Device Errata
MPC801

Silicon Revision A.2 - Mask Set F87S
May 18, 1998 (Version - 4)

These errata are valid on Revision A.2 silicon. Please note that any errata listed in this document apply to previous versions of revision A of the silicon, unless otherwise stated.

SIU ERRATA

SIU1 CLKOUT signal drive.

The MPC801's capability to change the strength of the output buffer drive of CLKOUT is not functional. The strength of the output buffer can be modified between 2 states:

Full strength - COM = 00 or COM = 01 in the SCCR register

Disabled - COM = 11

Workaround: Do not use fractional drive capability.

This will be fixed in future revision.

PIO ERRATA

PIO1 Port B 24, 25 open drain mode.

When PB[24],PB[25] pins are programmed to "dedicated" mode, they are automatically changed to be open-drain signals, regardless of the appropriate OD bits of the PBODR register.

Workaround: Connect an external pullup resistor on these pins to the VDDH power supply.

I2C ERRATA

I2C Error in FLT bit

An error will occure if the FLT bit is set to turn on the digital filter for the I2C. The digital filter is activated by setting the FLT bit in the I2C mode register and is turned off at reset.

(However, note that this digital filter is not required for normal operation. The MPC8xx I2C is fully compliant to the I2C specification even without this digital filter.)

Workaround: Do not turn on the digital filter for I2C clock filter.

This will be fixed in future revision.

GENERAL ERRATA

G1 Termination of Open Drain and Active Pullup pins

Open Drain pins and Active pull-up pins will drain "high" current when the input voltage to them is higher than VDDH.

Active pullup pins include: BB*, TS*, TA*, BI*

Open drain pins include: TEA*, HRESET*, SRESET*, and any general-pupose I/O programmed as open drain

Workaround: Connect the external pull-up resistor on these pins to the VDDH power supply.

This will be fixed in future revision.

G2. Core Operation Is Limited to 3.0 Volt Minimum.

The current versions of the MPC801 silicon are only tested and verified at 3.0V to 3.6V power. Because of this, low voltage operation (@2.2V) cannot be used for powering the core.

Workaround: None. To be tested and verified in this or future silicon.

G3. Higher Than Expected Keep Alive Power (KAPWR) Current When Main Power (VDDH & VDDL) Is Removed.

There are four nodes within the MPC801 that are floating when VDDH and VDDL power is not supplied to the device. When this condition occurs, which is typical in a Power Down Mode, the current drain on the Keep-Alive Power rail is greater than expected. (10 - 20 mA versus 10 µA)

Workaround: Provide adequate current source for KAPWR pin in Power Down Mode.

This will be fixed in future revision.

G4. EXTCLK and CLKOUT Clocks may not be in phase in half-speed bus mode.

When the MPC801 uses EXTCLK as an input clock source and MF=001 in PLPRCR (i.e. the frequency of EXTCLK is 1/2 of the internal clock) and the half- speed bus mode is used (EBDF=01 in SCCR), the output clock from CLKOUT could be 90 degrees or 180 degrees out of phase from the input clock. This will effect synchronous designs where the same clock source is used as an input to EXTCLK, as well as to an external synchronous device (e.g. a peripheral or ASIC).

Workarounds:

Case 1. Where multiple external devices need to operate synchronously with the MPC801:

Use the CLKOUT pin of the MPC801 as the source of clock for all external, synchronous devices (i.e. CLKOUT is the effective system master clock to be used for distribution).

Case 2. Where it is necessary to synchronize an external master clock (e.g. from a backplane), an MPC801, and external peripherals, to allow data transfers in all three directions:

There is no known workaround for this case, use full-speed bus operationuntil fixed in futuree revision .

This will be fixed in future revision.

G5. Potential problems caused by skew between EXTCLK and CLKOUT.

The PLL of the MPC801 locks on the falling edge of the input clock EXTCLK. This will affect the skew between EXTCLK and CLKOUT at the rising edge. The skew is dependent on the duty cycle of the input clock (but for a 50% duty cycle will not exceed 2nS). This will affect synchronous designs where the same clock source is used as an input to EXTCLK, as well as to an external synchronous device (e.g. a peripheral or ASIC).

Workarounds:

Case 1. Where multiple external devices need to operate synchronously with the MPC801:

Use the CLKOUT pin of the MPC801 as the source of clock for all external, synchronous devices (i.e. CLKOUT is the effective system master clock to be used for distribution).

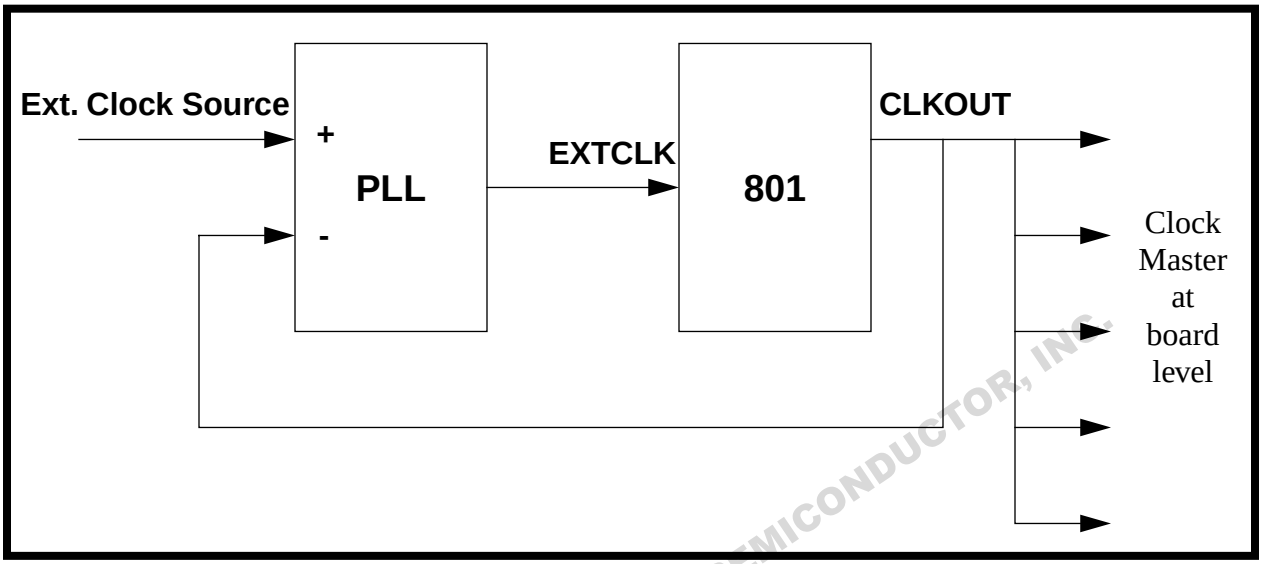
Case 2. Where it is necessary to synchronize an external master clock (e.g. from a backplane), an MPC801, and external peripherals, to allow data transfers in all three directions:

[NOTE: This workaround is a concept only. It has not been verified in hardware.]

Insert a PLL between the external master clock and the EXTCLK pin of the MPC801. Connect the phase comparison pin of the PLL to the CLKOUT pin of the MPC 801. Also use the CLKOUT signal as the reference clock for distribution to the local external peripherals.

Important Note: The PLL has to be capable of operating with a permanent offset of -2nS, therefore the range of lock should extend to about -4nS.

A diagram of this concept is given below:



This will be fixed in future revision, changing the PLL to lock on the rising edge of EXTCLK.

CPU.

CPU1 Bus Error unsupported by Data Cache bursts.

The Data Cache does not support bus error which occurs on the 2nd or 3rd data beat of a burst.

Workaround: Refrain from using bus error in these cases.

CPU2 Wrong data breakpoint detection on store instructions.

When a breakpoint on data elements of size of byte or half word is programmed for store instructions, the following erroneous scenarios may occur:

1) Breakpoint may be detected when it should not.

2) Breakpoint may not be detected when it should.

This scenarios can happen in cases where either byte or half-word size of data are programmed to be detected, and this data matches some other portion of the register that is currently stored to memory by using store byte or half-word instruction.

Examples:

1) Assume that the user has programmed a byte data breakpoint on store instructions and is looking for the byte element: 0x55. Assume that register R1 has the value: 0x00080000, R10 has the value: 0x55443322 and the following store instruction is performed:

```
stb R10,0x3(R1)
```

What happens is that byte 0x22 from R10 is stored to address 0x00080003, and this should not generate a breakpoint since 0x22 != 0x55, BUT: In some cases it does (unfortunately R10 does include the data 0x55 ...).

2) Assume that in the above case the user is programming for byte element 0x22, all other conditions are as above, then it might happen that a breakpoint will not be detected although it should.

NOTE:

This fault cases depend on the previous Load-Store instruction address. f the previous Load-Store instruction address LSB's is different from the current instruction address LSB's than it might happen.

Workaround: Do not program data breakpoints for byte or halfword data elements.

This will be fixed in future revision.

CPU3 Program Trace Mechanism Error

In the following case there is an error in the program trace mechanism.

Program

0x00004ff0: divw. r25,r27,r26

0x00004ff4: divw. r28,r27,r26

0x00004ff8: unimplemented

0x00004ffc: b 0x00005010

where 0x00005010 belong to a page with a page fault.

The divide takes a long time to complete so the instruction queue gets filled with the unimplemented instruction, the branch and the branch target (page fault).

When the sequencer takes the unimplemented instruction releases the fetch (that was blocked by the MMU error) this causes the queue to get another instruction in addition to the first page fault. Because the second fault is sequential to the branch target it is not reported by the queue flush (VF). This causes an incorrect value to be present in the VF flush information when the unimplemnted exception occurs.

Workaround: None.

This will be fixed in future revision.

CPU4 Incorrect value used for POW Bit of MSR register.

The value of the bit POW in the MSR register was taken with opposite polarity by the clock control logic. Correct operation of the logic will assure that: if PRQEN in the SCCR register is '1', the system clock will switch/ remain in high frequency as far as POW bit in the MSR register is reset ('0'). In order to allow to the system to switch to "low" frequency the POW bit should be previously set (1'). Under this definition if the source of an interrupt request is reset right after jumping to the software routine that handles it, the clock will continue to run in high frequency due to the fact that the POW bit is reset any time that an interrupt service routine is run by the processor. With the existent problem, if the Interrupt source is reset, the clock will switch to low frequency.

Workaround: Reset the source of the interrupt request BEFORE the RFI instruction is executed; or manipulate the CSRC bit in the PLPRCR register such that it is reset when entering the interrupt software routine and set before the RFI instruction is executed.

This will be fixed in future revision.

CPU5 I-cache replacement policy bug

I-cache replacement policy is not optimized. This does not affect the correctness of program execution, but will affect performance by an average of 10-20%. Once new silicon is available, performance should improve without any software changes required.

This will be fixed in future revision.

CPU6 Instruction MMU bug at page boundaries in show-all mode

The wrong instruction address is driven by the core when all the following conditions occur:

1. *MPC860 works in 'show all' mode (i.e. ISCT_SER bits=000 in ICTRL)*
2. *Sequential instruction crosses IMMU page boundary*
3. *Instruction cache fails to get ownership of the internal U-bus on the first clock*

In this case the address driven by the core will be of the previous page and not the current one.

Workaround: Possible workarounds include:

1. *Disable show all mode*
2. *Invalidate the page next to current (by using the tlbie instruction) when performing the TLB reload operation.*

This will be fixed in future revision.

CPU7 Possible data cache corruption when writing SPRs

A write access to a special-purpose register located in caches, MMUs or SIU might corrupt the contents of the data-cache. These special-purpose registers include:

SPR	spr_address
IMMR	0x3d30
IC_CST	0x2110
IC_ADR	0x2310
IC_DAT	0x2510
DC_CST	0x3110
DC_ADR	0x3310
DC_DAT	0x3510
MI_CTR	0x2180
MI_AP	0x2580
MI_EPN	0x2780
MI_TWC	0x2b80

MI_RPN	*0x2d80*
MI_DBCAM	*0x2190*
MI_DBRAM0	*0x2390*
MI_DBRAM1	*0x2590*
MD_CTR	*0x3180*
M_CASID	*0x3380*
MD_AP	*0x3580*
MD_EPN	*0x3780*
M_TWB	*0x3980*
MD_TWC	*0x3b80*
MD_RPN	*0x3d80*
M_TW	*0x3f80*
MD_DBCAM	*0x3190*
MD_DBRAM0	*0x3390*
MD_DBRAM1	*0x3590*
DEC	*0x2c00*
TB Write	*0x3880*
TBU Write	*0x3a80*
DPDR	*0x2d30*

Workaround: There are two possible work-arounds:

1. If the contents of the TLBs are not changed dynamically (fixed-page structure), any access to the above-mentioned registers should be avoided (except for initialization).

2. If the contents of the TLBs are changed dynamically (pages are loaded on demand), then each "mtspr" instruction which accesses one of these registers must be preceded by a store word and a load word instruction of a data operand equal to the spr_address of the respective register. As an example, to write the data from the general purpose register r1 to the special purpose register M_TW, the following procedure should be followed:

```
lis    r2,    some_address_msb      # an address in RAM
li     r3,    0x3f80                # the spr_address of the M_TW from
                                     # the table
stw    r3,    some_address_lsb(r2)  #  no interrupts
lwz    r3,    some_address_lsb(r2)  #  between these
mtspr  M_TW,  r1                    #  3 instructions
```

This will be fixed in future revision..

CPU8 Branch Prediction with Sequential Branch Instructions

IF

there are three branches in sequence in the run-time program flow

AND

the third branch is in the mis-predicted path of the second branch,

THEN

although the third branch is part of a predicted path, it will be "issued" from the instruction queue. If this instruction issue in the mis-predicted path happens at the same time that the condition of the prediction is resolved, then the resulting instruction cancellation will back up too far into the instruction queue. This will cause the instruction immediately preceding the first branch to be re-issued.

Note:

The following compilers are known never to generate code that is susceptible to this erratum:

Diab Data (all versions)

Metaware

We are continuing to investigate other compilers with their vendors; their status is unknown at this time. We will update this list as our investigation progresses.

Workaround:

For every conditional branch preceded in program order by another branch:

1) IF

the two possible targets of the conditional branch consist of a branch instruction and a non-branch instruction

THEN

force the prediction of the conditional branch to predict the non-branch instruction (using the y-bit in the opcode of the conditional branch).

2) IF

both of the possible targets of the conditional branch are branch instructions

THEN

(1) insert a non-branch instruction before the branch on the predicted path

OR

(2) insert an 'isync' instruction before the first branch.

This will be fixed in future revision.

CPU9 Missed Instruction after Branch.

IF

the target address of a branch instruction maps to the same Instruction Cache set number and the same word in the cache line as the originating branch instruction (e.g. instruction at XXXXXFFC branches to YYYYYFFC)

AND

the instruction at the target address resides in a different cache way as the originating branch instruction.

THEN

the instruction at the target address will not execute.

Note:

The behavior of this erratum is deterministic, not subject to chance. Therefore, code which does not currently show effects from this erratum will not be effected.

Workarounds:

1) Avoid the situation described above (by reorganizing the code, or inserting no-ops).

2) Disable the Instruction Cache.

3) Enable serialization of the core.

This will be fixed in future revision.