Hints and Tips for Hard- and Software Developments with MARC4 Microcontrollers

Programming Hints

- **Use of the SLEEP Instruction**
- **Oscillator Selection**
- **Access to Subport Registers**
- **Access to AUX Registers**
- **Unused Interrupts**

Application Hints

- **Use of BP20/NTE**

Development Hints

- **U9280M MTP, EMU-chip – EEPROM Block Read**
- **Settings for different MARC4 Hardware Versions**

Description

This application note provides MARC4 user programming hints that help to get the software running, and includes the description of some specific MARC4 hardware features. Specific hints on the MARC4 development are also given.

MARC4

Application Note

Rev. 4715D–4BMCU–04/04

Programming Hints

Use of the SLEEP Instruction

Products: ATA6020N, ATAR080, ATAR090, ATAR890, ATAR092, ATAR892, ATAR510, ATAM510, ATAM893, ATAM894, U9280M

Most applications use the SLEEP instruction only in the $AUTOSLEEP routine. This routine is automatically inserted by the qForth compiler and works without any problems.

If an application software uses the SLEEP instruction during the main program, the following must be observed:

To ensure proper execution of a SLEEP instruction following an I/O instruction, it is necessary to not insert I/O instructions between the IN or OUT command and the SLEEP command. The ATARx9x family requires 2 instructions, the ATAM893 requires 3 instructions.

We recommend you use 3 instructions (for example NOP NOP NOP) in both cases.

Example:
```
5 P2DAT OUT
NOP NOP NOP
SLEEP
```

Oscillator Selection

Products: ATA6020N, ATAR080, ATAR090, ATAR890, ATAR092, ATAR892, ATAR510, ATAM510, ATAM893, ATAM894, U9280M

The MARC4 devices contain a clock module with 4 different internal oscillators. The selection of the oscillator used is done by two registers, the Clock Management register (CM) and the System Configuration register (SC). The clock module incorporates a synchronization stage which allows it to switch from one oscillator to another during runtime.

To switch from the internal RC-oscillator to one of the crystal oscillators, the software has to first select the crystal oscillator by writing to the SC register. The software has to ensure the start-up time of the selected oscillator. Then the software switches the system clock from the internal RC-oscillator to the crystal oscillator by setting the CCS bit to 0 in the CM register.

Example:
```
0010b SC OUT        \ BOT = 2.0 V; select 4 MHz crystal oscillator
WaitStartup         \ wait or execute some not timing critical code
0001b CM AUX_OUT    \ switch system clock to crystal oscillator
```

Access to Subport Registers

Products: ATA6020N, ATAR080, ATAR090, ATAR890, ATAR092, ATAR892, ATAR510, ATAM510, ATAM893, ATAM894, U9280M

To read or write data to a subport register, two I/O instructions are necessary. It is possible that an interrupt occurs between these two I/O instructions. The first I/O instruction, which addresses the subport register, will be executed by the core, and then the interrupt gets active. If the interrupt service routine also reads or writes data to a subport register, the synchronization of the subport gets lost. To prevent this, it is useful to disable such interrupts.

Example:
```
CCR@  \ save current setting of interrupt enable flag
DI    \ disable all interrupts
Address-pointer Subport-Address OUT \ write address pointer
Data Subport-Address OUT \ write data to subaddress register
CCR!
```

The file C092.inc, which is delivered with the MARC4 development tools, uses the DI instruction to disable any interrupt during a subport access.

Access to AUX Registers

Products: ATA6020N, ATAR080, ATAR090, ATAR890, ATAR092, ATAR892, ATAR510, ATAM510, ATAM893, ATAM894, U9280M

Before the data of an auxiliary register can be read or written, the software has to write the address to the Auxiliary Switch Register (ASW). This enables the access to the auxiliary registers. If an interrupt occurs during this access, and if the interrupt also includes an access to an auxiliary register, then wrong data will be written to the first addressed module.

To prevent the auxiliary registers from receiving wrong data, it is useful to disable interrupts during the access mode.

Example:
```
CCR@ <ROT       (Data Pointer-- CCR Data Pointer)
DUP DI ASW
OUT             \ Set auxiliary reg. pointer
OUT             (CCR Data Pointer-- CCR)
CCR!            (CCR --)
```

The file C092.inc version 2.0 and higher uses the DI instruction to disable any interrupt during access mode.

Unused Interrupts

Products: ATA6020N, ATAR080, ATAR090, ATAR890, ATAR092, ATAR892, ATAR510, ATAM510, ATAM893, ATAM894, U9280M

Normally unused interrupts do not occur.

Nevertheless, due to strong ESD pulses there is a theoretical possibility that an interrupt flag in the interrupt pending register gets set. If the interrupts are enabled and no higher interrupt is active the MARC4 core jumps to the address of the unused interrupt service routine. This could result in a jump to any instruction of a routine on this address.

To avoid such behavior it is useful to define all interrupts.

Example:
```
:Int3   \ definition of unused interrupt
;       \ RTI
```

Application Hints

Use of Port BP20/NTE

Products: ATA6020N, ATAR080, ATAR090, ATAR890, ATAR092, ATAR892, ATAM893, ATAM894, U9280M

During any reset phase, the BP20/NTE input is driven towards V_{DD} by a strong internal pull-up transistor. This pin must not be pulled down (active or passive) to V_{SS} during reset by any external circuitry representing a resistor of less than 150 kΩ. This prevents the circuit from switching to test mode enable by accident through the application circuitry at pin BP20/NTE. Resistors less than 150 kΩ might lead to an undefined state of the internal test logic thus stopping the application firmware.

Development Hints

EEPROM Block Read

Products: M4EMUx9x - Emulation Chip

The emulation chip is a two chips in one ceramic package solution. It consists of the ATAM893 and the U3280M, which is a transponder front end with EEPROM.

After power-on, the block read of the EEPROM data could result in wrong data. To ensure proper data the first three read instructions should be single read instructions. After these three single read instructions the block read works in a normal way.

Example:

```
$Include EEP_89x.inc          \ use EEPROM library
ROW00_RL EEPROM4@             \ read data Row00 from EEPROM
ROW01_RL EEPROM4@
ROW02_RL EEPROM4@
```

Settings for Different MARC4 Hardware Versions

Products: ATA6020N, ATAR080, ATAR090, ATAR890, ATAR092, ATAR892, ATAM893, ATAM894

With the first download after power-on of the emulation hardware, the MARC4 core definition is loaded into the CPLD on the emulator POD. When selecting the option “Use default”, the standard default settings as integrated in “winIDEA” is used. After modifying the MARC4 hardware, it may be necessary to use another definition file. This can be done by selecting “Load from external file”. The new core file must be placed in the same directory as the file “winIDEA.exe”.

The MARC4 hardware version is defined by a specific letter following the part name. The right setting for the used target chip is shown in Table 1 on page 5.

Table 1. Emulator CPU Setup

Target Chip	Use Default	Load from External File
ATAR080 Version < G	X	
ATAR080 Version ≥ G		X
ATARx90 Version < K	X	
ATARx90 Version ≥ K		X
ATARx92 Version < Q	X	
ATARx92 Version ≥ Q		X
ATAM893 Version < T	X	
ATAM893 Version ≥ T		X
ATAM894 Version < R	X	
ATAM894 Version ≥ R		X
ATA6020N Version < P	X	
ATA6020 Version ≥ P		X

The core definition file, called “C092.hex””, must be placed in the same directory as the “winIDEA.exe”. It can also be downloaded from Atmel’s website “www.atmel.com”.

The MTP programmer software stores the settings in a configuration file called “TemICP.ini”. Inside this configuration file a section [Default] is used to enable special test modes. This section contains 3 entries:

Def 1012 = 255

Def 1013 = 255

The 3rd entry, Def 1009, must have the value according to Table 2

Table 2. Setting of Def 1009

Target Chip	Def1009
T48C893/ATAM893 Version < T	255
T48C893/ATAM893 Version ≥ T	223
T48C894/ATAM894 Version < R	255
T48C894/ATAM894 Version ≥ R	223

Revision History

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

Changes from Rev. 4715C - 03/04 to Rev. 4715D - 04/04

1. Section “Settings for Different MARC4 Hardware Versions” on page 4 changed.

Atmel Corporation

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

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

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

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

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

4715D–4BMCU–04/04