev 1.1
MCF52277CVM160	M26H	Rev 1.1
SC52277F2CVM160	2M26H	Rev 1.2
MCF52277CVM160	3M26H	Rev 1.3

You can also use the chip identification register (CIR) to determine the silicon revision. The table below lists the CIR[PRN] field values that correspond to given revisions.

Table 3. CIR[PRN] to Revision

CIR[PRN] value	Revision
0	Rev 1.1
1	Rev 1.2
2	Rev 1.3

Revision History

The table below provides a revision history for this document.

Table 4. Document Revision History

Rev. No.	Date	Substantive Changes
5	4/2008	Added level 2 trigger operation errata
6	12/2008	Added SECF121 "LCD Monochrome Mode Function Does Not Operate Correctly"
7	2/2009	Updated SECF032, “LCD LSCLK Signal Misses Pulse Before LCD_OE Asserts”. The errata is also present in any mode that has an 18-bit output. So, it affects 4bpp and 8bpp modes as well.
8	2/2009	Updated Status sections for SECF046, “Resolution of ASP ADC Channels”, SECF033, “PLL Loss of Lock and Clock Jitter”, and SECF121, “LCD Monochrome Mode Function Does Not Operate Correctly”

Rev. No.	Date	Substantive Changes
9	5/2009	Added Rev 1.2 mask to Table 1 Added Table 2 Updated SECF032, "LCD LSCLK Signal Misses Pulse Before LCD_OE Asserts" to SECF033, "PLL Loss of Lock and Clock Jitter" to reflect the new Rev 1.2 mask. Added SECF122, "Possible SRAM Data Corruption When Booting from Internal SRAM Through the Serial Boot Facility" Added SECF126, "Signal Multiplexing for the LCD Data Signals Does Not Function as Documented"
10	6/2009	Changed workaround of SECF122, "Possible SRAM Data Corruption When Booting from Internal SRAM Through the Serial Boot Facility" to TBD
11	6/2009	Added new workaround to SECF122, "Possible SRAM Data Corruption When Booting from Internal SRAM Through the Serial Boot Facility"
12	8/2009	Added SECF150
13	9/2009	Added new revision 1.3 device, which corrects SECF121.
14	6/2010	Added SECF162. Added SECF178. Corrected PRN values in Table 3 .
15	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.

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 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.

SECF032: LCD LSCLK Signal Misses Pulse Before LCD_OE Asserts

Errata type: Silicon

Affected component: LCD

Description: When operating in 4bpp, 8bpp, or 18bpp mode, the LSCLK signal misses a clock pulse in the clock cycle before LCD_OE asserts. Since the missing clock pulse occurs when LCD_OE is negated, most LCD panels operate correctly with no issue.

Workaround: If a continuous clock is needed even with LCD_OE negated, then 12bpp or 16bpp modes can be used. In these modes the LSCLK does not miss the pulse.

Fix plan: Fixed on Rev 1.2 devices and later mask sets.

SECF033: PLL Loss of Lock and Clock Jitter

Errata type: Silicon

Affected component: PLL

Description: The original version of the PLL is sensitive to noise that can cause cycle-to-cycle jitter on the output clocks and, in some cases, cause the PLL to lose lock. This behavior of the PLL is related to the voltage used for the SDVDD rail and activity on the FlexBus.

Using a nominal SDVDD of 3.3V is the worst case. FlexBus activity can trigger a loss-of-lock condition. When a loss of lock occurs, the output clock switches to the limp mode clock. Then, when lock is regained the system clocks return to using the PLL output. In addition, the output clock may also show cycle-to-cycle jitter even when the loss of lock does not occur (up to 2 ns).

At a nominal 2.5V or 1.8V SDVDD, the PLL loss of lock does not occur. However, the PLL output clock can have cycle-to-cycle jitter. The cycle-to-cycle jitter should be less than 700 ps.

Workaround: When using 3.3V for SDVDD, loss-of-lock detection by the PLL can be disabled by setting bit four in the PLL status register (PSR). This prevents the output clocks from switching to the limp mode clock, but there can still be cycle-to-cycle jitter on the output clocks. However, a loss-of-lock event can occur when booting from memory on the FlexBus before there is a chance to disable the loss-of-lock detect in software.

Workaround: Use 2.5V or 1.8V for the SDVDD supply. If there are no sychronous devices in the system running from a clock source other than the ColdFire device, then no workaround should be needed.

Workaround: Use parallel boot, but boot the processor in limp mode. Once the loss-of-lock detect is disabled the PLL can be enabled.

Workaround: Use a serial device and the serial boot facility (SBF) to configure the processor and execute code to disable the loss-of-lock detect.

Workaround:	Use an asychronous non-volatile memory device for booting and disable the loss-of-lock detect in software as early as possible.
Fix plan:	On Rev 1.2 and later mask sets, during a loss-of-lock condition the device operates with the PLL output clock, if the PLL clock is present. The device does not switch to the limp clock, unless software programs it for limp mode operation.

SECF040: RTC Operation in Stop Mode

Errata type:	Silicon
Affected component:	RTC
Description:	On the original silicon, the RTC module stops counting when the device enters stop mode.
Workaround:	No workarounds
Fix plan:	On Rev 1.2 and later mask sets, the RTC continues to count in stop mode as long as its input clock is running. This way the RTC is able to wake the processor from stop mode.

SECF046: Resolution of ASP ADC Channels

Errata type:	Silicon
Affected component:	Touchscreen/ASP
Description:	On the original silicon, the resolution of the ADC channels that feed into the ASP block is lower than specified in the *MCF5227x Data Sheet* (approximately four bits of accuracy). The ADC logic is a new block, so the need for tuning of the circuit was expected and planned for in the silicon schedule.
Workaround:	No workarounds.
Fix plan:	The accuracy of the ADC circuit was increased on the Rev 1.2 and later mask sets. Please see the *MCF5227x Data Sheet* for the ASP ADC specifications.

SECF121: LCD Monochrome Mode Function Does Not Operate Correctly

Errata type:	Silicon
Affected component:	LCD
Description:	The LCD controller's monochrome mode does not work correctly.
Workaround:	Do not use monochrome mode. CSTN and TFT modes work correctly.
Fix plan:	Fixed on Rev 1.3 devices and later mask sets.

SECF122: Possible SRAM Data Corruption When Booting from Internal SRAM Through the Serial Boot Facility

Errata type:	Silicon
Affected component:	SBF
Description:	When booting the device in SBF mode with a non-zero SBFSR[BLL] (using the SBF to copy code into the internal SRAM), attempting to execute a movec instruction will write data to the corresponding lonword address in the on-chip SRAM instead of the CPU space register.

Workaround:

Do not execute movec instructions when using SBF mode to boot from the internal SRAM. This means that the RAMBAR, VBR, CACR, and ACR*n* registers cannot be programmed. The inability to program the CPU space registers has the following effects:

- The core has single cycle access to the on-chip SRAM starting at address 0x0000_0000. To avoid overlap with the on-chip SRAM, do not use the first 128 KBytes of the FlexBus memory space (0x0000_0000 - 0x3FFF_FFFF).
- The SRAM is still accessible in the 0x8000_0000 - 0x8FFF_FFFF through the SRAM's back door. Use this address range for any non-core masters that need access to the SRAM.
- Cache cannot be used.

In addition, the BDM functionality is impacted. BDM can be used to read and write memory and module registers. The BDM uses CPU space to read and write the PC, so run and single-step functions do not work correctly.

Fix plan: TBD

SECF126: Signal Multiplexing for the LCD Data Signals Does Not Function as Documented

Errata type: Silicon

Affected component: LCD

Description:

The signal multiplexing for the LCD_D*n* pins does not work as specified in the PAR_LCDH and PAR_LCDL register descriptions in the *MCF5227x Reference Manual*. Setting any of the PAR_LD*n* bit fields to 10 (alternate 1 function) may affect the signal used on non-corresponding pins, as shown in the following table.

For example, if PAR_LD0 = 10 and PAR_LD2 = 00, the LCD_D0 pin is configured as the PWM1 signal and LCD_D2 is configured as LCD_D0, not GPIO.

NOTE

If all PAR_LD*n* fields are 11, the LCD_D*n* pins function as LCD data signals and are not affected by this errata.

NOTE

If all PAR_LD*n* fields are 00, the LCD_D*n* pins function as GPIO and are not affected by this errata.

Table 6. PAR_LD*n* Affect on Pin

PAR_LCDH or PAR_LCDL Bit Name	10 (First Priority)	11 (Second Priority)	00 (Third Priority)
PAR_LD0	LCD_D0 = PWM1 **LCD_D2 = LCD_D0**	LCD_D0 = LCD_D0	LCD_D0 = GPIO
PAR_LD1	LCD_D1 = PWM3 **LCD_D3 = LCD_D1**	LCD_D1 = LCD_D1	LCD_D1 = GPIO
PAR_LD2	LCD_D2 = LCD_D0 **LCD_D4 = LCD_D2**	LCD_D2 = LCD_D2	LCD_D2 = GPIO
PAR_LD3	LCD_D3 = LCD_D1 **LCD_D5 = LCD_D3**	LCD_D3 = LCD_D3	LCD_D3 = GPIO
PAR_LD4	LCD_D4 = LCD_D2 **LCD_D8 = LCD_D4**	LCD_D4 = LCD_D4	LCD_D4 = GPIO
PAR_LD5	LCD_D5 = LCD_D3 **LCD_D9 = LCD_D5**	LCD_D5 = LCD_D5	LCD_D5 = GPIO
PAR_LD6	LCD_D6 = PWM5 **LCD_D10 = LCD_D6**	LCD_D6 = LCD_D6	LCD_D6 = GPIO
PAR_LD7	LCD_D7 = PWM7 **LCD_D11 = LCDD7**	LCD_D7 = LCD_D7	LCD_D7 = GPIO
PAR_LD8	LCD_D8 = LCD_D4 **LCD_D14 = LCD_D8**	LCD_D8 = LCD_D8	LCD_D8 = GPIO
PAR_LD9	LCD_D9 = LCD_D5 **LCD_D15 = LCD_D9**	LCD_D9 = LCD_D9	LCD_D9 = GPIO
PAR_LD10	LCD_D10 = LCD_D6 **LCD_D16 = LCD_D10**	LCD_D10 = LCD_D10	LCD_D10 = GPIO
PAR_LD11	LCD_D11 = LCD_D7 **LCD_D17 = LCD_D11**	LCD_D11 = LCD_D11	LCD_D11 = GPIO
PAR_LD12	LCD_D12 = CANRX	LCD_D12 = LCD_D12	LCD_D12 = GPIO
PAR_LD13	LCD_D13 = CANTX	LCD_D13 = LCD_D13	LCD_D13 = GPIO
PAR_LD14	LCD_D14 = LCD_D8	LCD_D14 = LCD_D14	LCD_D14 = GPIO
PAR_LD15	LCD_D15 = LCD_D9	LCD_D15 = LCD_D15	LCD_D15 = GPIO
PAR_LD16	LCD_D16 = LCD_D10	LCD_D16 = LCD_D16	LCD_D16 = GPIO
PAR_LD17	LCD_D17 = LCD_D11	LCD_D17 = LCD_D17	LCD_D17 = GPIO

Workaround: Use the table above to properly select the desired signals.

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

SECF150: Potential Boot Failure When Using Unified SDR bus/FlexBus Mode

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 reading boot code from flash.

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: Add a pullup resistor on the SD_CKE signal. This allows the SDRAM memory to recognize clocks during processor startup, and should allow banks to return to the IDLE state.

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

SECF162: Internal Resets Are Not Functional in SBF Mode

Errata type: Silicon

Affected component: SBF and Reset

Description: When operating in SBF mode (BOOTMOD[1:0] = 11), internal SoC reset sources do not work reliably. This includes the software reset request (RCR[SOFTRST]) and the software and core watchdog timers.

Workaround: Do not use internal SoC reset sources in SBF mode. External resets (toggling the $\overline{\text{RESET}}$ input pin) and power-on resets (POR) work correctly. If watchdog capability is needed in SBF mode, then use an external watchdog timer. If the capability for a software reset request is needed, then an external circuit could be added that monitors a GPIO line and asserts the RESET pin when the processor toggles the GPIO.

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

SECF178: Primary LCD data functions are not available on the MCF52274

Errata type: Silicon

Affected component: GPIO-pin muxing

Affected component: LCDC

Description:

NOTE

The MCF52277 device is unaffected.

On the MCF52274 device, the primary functions of the LCD data lines on the device are not available (LCD_D[17:14], LCD_D[11:8], and LCD_D[5:2]). These pins can only be configured for their alternate functions—LCD_D[11:0] or GPIO.

In non-TFT mode, the data lines are sequential. Therefore, this errata does not apply, since the MCF52274 only supports a 12-bit data bus from the LCD controller. LCD_D[11:0] are the desired pin functions for non-TFT operation.

In TFT mode, the LCD controller skips over some of the data lines. For 12-bit TFT mode, LCD_D[17:14], LCD_D[11:8], and LCD_D[5:2] are needed (the primary pin functions that are not available on the MCF52274).

Workaround:

NOTE

No workaround is required for non-TFT mode operation on the MCF52274.

There are two workarounds that can be used to support TFT operation on the MCF52274. In both cases configure the PAR_LCDH and PAR_LCDL registers to enable the alternate1 pin functions (LCD_D[11:0]). Then either:

1. Configure the LCD controller for 18bpp TFT mode. In this case 12bpp color is supported from the LCD. The drawback is that the LCD controller fetches 32-bits of data from the graphic buffer per pixel. This is more data than is actually being used (since only 12 bits are driven to the panel per pixel), so it is an inefficient use of the system bus bandwidth.
2. Configure the LCD controller for 16bpp TFT mode. In this case 11bpp color is supported from the LCD (LCD_D0 is not used in 16bpp TFT mode), but the LCD controller only fetches 16-bits of data from the graphic buffer per pixel. So this approach trades off one bit of color to get more efficient use of the system bus. For system bus bandwidth-limited systems this approach might be the best option.

The other drawback to both of these approaches is that the color information for each pixel as interpreted by the LCD controller is incorrect. So, any function that would uses the LCD controller's alpha-blending features does not work correctly.

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

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 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: 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: MCF52277DE
Rev. 15, 8/2010