

# MARC4 Microcontroller Enhancement Report

## Issues

- Microcontroller - Issue 1: Enhancement of Power-on Reset
- Microcontroller - Issue 2: Enhancement of Crystal Oscillator
- Microcontroller - Issue 3: Enhancement of Interrupt Handling
- Microcontroller - Issue 4: Enhancement of the DI Instruction
- Microcontroller - Issue 5: Enhancement of the DI/RTI-combination
- Microcontroller - Issue 6: FAB Transfer of Microcontroller
- EE - Issue 1: FAB Transfer of EEPROM
- EE - Issue 2: Improve Process Variation Robustness
- EE - Issue 3: Improve EEPROM EMC Robustness

## 1. Introduction

Atmel®'s MARC4 microcontroller family is subject of a continuous improvement process. The MARC4 microcontroller family consists of different members with or without EEPROM or RF-transmitter integrated. This document describes the different improvements done in each part, separated into a controller and a peripheral section.

The improvements or enhancements result from customer requests or requirements as well as from internal improvement processes. They are not only related to electrical or functional issues but also to fabrication or production flow issues.

An overview is given in [Table 1-1 on page 2](#). A detailed description of each enhancement/improvement, as well as how the enhancements have been realized, is provided on the following pages.



**MARC4**

**Enhancement  
Report**

4668D-4BMCU-02/08



**Table 1-1. Hardware Revisions**

Product			Issue Solved in Revision								
Family	Revision Code of Micro-Controller	Revision Code of Product with EE <sup>2</sup>	Micro-controller Issue 1	Micro-controller Issue 2	Micro-controller Issue 3	Micro-controller Issue 4	Micro-controller Issue 5	Micro-controller Issue 6	EE Issue 1 T505M-O	EE Issue 2 T505M-P	EE Issue 3 T505M-R
			(Reset)	(Osc.)	(Int.)	(DI)	(DI/RTI)	(Fab.)	(E <sup>2</sup> -Fab.)	(Proc.)	(EMC)
M44C092 ATAR092	H	H	x	x	x	x	--	--			
	K	K	✓ (a)	x	x	x	--	--			
	L	L	✓ (a)	✓	x	x	--	--			
	N	N	✓ (a)	✓	✓	x	--	--			
	O	O	✓ (a)	✓	✓	x	--	--			
	P	P	✓ (a)	✓	✓	✓	x	--			
	V	V	✓ (a)	✓	✓	✓	✓	✓			
M44C892 ATAR892	H	H	x	x	x	x	--	--	✓		
	K	K	✓ (a)	x	x	x	--	--	✓		
	L	L	✓ (a)	✓	x	x	--	--	✓		
	L	S	✓ (a)	✓	x	x	--	--		✓	
	N	N	✓ (a)	✓	✓	x	--	--	✓		
	N	T	✓ (a)	✓	✓	x	--	--		✓	
	O	O	✓ (a)	✓	✓	x	--	--	✓		
	P	P	✓ (a)	✓	✓	✓	x	--			
	O	U	✓ (a)	✓	✓	x	--	--		✓	
	O	W	✓ (a)	✓	✓	x	--	--			✓
ATAR862	V	V	✓ (a)	✓	✓	✓	✓	✓			✓
	O	M	✓ (a)	✓	✓	x	--	--	✓		
	O	N	✓ (a)	✓	✓	x	--	--		✓	
	O	R	✓ (a)	✓	✓	x	--	--			✓
	V	P	✓ (a)	✓	✓	✓	✓	✓		✓	
U9280M	V	S	✓ (a)	✓	✓	✓	✓	✓			✓
	M	M	x	x	x	x	--	--			
	L	L	✓ (a)	✓	x	x	--	--			
M44C090 ATAR090	C	C	x	x	x	x	--	--			
	D	D	✓	x	x	x	--	--			
	E	E	✓	✓	x	x	--	--			
	F	F	✓ (a)	✓	x	x	--	--			
	G	G	✓ (a)	✓	✓	x	--	--			
	H	H	✓ (a)	✓	✓	✓	x	--			
	R (pend.)		✓ (a)	✓	✓	✓	✓	✓			

Note: (✓) = partially solved, ✓ = solved, (a) = Low Power Reset Enhancement, x = affected, -- = not applicable

= Not for new designs

= Not for new designs, stopped

= Not for new designs, end of life planned

= Stopped

= Not available

Table 1-1. Hardware Revisions (Continued)

Product			Issue Solved in Revision								
Family	Revision Code of Micro-Controller	Revision Code of Product with EE <sup>2</sup>	Micro-controller Issue 1	Micro-controller Issue 2	Micro-controller Issue 3	Micro-controller Issue 4	Micro-controller Issue 5	Micro-controller Issue 6	EE Issue 1 T505M-O	EE Issue 2 T505M-P	EE Issue 3 T505M-R
			(Reset)	(Osc.)	(Int.)	(DI)	(DI/RTI)	(Fab.)	(E <sup>2</sup> -Fab.)	(Proc.)	(EMC)
M44C890 ATAR890	C	C	x	x	x	x	--	--	✓		
	D	D	✓	x	x	x	--	--	✓		
	D	K	✓	x	x	x	--	--		✓	
	D	S	✓	x	x	x	--	--			✓
	E	E	✓	✓	x	x	--	--	✓		
	E	L	✓	✓	x	x	--	--		✓	
	F	F	✓ (a)	✓	x	x	--	--	✓		
	F	M	✓ (a)	✓	x	x	--	--		✓	
	G	G	✓ (a)	✓	✓	x	--	--	✓		
	H	H	✓ (a)	✓	✓	✓	x	--	✓		
	G	N	✓ (a)	✓	✓	x	--	--		✓	
	G	P	✓ (a)	✓	✓	x	--	--			✓
R (pend.)		✓ (a)	✓	✓	✓	✓	✓			✓	
M44C080 ATAR080 T6020M ATA6020N	B	B	x	x	x	x	--	--			
	C	C	✓	x	x	x	--	--			
	D	D	✓	x	x	x	--	--			
	E	E	✓	✓	✓	x	--	--			
	F	F	✓	✓	✓	✓	x	--			
H (pend.)		✓	✓	✓	✓	✓	✓				
T48C893 ATAM893 (EE on Chip)	N	N	x	x	x	x	--	--			
	P	P	✓ (a)	x	x	x	--	--			
	V2.4	V2.4	✓ (a)	(✓)	x	x	--	--			
	R	R	✓ (a)	✓	✓	x	--	--			
	S	S	✓ (a)	✓	✓	✓	x	--			
T	T	✓ (a)	✓	✓	✓	✓	--				
ATAM862 (EE on Chip)	R	M	✓ (a)	✓	✓	x	--	--			
	S	N	✓ (a)	✓	✓	✓	x	--			
	T	P	✓ (a)	✓	✓	✓	✓	--			
T48C894 ATAM894 (EE on chip)	V1.0	V1.0	✓ (a)	✓	x	x	--	--			
	V1.1	V1.1	✓ (a)	✓	x	x	--	--			
	V1.2/3/4	V1.2/3/4	✓ (a)	(✓)	x	x	--	--			
	N	N	✓ (a)	✓	✓	x	--	--			
	P	P	✓ (a)	✓	✓	✓	x	--			
R	R	✓ (a)	✓	✓	✓	✓	--				
M44C510 ATAR510	E	E	x	x	x	x	--	--			
	F	F	✓	✓	✓	(✓)	x	--			
	G	G	✓	✓	✓	x	--	--			
T48C510	N	N	x	x	x	x	--	--			

Note: (✓) = partially solved, ✓ = solved, (a) = Low Power Reset Enhancement, x = affected, -- = not applicable

= Not for new designs

= Not for new designs, stopped

= Not for new designs, end of life planned

= Stopped

= Not available



## 2. Microcontroller - Issue 1: Enhancement of Power-on Reset

### 2.1 Circuit Does Not Always Start Operating After Power-on

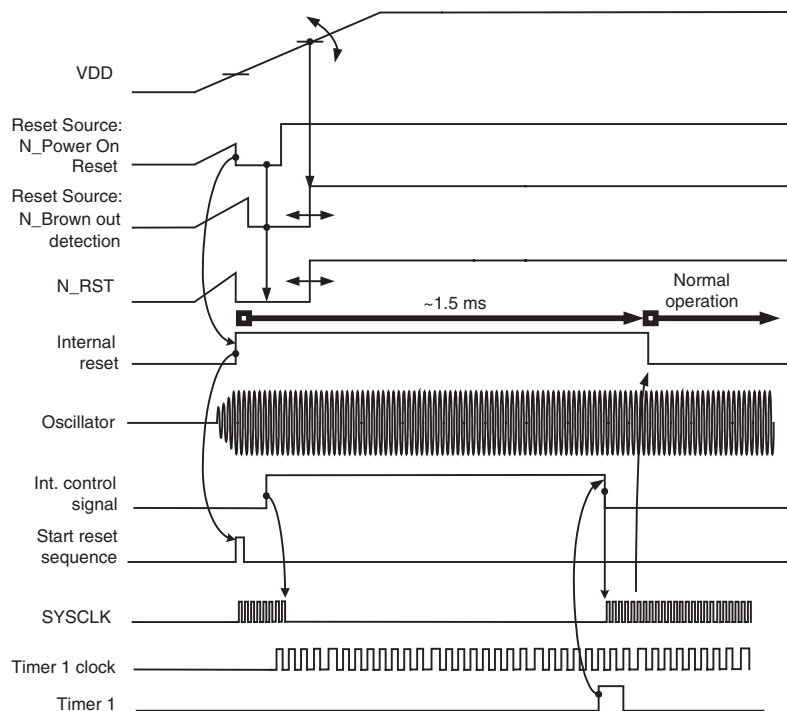
When power is applied to the circuit from the VDD line it is sometimes possible that the MARC4 will not start operating properly. This occurs under very special conditions and is a result of bouncing on the VDD line. Observing the I/O-pins one will find that they all stay in input mode with their pull resistors active (for example BP20, strong pull up), indicating that the circuit is still in the reset state.

### 2.2 Description

#### 2.2.1 Normal Reset Operation

The input to the internal reset signal generator is *N\_RST*. It is build from a couple of different reset sources (described in the corresponding datasheets under “Master Reset”). One is the *N\_Power\_On\_Reset* and another one is the *N\_Brown\_Out\_Detection* that ensures a correct VDD level before *N\_RST* ends. With a rising VDD, the first signals that become active are the *N\_Power\_On\_Reset* and the internal RC-oscillator. *N\_Power\_On\_Reset* forces *N\_RST* to a low level, starting the *internal Reset*. The rising edge of *internal\_Reset* generates a *Start\_Reset\_Sequence*-pulse, thus enabling *SYSCLK* to deliver initialization pulses for different flip-flops. It is also used to set an *internal control signal*, responsible for the reset sequence, to high. Three clock pulses later *SYSCLK* is disabled, while the reset sequence proceeds under timer 1 control. If timer 1 has finished the *internal control signal* ends and *SYSCLK* is enabled again to complete the *internal reset*. Under normal conditions or with different VDD rise times this works quite well. Even a voltage drop at VDD down to a level of approximately 1.3V does not interfere with the circuit.

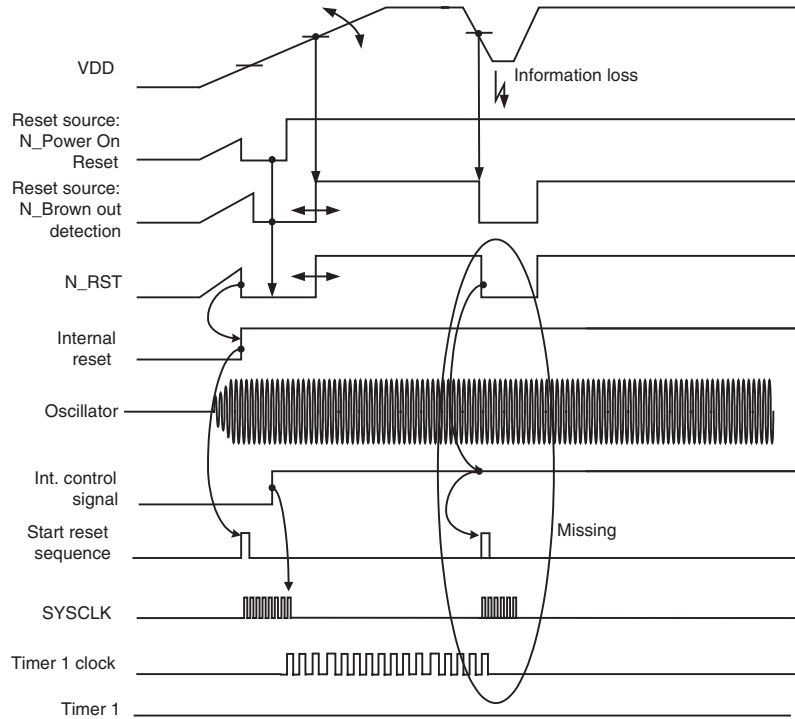
Figure 2-1. Normal Reset Function



## 2.2.2 Special Conditions due to Bouncing of VDD

In case of a voltage drop below ~1 V during the reset sequence, it is possible that some already initialized flip-flops may lose their information. As the internal reset cannot be interrupted the reset sequence does not restart (circled in Figure 2-2) and the flip-flops concerned will not be initialized again. If this happens to the flip-flop responsible for the clock selection of timer 1, it could lead to a deadlock or lock-up condition where the circuit is not able to finish the reset and start operating.

**Figure 2-2.** Reset Function Due to Bouncing



## 2.2.3 Workaround

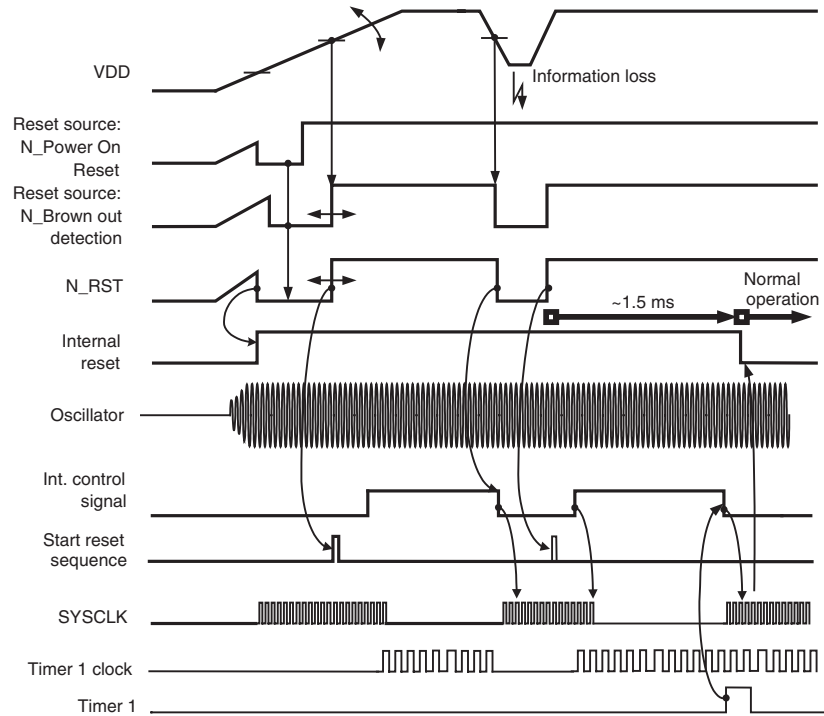
The only way to leave this deadlock or lock-up state and return to proper operation is to disconnect the power supply. Wait a few seconds to discharge all internal and external capacitors and then connect the power supply again.

## 2.3 Reset Enhancement

The enhancement introduced to the reset circuitry has the following effects.

1. The starting point of the reset sequence is moved to the rising edge of the  $N\_RST$  signal ensuring that the initialization starts operating under proper VDD conditions.
2. The  $N\_RST$  signal always restarts the reset sequence. This is the same condition as for power on. All flip-flops and registers that may have lost their information during the voltage drop are initialized again and the circuit starts operating properly.

**Figure 2-3.** Enhanced Reset Function



### 2.3.1 Declaration

This enhancement is fully compatible with the former normal reset operation and does not require any changes to either the software nor the application board for parts already in production.

The modification is performed by a minor change of the internal wiring (metal\_mask) only.

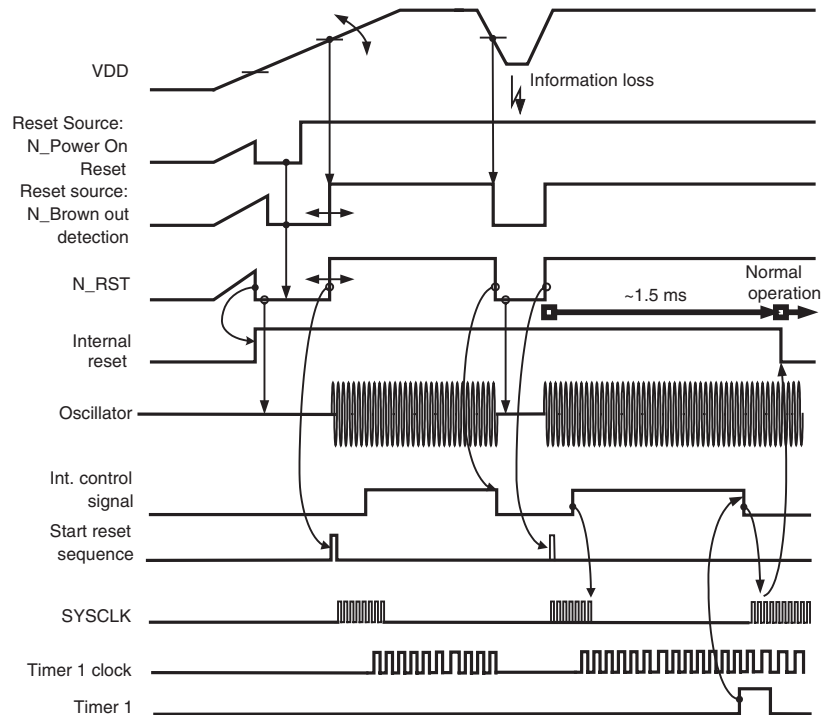
Several customer ROM versions have been transferred to this modified reset. They are fully evaluated, tested and delivered. All customers have reported that the reset issues no longer occurs and is no longer an issue.

## 2.4 Low Power Reset Enhancement

The enhancement introduced to the reset circuitry has the following effects.

1. The starting point of the reset sequence is moved to the rising edge of the  $N\_RST$  signal, ensuring that the initialization starts operating under proper VDD conditions.
2. The  $N\_RST$  signal always restarts the reset sequence. This is the same condition as for power on. All flip-flops and registers that may have lost their information during the voltage drop are initialized again and the circuit starts operating properly.
3. Reduce the power consumption as long as the supply voltage stays below the operating range. This is performed by stopping the oscillator (and all derived clock signals) while a reset condition is active ( $N\_RST = \text{low}$ ).

**Figure 2-4.** Enhanced Reset Function (Low Power)



### 2.4.1 Declaration

This enhancement is fully compatible to the former normal reset operation and does not require any changes to either the software or the application board for parts already in production.

The modification is performed by a minor change of the internal wiring (metal\_masks, via\_mask) only.

## 3. Microcontroller - Issue 2: Enhancement of Crystal Oscillator

### 3.1 Crystal Oscillator Does Not Always Restart

When the crystal oscillator has stopped due to disturbances, for example, through moisture or a short-circuit condition, it does not always restart but stays in stopped mode until the power is disconnected and applied again.

### 3.2 Description

#### 3.2.1 Normal Operation

The integrated crystal oscillator starts operating from the externally applied supply voltage level. When the oscillation has stabilized, the supply voltage is switched down to a low-level power-saving mode. The oscillation is monitored with the internal oscillator supervisor. If the oscillation is interrupted through internal or external disturbances, the supervisor switches the oscillator power supply back to normal supply voltage and generates an additional start-up current that causes the oscillation to restart.

#### 3.2.2 Special Conditions

Under some conditions, when the oscillator is completely stopped due to a short-circuit condition, for example by moisture, the supervisor itself can be shut off. This causes a permanent stop of the crystal oscillator, even if the short-circuit condition is removed.

#### 3.2.3 Workaround

The only way to leave this deadlock or lock-up state and return to proper operation is to disconnect the power supply. Remove the short-circuit condition, and then connect the power supply again.

#### 3.2.4 Restart Enhancement

The enhancement introduced to the oscillator monitor circuit is as follows. To remove the influence of a short-circuit condition to the supervisor, the sensitive connection was moved to another internal control node. Now an external short-circuit condition cannot lead to a lock-up state of the supervisor, thus enabling it to restart the oscillator properly when the short-circuit has been removed.

#### 3.2.5 Declaration

This enhancement is fully compatible to the former normal oscillator operation and does not require any changes to either the software or the application board for parts already in production.

The modification is performed by a minor change of the internal wiring (metal\_masks) only.

## 4. Microcontroller - Issue 3: Enhancement of Interrupt Handling

### 4.1 Handling of Interrupts

Sometimes it seems that the circuit does react only to the first in a sequence of interrupt signals.

### 4.2 Description

If an interrupt occurs during a special clock phase of an I/O instruction, the circuit acknowledges the interrupt, jumps into the interrupt service routine and blocks this interrupt routine from being entered again by a next interrupt from the same source. The blocking of this interrupt routine is not released until the next I/O instruction is executed.

This behavior does not appear if the circuit is in Sleep mode and the interrupt wakes up the controller.

#### 4.2.1 Workaround

To prevent the interrupt from being permanently blocked, it is necessary to execute an I/O instruction before expecting the next interrupt on the blocked channel. Another method is to have the watchdog running and generate a reset if this lock-up happens.

This I/O instruction may be located either inside the interrupt service routine itself or inside that part of the program that is executed after the RTI and before the next interrupt.

If the repetition rate of an interrupt is longer than it takes to return from its service routine, just introduce an I/O instruction at the very beginning of that interrupt service routine.

Possible I/O instructions are:

**IN:** reading of any I/O-port or subport register

**OUT:** writing to any I/O-port or subport register

**SWI:** any software interrupt

or an **interrupt** from any other source but the possibly blocked

#### 4.2.2 Examples that REQUIRE additional I/O Instructions

Some programs operate in a way that interrupts are not followed by any I/O instruction directly:

When interrupts are only used to decrement or increment a value on the stack and then wait for the next interrupt, this may lead to a deadlock condition. One way to return to normal operation is an interrupt from another source. Another one is a reset, for example, from the watchdog.

INT4 - example: Timer 2 may be used to generate INT4 just to decrement a value on the stack (i.e., counting interrupts). If this happens in a loop (INT4 →DEC →RTI →Wait) until the variable is zero an additional I/O instruction should be introduced into the flow to prevent INT4 from blocking.

### 4.2.3 Examples that DO NOT REQUIRE Additional I/O Instructions

Many interrupts do not require a special handling as they are normally followed by I/O instructions:

Debouncing of key inputs: If INT1, INT3 or INT5 is used as an input for single keys, debouncing is often done by reading the input more than once.

INT1: If INT1 is used with BP52 and BP53 it is necessary to read port 5 to identify which input caused the interrupt. In this case no further I/O instruction is required.

INT2: In many cases timer 1 is used to generate INT2 to wake up the controller from Sleep mode. In this case no additional I/O instruction is required.

INT3: If used with BP40 and BP43, same as with INT1. If INT3 signals buffer full/empty in SSI mode it is normally followed by an IN or OUT from or to the receive- or transmit-buffer. No further I/O instructions are required.

INT5: INT5 events (Compare1, Compare2, Edge) can be identified by reading the timer 3 status register. This read is also necessary to reset the status bit before the next event occurs. Therefore no additional I/O instruction is required.

INT6: If used with BP50 and BP51, same as with INT1.

### 4.2.4 Interrupt Enhancement

The enhancement introduced to the interrupt circuit is as follows. To prevent the interrupt from being blocked, a slight modification of the internally generated timing was performed. The modification does not influence the interrupt latency or reaction time.

### 4.2.5 Declaration

Circuits already in production, that did not show such a behavior up to now, are not expected to show it in the future.

This modification was performed by a minor change in the metal-1 mask.

Customers releasing a new firmware for a ROM-mask controller will automatically be set to the latest revision part, having this issue solved.

## 5. Microcontroller - Issue 4: Enhancement of the DI Instruction

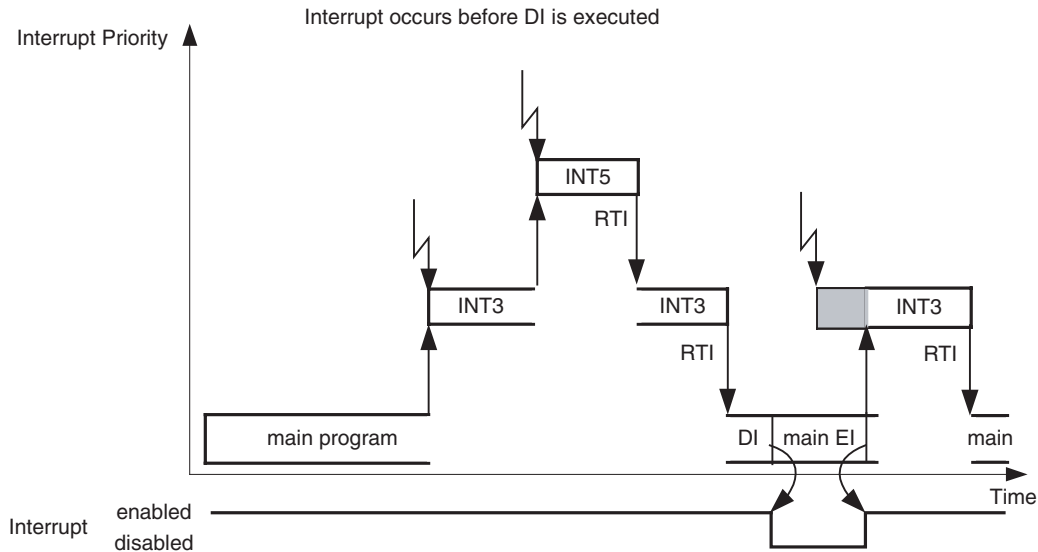
Sometimes it seems that the DI instruction does not fully protect the program against interrupts.

### 5.1 Description

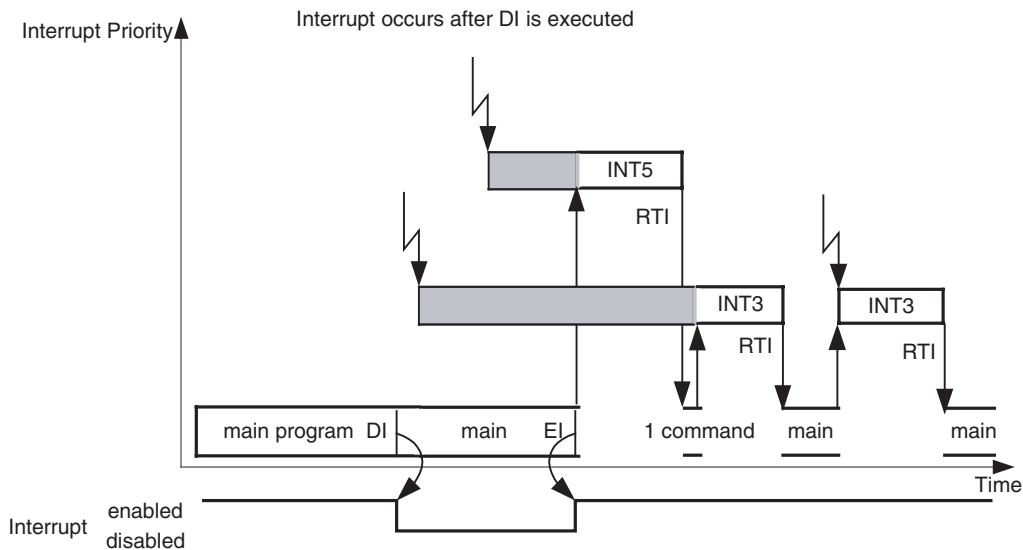
#### 5.1.1 Normal Conditions

Normally, the DI instruction is used to prevent sensitive parts of the program from being interrupted. The following 2 diagrams describe the normal behavior of the DI instruction.

**Figure 5-1.** Normal Interrupt Execution 1



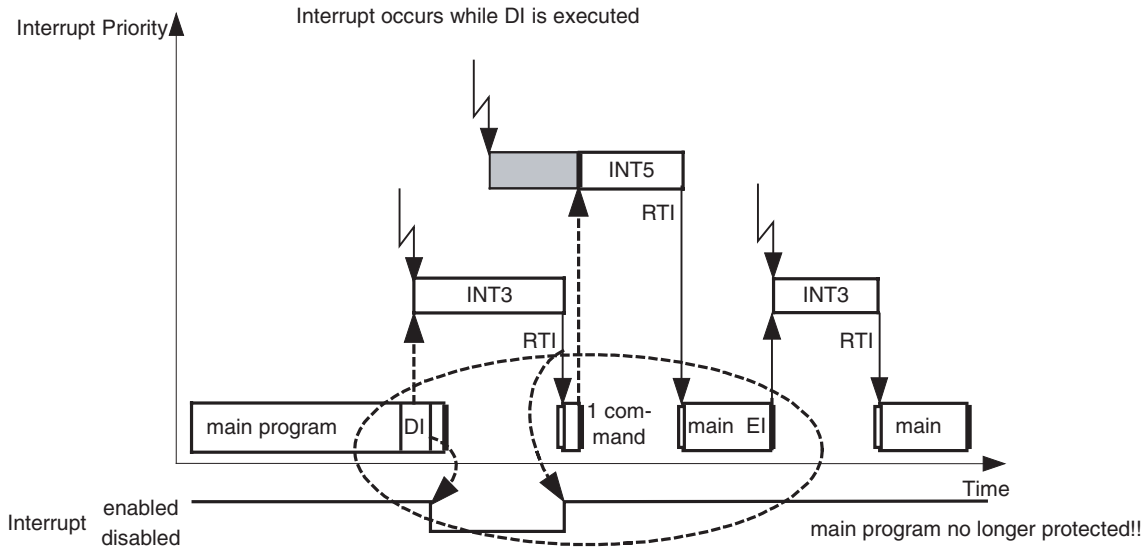
**Figure 5-2.** Normal Interrupt Execution 2



### 5.1.2 Special Conditions

If an interrupt occurs during a special clock phase while the DI instruction is being executed, the circuit acknowledges the interrupt, jumps into the interrupt service routine and at the same time finishes the DI instruction. Now the interrupt enable flag is reset (disable interrupt) and further interrupts are stored in the pending register. Leaving the service routine, the RTI (return from interrupt) sets the interrupt enable flag (enable interrupt). Program execution now continues at the instruction following the DI, but with interrupts enabled. If another interrupt occurs the DI no longer prevents the program from being interrupted.

**Figure 5-3.** Special Conditions



### 5.1.3 Workaround

#### 5.1.3.1 Method 1

The only way to ensure proper DI execution is to test that DI has been executed. This is best performed in a loop, until the interrupt enable flag is reset. Now, if the DI is interrupted the program returns into the loop, executes DI again and then continues with the program to be protected.

This can be easily performed by introducing one of the following code sequences into the include file C092.INC and use DI\_F or DI\_L instead of the original DI.

## Speed optimized version:

```
CODE DI_F
Begin
DI
CCR@
SHR
DROP
TOG_BF
UNTIL
END-CODE
```

## Code size optimized version:

```
: DI_L
Begin
DI
CCR@
SHR
DROP
TOG_BF
UNTIL
[ Z ] ;
```

### 5.1.3.2 *Method 2*

In addition to the above described safe method, there is another way to drastically reduce the probability of a DI-malfunction. Just use 2 consecutive DI commands instead of only one. If the first one is disturbed by an interrupt, the return from interrupt will land at the second DI and continue normal program execution.

### 5.1.4 **Possible Risk and Chance to Interrupt DI Execution**

Up to now the described behavior has not been observed in any ROM part. But analyzing this issue, has shown that it might happen in every MARC4 circuit.

It is a time window of about 1 system clock during the execution of the DI command, that an interrupt has to meet to lead to the described behavior.

But neither the part of the program that should be protected by the DI nor the interrupt service routine are than executing in a wrong manner. Even the return from interrupt does not lead to any failure. The program just continues execution.

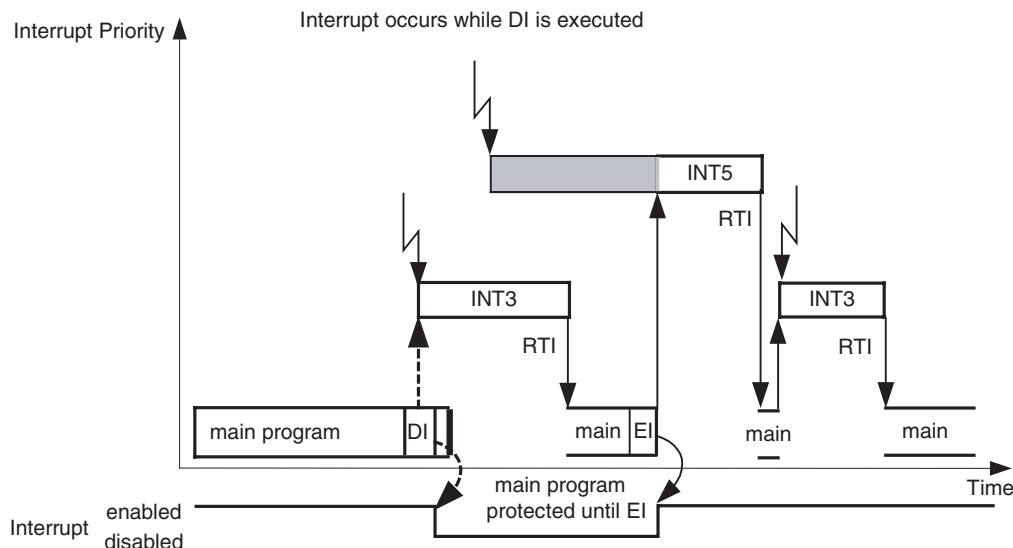
Only if a subport access (consecutive access to 2 registers) that follows the DI is interrupted by another event, may result in an unexpected function. So it is not the first interrupt -- that prevents the DI from being executed correct -- but the next one, following just after the return from interrupt that makes it critical.

Both methods described above are suitable to come around that issue. Method 1 should be preferred if the occurrence of consecutive interrupts or the repetition rate can not be predicted, while method 2 can be used if the programmer is sure, that no 2 interrupts can appear in a way that a subport access might be interrupted.

### 5.1.5 DI Enhancement

The enhancement introduced to the circuit is as follows. Due to the fact that it is not the DI itself that causes possible problems but the interrupt that follows after the return from interrupt (RTI), we modified the function of the RTI instruction. From now on the RTI not longer perform an EI (enable interrupt).

**Figure 5-4.** Enhanced Interrupt Execution



Provided the customer uses the compiler with the default setting \$OPTIMIZE, the compiler takes care about saving the status register (CCR) as well as the index registers (X, Y) when entering an interrupt service routine. It also restores these values when leaving the service routine.

When entering an interrupt service routine normally the interrupt is enabled. Saving the CCR keeps this information. Restoring it at the end of the routine enables the interrupt again. There is no need to perform an additional EI by the RTI instruction.

In case of an interrupt occurring simultaneously to the DI the interrupt flag has the status “disable” when the service routine is entered. This information is saved and restored just before the RTI. Now the interrupt is disabled again and the modified RTI does not enable it. The program to be protected still is protected.

**Note:** If the customer uses the \$NOOPTIMIZE- or the \$OPTIMIZE - SAVECONXT- compiler switch for the interrupt routines the compiler does not save CCR, X, Y registers. In this case it is up to the customer to take care about the interrupt flag.

### 5.1.6 Declaration

The described solution does not require any changes in the application hardware or software as long as the \$NOOPTIMIZE or the \$OPTIMIZE – SAVECONXT switch has not been used for compiling the interrupt service routines.

The described modification is performed by a minor change in the metal-2 layer.

Customers releasing a new firmware for a ROM-mask controller will automatically be set to the latest revision part, having this issue solved.

## 6. Microcontroller - Issue 5: Enhancement of the DI/RTI-combination

Products affected:

ATAM893S, ATAM894P, ATAR092P, ATAR892P, ATAR090H, ATAR890H, ATAR080F, ATA6020N, ATAR510F, ATAM862N3/N4/N8.

The latest product enhancement (DI enhancement) introduced to all MARC4 products during 2003, leads under some conditions to an unforeseen and unwanted behavior. Under special conditions the circuit does not longer process interrupts. Affected are the above mentioned hardware releases.

### 6.1 Description

If a DI-command is immediately followed by a CALL- or SCALL- command and an interrupt meets this DI-command, this interrupt is kept in the interrupt active register permanently despite a correct RTI execution. Even if the EI-command in the interrupted program has been executed, this bit in the interrupt active register stays set and disables the execution of any interrupt of the same or lower priority. Only an interrupt with a higher priority or a reset is able to solve this blocking status.

If the command following the DI is a NOP the interrupt active register seems to be cleared correctly after RTI and executing interrupts continues normally.

#### 6.1.1 Workaround

1. a) Experienced customers using MTP-parts (ATAM893S, ATAM894P) can for example modify the HEX-file directly. Within each interrupt service routine replace the HEX-code for CCR! in front of the RTI with a SCALL to a free address. Introduce the HEX-codes for the commands LIT\_1, OR, CCR! and EXIT at this address. Now, with each RTI the interrupt enable flag is set. This software modification overrides the last hardware modification, i.e. the parts behave as if they had not seen the DI-enhancement and the interrupt active register is cleared correctly.

b1) An equivalent modification can be done in the source code. But the modification depends on what the compiler saved at the beginning of an interrupt service routine:

```
0080 : INT1
0080 0D   CCR@   \$$SAVEREG
0081 73   Y@
0082 72   X@
```

In this case CCR, X and Y registers are saved, therefor the modification should look as follows:

```
2>R 2>R 1 or 2R> 2R>
```

either at the beginning or at the end of the interrupt service routine.

b2) If only CCR and one of the registers is saved:

```
01E0 : INT7
01E0 0D   CCR@   \$$SAVEREG
01E1 73   Y@
```

the modification may look as follows:

```
ROT 1 or <ROT
```

- c) Customers using MTP-parts can switch their order to the predecessor release.

2. Customers using ROM-parts can switch their order to the predecessor release.  
Up to now it is not known, that parts working correctly under all conditions might fail in the future or under different environmental conditions.

### 6.1.2 DI/RTI Enhancement

The DI-modification introduced to all MARC4 circuits, described as “Issue 4”, caused this faulty behavior in conjunction with the RTI command. The measure to remove the EI-function from the RTI resulted in a side effect due to an undetected timing error in the interrupt controllers reset sequence.

Correcting the timing was performed with a change of the internal wiring only. The modified MTP's ATAM893T and ATAM894R have been tested extensively and successfully. These parts are available as samples Q1/2004.

## 7. Microcontroller - Issue 6: FAB Transfer of Microcontroller

Products affected: ATAR892, ATAR092, ATAR862

In order to offer a second source for MARC4 products the design of the M44C092/ATAR092 was transferred to a FAB in France. The Z96E process (0.5  $\mu\text{m}$ ) at MHS SAS in Nantes (former Atmel/Nantes) is quite comparable to the original TSMC process (0.6  $\mu\text{m}$ ) and required only slight modifications of the design. The modifications confined to adaptations due to parametric differences (resistors, threshold voltages, capacitors for oscillators and voltage references).

The production started beginning of 2006 in Nantes/France. The circuit is fully qualified and compatible with the existing MARC4 products. It includes the enhancements "Microcontroller - Issue 1" to "Microcontroller - Issue 5" as well as "EE Issue - 1" to "EE - Issue 3" for parts containing the EEPROM T505M as described in this paper. The corresponding MTP is the ATAM893T, that can be configured accordingly.

The datasheets remain unchanged. PPAP's are updated accordingly.

## 8. EE - Issue 1: FAB Transfer of EEPROM

Products affected: M44C890, M44C892, U9280M

Due to the fact, that TSMC has closed their FAB1, the EEPROM process used for U505M and the U3280M has been transferred to TSMC FAB2. These chips are part of the dual die ICs mentioned above. Wafer parameters will be unchanged. The masks were copied 1:1 for the FAB2 equipment.

There is no change in wafer probing and final testing.

Valid datasheet and IC functionality are not affected.

The IC marking will be unchanged.

## 9. EE - Issue 2: Improve Process Variation Robustness

Products affected: ATAR890, ATAR892, M44C890, M44C892, ATAR862

### 9.1 Issue

Customers complained a reduced lifetime of the EEPROM circuit T505M-O used in all multi die ICs listed above.

### 9.2 Description

Together with TSMC we found out, that some charge pump capacitors have been damaged. This led to the effect, that the EEPROM programming was no longer possible. Deeper analysis showed that an under etching due to process variation damaged the capacitors.

### 9.3 Enhancement

In order to make the physical layout of the charge pump capacitor more robust against process variations at the metal dry etching, new masks were created. The metal layer now covers the dielectric layer of the charge pump capacitor to prevent an under etching of the dielectric layer.

The new revision T505M-P is qualified and the PPAPs are updated accordingly.

## 10. EE - Issue 3: Improve EEPROM EMC Robustness

Products affected: ATAR892, ATAR890, ATAR862

### 10.1 Issue

Applications using MARC4 products of the families ATAR892, ATAR890 and ATAR862 are sometimes used under very harsh environmental conditions. A few customers complained that the EEPROM sometimes contained corrupted data.

### 10.2 Description

The long and deep analysis showed, that the issue was triggered from system level ESD pulses, significantly higher than specified, resulting in EMI. This EMI induced a high voltage to the internal supply of the EEPROM. The high voltage was able to force a Flip Flop of the test register into the opposite state, leading to the effect that in this test mode some data were corrupted after one of the following write commands.

### 10.3 Enhancement

Introducing a better blocking and noise suppression to the supply lines of the board by means of a small L/C-filter solved the problem on application level.

But in order to support the robustness of the system design against electromagnetic influence also the test mode controller of the EEPROM chip T505M was changed slightly. The change prevents the multi chip modules from entering and keeping a test mode due to electromagnetic influence to the supply lines or the chip-to-chip serial interface.

The changes are as follows:

1. The time window to activate the test mode is limited to the first TWI command after any reset.
2. Only if this first command has set the Test-Enable-Flip-Flop is the second TWI command able to enter a test mode.
3. The validity of the test mode period is restricted to the third TWI command which contains the data for the test mode selected with the second command.
4. The test register is reset permanently if the fourth TWI command is an application mode command.

Only after the next reset (Power on reset, brown out detection, watch dog or clock supervisor that might be triggered for example due to battery change, ESD/EMI event, etc) is it possible to reactivate the test mode controller.

All datasheets remain unchanged. The improved T505M-R is already qualified and the PPAP is updated accordingly.

The new product revisions are: ATAR892V/W, ATAR890P, ATAR862R/S

Starting Oct. 2006, customers releasing a new firmware for a ROM-mask controller will automatically be set to the latest product revision. On customers' request, existing ROM-mask controllers using the old EEPROM revisions T505M-O/P can be switched to the improved product revision without the need of a firmware modification.

## 11. Revision History

Please note that the following page numbers referred to in this section refer to the specific revision mentioned, not to this document.

Revision No.	History
4668D-4BMCU-02/08	<ul style="list-style-type: none"><li>• Issues on page changed</li><li>• Section 1 “Introduction” on page 1 changed</li><li>• Table 1-1 “Hardware Revisions” on pages 2 to 3 changed</li><li>• Section 7 “Microcontroller - Issue 6: FAB Transfer of Microcontroller” on page 17 changed</li><li>• Section 8 “EE - Issue 1: FAB Transfer of EEPROM” on page 17 added</li><li>• Section 9 “EE - Issue 2: Improve Process Variation Robustness” on page 17 added</li><li>• Section 10 “EE - Issue 3: Improve EEPROM EPC Robustness” on page 18 added</li></ul>
4668C-4BMCU-12/04	<ul style="list-style-type: none"><li>• Issues on page 1 changed</li><li>• Table 1 “Hardware Revisions” on page 2 changed</li><li>• Section “Issue 6: FAB Transfer of Microcontroller and EEPROM” on page 15 added</li></ul>
4668B-4BMCU-04/04	<ul style="list-style-type: none"><li>• Issues on page 1 changed</li><li>• Table 1 “Hardware Revisions” on page 2 changed</li><li>• Section “DI Enhancement” on pages 12-13 changed</li><li>• Section “Declaration” on page 13 changed</li><li>• Section “Issue 5: Enhancement of the DI/RTI-combination” on pages 14 to 15 added</li></ul>



## Headquarters

---

**Atmel Corporation**  
2325 Orchard Parkway  
San Jose, CA 95131  
USA  
Tel: 1(408) 441-0311  
Fax: 1(408) 487-2600

## International

---

**Atmel Asia**  
Room 1219  
Chinachem Golden Plaza  
77 Mody Road Tsimshatsui  
East Kowloon  
Hong Kong  
Tel: (852) 2721-9778  
Fax: (852) 2722-1369

**Atmel Europe**  
Le Krebs  
8, Rue Jean-Pierre Timbaud  
BP 309  
78054  
Saint-Quentin-en-Yvelines Cedex  
France  
Tel: (33) 1-30-60-70-00  
Fax: (33) 1-30-60-71-11

**Atmel Japan**  
9F, Tonetsu Shinkawa Bldg.  
1-24-8 Shinkawa  
Chuo-ku, Tokyo 104-0033  
Japan  
Tel: (81) 3-3523-3551  
Fax: (81) 3-3523-7581

## Product Contact

---

**Web Site**  
[www.atmel.com](http://www.atmel.com)

**Technical Support**  
[marc4@atmel.com](mailto:marc4@atmel.com)

**Sales Contact**  
[www.atmel.com/contacts](http://www.atmel.com/contacts)

**Literature Requests**  
[www.atmel.com/literature](http://www.atmel.com/literature)

---

**Disclaimer:** The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2008 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.