ST10R272L-B0

16-BIT ROMLESS LOW VOLTAGE MCU WITH MAC

ERRATA SHEET

1 INTRODUCTION

This errata sheet describes the functional and electrical problems known in the step B0 of the ST10R272L-B0. This is the erratasheet of the ST10R272L datasheet version 1.2 of april 2000.

2 FUNCTIONAL PROBLEMS

The following malfunctions are known in this step:

2.1 ST_PWRDN.1: EXECUTION OF PWRDN INSTRUCTION WHILE $\overline{\text{NMI}}$ PIN IS HIGH

When PWRDN instruction is executed while $\overline{\text{NMI}}$ pin is at a high level, power-down mode should not be entered, and the PWRDN instruction should be ignored. However, under the conditions described below, the PWRDN instruction may not be ignored, and no further instructions are fetched from external memory, i.e. the CPU is in a quasi-idle state. This problem will only occur in the following situations:

1) the instructions following the PWRDN instruction are located in an external memory, and a multiplexed bus configuration with memory tristate waitstate (bit MTTCx= 0) is used,

2) the instruction preceding the PWRDN instruction writes to external memory or an XPeripheral (XRAM, CAN), and the instructions following the PWRDN instruction are located in external memory. In this case, the problem will occur for any bus configuration.

Note: the on-chip peripherals still work correctly: if the Watchdog Timer is not disabled, it will reset the device upon an overflow. Interrupts and PEC transfers, however, can not be processed. If $\overline{\text{NMI}}$ is asserted low while the device is in this quasi-idle state, power-down mode is entered.

No problem will occur if the $\overline{\text{NMI}}$ pin is low: the chip will normally enter power-down mode.

Workaround: Ensure that no instruction which writes to external memory or an XPeripheral precedes the PWRDN instruction, otherwise insert e.g. a NOP instruction in front of PWRDN. When a multiplexed bus with memory tristate waitstate is used, the PWRDN instruction should be executed from internal RAM or XRAM.

2.2 ST_MAC.9: COCMP INSTRUCTION INVERTED OPERANDS

According to the ST10 Family Programming Manual, the CoCmp instruction substracts a 40-bit signed operand from the 40-bit accumulator content (acc - op2\op1), and updates the N, Z and C flags in the MSW register, leaving the accumulator unchanged.

In error, the reverse operation (op2\op1 - acc) has been implemented in the MacUnit. Therefore, the N and C flags are set according to the reverse operation (Z flag is not affected).

Workaround: Change interpretation of the N and C flags in the MSW register.

2.3 ST_MAC.10: E-FLAG EVALUATION FOR COSHR AND COASHR INSTRUCTIONS WHEN SATURATION MODE IS ENABLED

The Logical and the Arithmetic Right Shift instructions (CoShr/CoAshr) are specified not to be affected by the saturation mode (MS bit of the MCW reg.). The result loaded in the accumulator is never saturated but **the evaluation of E flag in the MSW register is erroneous when saturation mode is enabled** and the E flag would be set.

MS bit is set: saturation mode is enabled			Accumulator			SL	E	SV	C	Z	N
Example:	Mov	R5, #5555h	--	----	----h	-	-	-	-	-	-
	CoLoad	R5, R5	00	5555	5555h	0	0	0	0	0	0
	Nop		00	5555	5555h	0	0	0	0	0	0
	Mov	MSW, #007Fh	7F	5555	5555h	0	0	0	0	0	0
	Nop		7F	5555	5555h	0	0	0	0	0	0
	CoShr	#1	3F	AAAA	AAAAh	0	0	0	0	0	0
						error					

In this example, the E flag is kept clear however MAE is used.

Workaround: Disable saturation mode before using Logical and Arithmetic Right Shift instructions and re-enable it just after.

2.4 CORE.4: INCORRECT INSTRUCTION FETCH ON JUMP TO ITSELF

The bug happens In the following program sequence:

```
        ...
        ...
Label_A: JMPR cc_XX, Label_A
        Word Intruction 1;
        Word Intruction 2
        Word Intruction 3
             ...
        ...
        ...
```

In the following conditions:

- code is fetched from External Memory,
- the loop JMPR cc_XX, Label_A is being executed,
- a PEC transfer with **PSW** as destination triggers a change of the condition cc_XX and so, the loop is finished,

the Word Intruction1 is *never* executed.

Workaround: If JMPA is used instead of JMPR, the bug does not occur.

2.5 EBC.3: VISIBLE MODE

When visible mode is enabled (syscon.1 = 1), data of a read access to an XBUS peripheral is not driven to the external bus (Port 0). Instead, Port 0 is tri-stated during these read accesses.

If all external devices are configured in 8-bit demultiplexed mode, an XBUS-peripheral write can cause a conflict on P0H (Port 0 [8:15]).

2.6 EBC.4: XPERS ACCESS IN XPERSHARE/EMULATION MODE

In emulation mode and if the Startup Configuration 8-bit multiplexed mode is selected, P0H (Port 0 [8:15]) is always an output and write accesses to XPERs cannot be done as these would cause a conflict.

Workaround: Use a Startup Configuration other than 8-bit multiplexed mode.

If HOLD mode is entered (P6.5 = 0) following an 8-bit multiplexed mode access and if Xpershare is enabled, Xper accesses will cause a conflict on the internal XBUS xb_data [15:7] bus.

Workaround: None.

3 HISTORY OF FIXED FUNCTIONAL PROBLEMS OF THE ST10R272L:

Functional Problem	Short Description	fixed in Step
CPU.17	Arithmetic Overflow by DIVLU instruction	B0
TRAP_B.1	ATOMIC / EXTended sequences in Class B Hardware Trap	B0
Kfm_BR03	Pipeline conflict after CoStore operation	B0
Kfm_BR04	Wrong PSW value after byte instruction	B0
Kfm_BR05	Wrong result for BFLDL/BFLDH instructions	B0
CORE.3	Incorrect instruction fetch on Cache Jump	B0
CORE.4	Incorrect instruction fetch on Jump to itself	B0
CLK.1	ADAPT mode entered during power-up synchronous reset	B0
DAD.6	Incorrect PEC source fetch after JMPS instruction	B0
DAD.7	Incorrect pipeline dependency detection between DPRAM and (E)SFR address space	B0