# AVR32717: Compatibility Note AT32UC3Ax Revision E to Revision H or later

## 1 Introduction

The AT32UC3Ax revision H introduces fixes and incompatibility with the previous revision E. This document outlines the software operations to migrate software from an AT32UC3Ax revision E, also known as engineering sample (ES), to AT32UC3Ax revision H and later.

The AVR®32Studio, the AVR32 GNU Toolchain and the Software Framework have also been upgraded. The 2nd part of this document also outlines the software operations necessary to migrate a software project from older to newer tools.

32-bit AVR® Microcontrollers

Application Note

Rev. 32102C-AVR32-08/08

# 2 Parts Concerned

**Table 2-1.** List of Concerned Parts

| AT32UC3Ax Engineering Samples Rev E | AT32UC3Ax Production parts Rev H and later |
|---|---|
| | AT32UC3A0512 Rev H and later |
| | AT32UC3A0256 Rev H and later |
| | AT32UC3A0128 Rev H and later |
| AT32UC3A0512 Rev E | AT32UC3A1512 Rev H and later |
| AT32UC3A1512 Rev E | AT32UC3A1256 Rev H and later |
| | AT32UC3A1128 Rev H and later |

## 2.1 How to differentiate Device Revisions

### 2.1.1 Package Label

- The packages of the engineering samples are labeled with “-xES”. 32UC3Axxxx-UES means it is a Rev E.
- The packages of industrial production parts (non engineering samples) are labeled with “–U” (used for production of industrial parts).

In the following of this document, we will use the “ES” notation to designate engineering samples parts (ie. Rev E).

### 2.1.2 Device Identification Register

The JTAG Device Identification register (DID) contains the device chip revision (“RN” field, revA = 0x0 revB = 0x01 etc…).

*Note: Refer to device datasheet section JTAG and Boundary Scan for the JTAGID code description.*

This is the C code to read the JTAG DID register in software, for IAR® or GCC compiler:

For IAR compiler:

```
#include <avr32/io.h>
#include <intrinsics.h>
static inline unsigned int read_jtag_id(void) {
      return __get_debug_register(AVR32_DID);
}
```

For GCC compiler:

```
#include <avr32/io.h>
static inline unsigned int read_jtag_id(void) {
      return __builtin_mfdr(AVR32_DID);
}
```

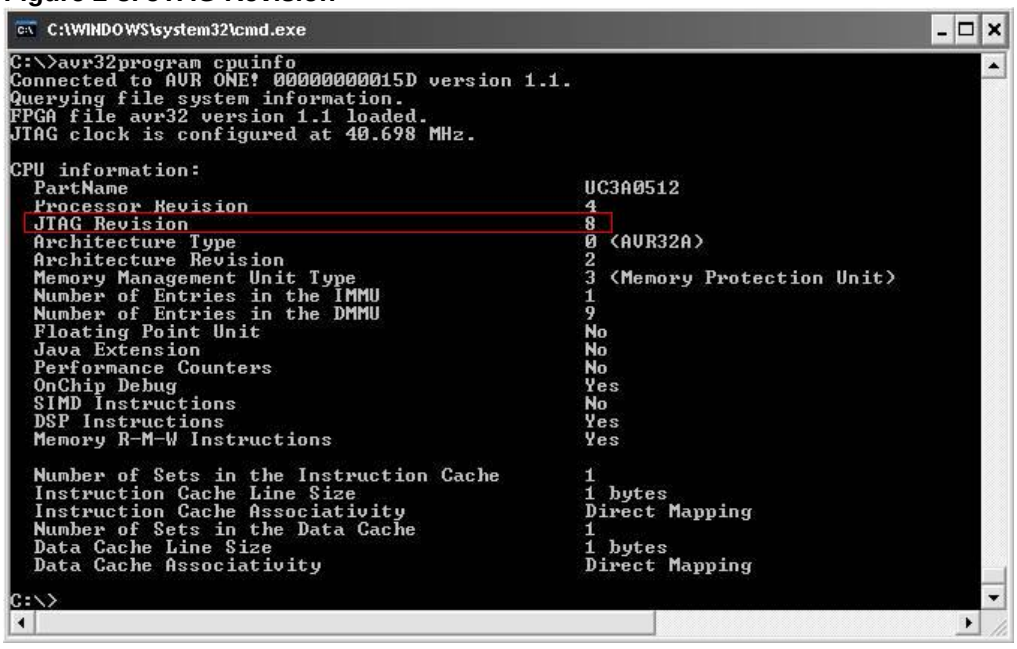
**Table 2-2. JTAG DID Register Description**

| R/W | Bit Number | Field Name | Init. Val. | Description |
|---|---|---|---|---|
| R | 31:28 | RN | Part specific | RN – Revision Number |
| R | 27:12 | PN | Part specific | PN – Product Number |
| R | 11:1 | MID | 0x01F | Manufacturer ID<br>0x01F = ATMEL |
| R | 0 | Reserved | 1 | Reserved<br>This bit always reads as 1. |

### 2.1.3 Using the JTAGICEmkII or the AVR One!

Avr32program is a utility from the AVR32 GNU toolchain which can read the JTAG ID revision from a command line window (“avr32program cpuinfo”, read the “JTAG Revision” line).

**Figure 2-3. JTAG Revision**




JTAG revision: revE = 0x4, revH = 0x7, revI= 0x8.

# 3 Software Migration

The errata section in the device datasheet provides the list of known issues per revision. This migration note does not address every item of the datasheet errata. Only the items affecting software binary compatibility are addressed in the following.

## 3.1 CPU Cycle Counter

On Rev E:

1. The CPU Cycle Counter does not reset the COUNT system register on COMPARE match.
2. The COUNT register is clocked by the CPU clock, so when the CPU clock stops, so does incrementing of COUNT.

On Rev H:

1. The CPU cycle counter reset the COUNT system register on COMPARE match.
2. The COUNT register has a dedicated clock active only in active mode and in the following sleep modes: IDLE, FROZEN and STANDBY.

Migration from Rev E to Rev H or later:

1. Any software which relied on the fact that the COUNT system register was not reset on COMPARE match must be updated to reflect the fact that the COUNT system register is now reset on COMPARE match.
2. Any software which relied on the fact that COUNT register was not incremented in any of the sleep modes must be updated to reflect the fact that the COUNT register is now incremented in the following sleep modes: IDLE, FROZEN and STANDBY.

Software implementation examples can be found in the AVR32 UC3 software framework under the directory DRIVERS/CPU/CYCLE_COUNTER/EXAMPLE.

## 3.2 Timer Counter

On Rev E, the following timer counter clock inputs are different than in Rev H and later.

On Rev E:

- TIMER_CLOCK2 is connected to PBA Clock / 4
- TIMER_CLOCK4 is connected to PBA Clock / 16
- TIMER_CLOCK5 is connected to PBA Clock / 32

On Rev H:

- TIMER_CLOCK2 is connected to PBA Clock / 2
- TIMER_CLOCK4 is connected to PBA Clock / 32
- TIMER_CLOCK5 is connected to PBA Clock / 128

Migration from Rev E to Rev H or later:

Any software that was relying on the previous timer clock input frequency must be updated to match the new timer clock input frequency.

Software implementation examples can be found in the AVR32 UC3 software framework under the directory DRIVERS/TC.

## 3.3 Flashc

### 3.3.1 General Purpose Fuse Register Low Address

On Rev E the address of the Flash General Purpose Fuse Register Low (FGPFRLO, previously named FPGPFR) was 0xFFFE140C.
On Rev H the address of the Flash General Purpose Fuse Register Low (FGPFRLO) is 0xFFFE1410.

Migration from Rev E to Rev H or later:

Any software that hard coded the FGPFRLO address value must be updated with the new FGPFRLO address. If the software was relying on the flash header file definition, FPGPFR should be renamed to FGPFRLO.

### 3.3.2 PAGEN Semantic Field for Program GP Fuse Byte

On Rev E, the PAGEN semantic field for Program GP Fuse Byte was WriteData[7:0], ByteAddress[1:0].
On Rev H, the PAGEN semantic field for Program GP Fuse Byte is WriteData[7:0], ByteAddress[2:0].

Migration from Rev E to Rev H or later:

Any software that relied on the PAGEN Rev E semantic field must be updated with the Rev H semantic field.

Software implementation examples can be found in the AVR32 UC3 software framework under the directory DRIVERS/FLASHC.

## 3.4 USB

On Rev E, the UHADDR1/2/3 registers are not available. UHCON.HADDR (UHCON[6:0]) field register is used for the USB device address whatever the pipe.
On Rev H, the HCON.HADDR (UHCON[6:0]) field register is not available. UHADDR1/2/3 registers are used to select one USB device address per pipe.

Migration from Rev E to Rev H or later:

Any software that relied on the UHCON.HADDR (UHCON[6:0]) field register must use the UHADDR1/2/3.UHADDR_Px instead: fill all the UHADDRx.UHADDR_Py with the value that was in the corresponding UHCON.HADDR.

Software implementation examples can be found in the AVR32 UC3 software framework under the directory DRIVERS/USBB.

## 3.5 Real Time Counter (RTC)

On Rev E, the RTC CLKEN bit (bit number 16) of CTRL register is not available.

On Rev H, the RTC CLKEN bit of CTRL register is available and has to be set in order to enable the RTC.

Migration from Rev E to Rev H or later:

Any software that used the RTC on Rev E must now use in Rev H the CLKEN bit of RTC CTRL register to enable the RTC.

Software implementation examples can be found in the AVR32 UC3 software framework under the directory DRIVERS/RTC.

# 4 Tools

## 4.1 AVR32Studio and GNU Toolchain 2.0 and later

The device header files set is different for ES parts compared with non-ES parts. To manage it in the GNU toolchain, the following targets are supported:

### 4.1.1 UC3A Part Name

With GCC compiler, the 'mpart' option should be set to one of the following:

- UC3A0512ES: Rev E.
- UC3A1512ES: Rev E.
- UC3A0512: Rev H and later.
- UC3A0256: Rev H and later.
- UC3A0128: Rev H and later.
- UC3A1512: Rev H and later.
- UC3A1256: Rev H and later.
- UC3A1128: Rev H and later.

### 4.1.2 Header Files Organization

The header files addressed in this section can be found for Windows® user under the directory: C:\Program Files\Atmel\AVR Tools\AVR32 Toolchain\avr32\include\avr32. They are installed within the GNU Toolchain release.

- avr32/io.h file: this file is the entry point and is used to include the appropriate other header files according to the device chosen. This file is found in the GNU toolchain.

```
<snip>
#  elif __AVR32_UC3A0512ES__
#    include <avr32/uc3a0512es.h>
#  elif __AVR32_UC3A1512ES__
#    include <avr32/uc3a1512es.h>
<snip>
#  elif __AVR32_UC3A0512__
#    include <avr32/uc3a0512.h>
<snip>
```

- Device specific files: avr32/uc3a0512.h, avr32/uc3a0512es.h … included by io.h according to the chosen device.
- IP specific header files: avr32/usart_319.h, avr32/usart_400.h… included by the 'device'.h specific header file.

### 4.1.3 Project Migration from AVR32 GNU Toolchain 1.y.z to AVR32 GNU Toolchain 2.0 or later

In AVR32 GCC toolchain version 2.0 and later, some headers files definitions have been renamed. When migrating any software from older toolchain version 1.y.z to toolchain 2.0 and later, the following definitions must be updated:

**Table 4-1.** Project Migration from AVR32 GNU Toolchain 1.y.z to AVR32 GNUToolchain 2.0 or later

| Old Definition in Toolchain version 1.y.z | New definition in Toolchain version 2.0 or later |
|---|---|
| Interrupt Controller header file: | |
| AVR32_INTC_IPRx_INTLEV_OFFSET | AVR32_INTC_IPR_INTLEV_OFFSET |
| AVR32_INTC_IPRx_INTLEV_MASK | AVR32_INTC_IPR_INTLEV_MASK |
| AVR32_INTC_IPRx_INTLEV_SIZE | AVR32_INTC_IPR_INTLEV_SIZE |
| AVR32_INTC.icr3 | AVR32_INTC.icr |
| Flashc header file: | |
| AVR32_FLASHC_FGPFR_GPFx | AVR32_FLASHC_FGPFR_LOCKx |
| AVR32_FLASHC_FGPFR_GPFx_MASK | AVR32_FLASHC_FGPFR_LOCKx_MASK |
| AVR32_FLASHC_FGPFR_GPFx_SIZE | AVR32_FLASHC_FGPFR_LOCKx_SIZE |
| AVR32_FLASHC_FGPFR_GPFx_OFFSET | AVR32_FLASHC_FGPFR_LOCKx_OFFSET |
| PWM header file: | |
| AVR32_PWM_PWM_LINES_MSB | AVR32_PWM_LINES_MSB |
| AVR32_PWM_PWM_0_PIN | AVR32_PWM_0_PIN |
| AVR32_PWM_PWM_0_FUNCTION | AVR32_PWM_0_FUNCTION |
| AVR32_PWM_PWM_1_PIN | AVR32_PWM_1_PIN |
| AVR32_PWM_PWM_1_FUNCTION | AVR32_PWM_1_FUNCTION |
| AVR32_PWM_PWM_2_PIN | AVR32_PWM_2_PIN |
| AVR32_PWM_PWM_2_FUNCTION | AVR32_PWM_2_FUNCTION |
| AVR32_PWM_PWM_3_PIN | AVR32_PWM_3_PIN |
| AVR32_PWM_PWM_3_FUNCTION | AVR32_PWM_3_FUNCTION |
| AVR32_PWM_PWM_4_0_PIN | AVR32_PWM_4_0_PIN |
| AVR32_PWM_PWM_4_0_FUNCTION | AVR32_PWM_4_0_FUNCTION |
| AVR32_PWM_PWM_5_0_PIN | AVR32_PWM_5_0_PIN |
| AVR32_PWM_PWM_5_0_FUNCTION | AVR32_PWM_5_0_FUNCTION |
| AVR32_PWM_PWM_4_1_PIN | AVR32_PWM_4_1_PIN |
| AVR32_PWM_PWM_4_1_FUNCTION | AVR32_PWM_4_1_FUNCTION |
| AVR32_PWM_PWM_5_1_PIN | AVR32_PWM_5_1_PIN |
| AVR32_PWM_PWM_5_1_FUNCTION | AVR32_PWM_5_1_FUNCTION |
| AVR32_PWM_PWM_6_PIN | AVR32_PWM_6_PIN |
| AVR32_PWM_PWM_6_FUNCTION | AVR32_PWM_6_FUNCTION |
| Power Manager header file: | |
| AVR32_PM_GCLK_USB | AVR32_PM_GCLK_USBB |
| RTC: | |
| AVR32_RTC_RTC_IRQ | AVR32_RTC_IRQ |

Note 1: The UC3 software framework version 1.1.1 is implemented with definitions from the left column.

Note 2: The UC3 software framework version 1.2.z and 1.3.z or later are implemented with definitions from the right column.

Project Migration from Toolchain 1.y.z to Toolchain 2.0 or later:

Any software that relied on the Toolchain 1.y.z must be updated with the header definition from the Toolchain 2.0 or later version.

### 4.1.4 Project Migration from UC3Ax512ES Target (Rev E) to UC3Ax512 Target (Rev H or later) in Toolchain 2.x

The UC3A0512 (Rev H and later) header files fix the errata of the new available alternate functions on some pins. These pins were not available for UC3A0512ES (Rev E). To migrate a project from UC3A0512ES (Rev E) to UC3A0512 (Rev H or later), the following defines must be updated:

**Table 4-2.** Project Migration from UC3Ax512ES Target (Rev E) to UC3Ax512 Target (Rev H or later) in Toolchain 2.x

| UC3Ax512ES (Rev E) in Toolchain 2.x | | UC3Ax512 (Rev H) in Toolchain 2.x | |
|---|---|---|---|
| SPI0: | | SPI0: | |
| #define AVR32_SPI0_MISO_0_PIN | 11 | #define AVR32_SPI0_MISO_0_0_PIN | 11 |
| #define AVR32_SPI0_MISO_0_FUNCTION | 0 | #define AVR32_SPI0_MISO_0_0_FUNCTION | 0 |
| #define AVR32_SPI0_MOSI_0_PIN | 12 | #define AVR32_SPI0_MOSI_0_0_PIN | 12 |
| #define AVR32_SPI0_MOSI_0_FUNCTION | 0 | #define AVR32_SPI0_MOSI_0_0_FUNCTION | 0 |
| #define AVR32_SPI0_NPCS_0_PIN | 10 | #define AVR32_SPI0_NPCS_0_0_PIN | 10 |
| #define AVR32_SPI0_NPCS_0_FUNCTION | 0 | #define AVR32_SPI0_NPCS_0_0_FUNCTION | 0 |
| #define AVR32_SPI0_NPCS_1_PIN | 8 | #define AVR32_SPI0_NPCS_1_0_PIN | 8 |
| #define AVR32_SPI0_NPCS_1_FUNCTION | 1 | #define AVR32_SPI0_NPCS_1_0_FUNCTION | 1 |
| #define AVR32_SPI0_NPCS_2_PIN | 9 | #define AVR32_SPI0_NPCS_2_0_PIN | 9 |
| #define AVR32_SPI0_NPCS_2_FUNCTION | 1 | #define AVR32_SPI0_NPCS_2_0_FUNCTION | 1 |
| #define AVR32_SPI0_NPCS_3_PIN | 7 | #define AVR32_SPI0_NPCS_3_1_PIN | 7 |
| #define AVR32_SPI0_NPCS_3_FUNCTION | 2 | #define AVR32_SPI0_NPCS_3_1_FUNCTION | 2 |
| #define AVR32_SPI0_SCK_0_PIN | 13 | #define AVR32_SPI0_SCK_0_0_PIN | 13 |
| #define AVR32_SPI0_SCK_0_FUNCTION | 0 | #define AVR32_SPI0_SCK_0_0_FUNCTION | 0 |
| SPI1: | | SPI1: | |
| #define AVR32_SPI1_MISO_0_PIN | 17 | #define AVR32_SPI1_MISO_0_0_PIN | 17 |
| #define AVR32_SPI1_MISO_0_FUNCTION | 1 | #define AVR32_SPI1_MISO_0_0_FUNCTION | 1 |
| #define AVR32_SPI1_MOSI_0_PIN | 16 | #define AVR32_SPI1_MOSI_0_0_PIN | 16 |
| #define AVR32_SPI1_MOSI_0_FUNCTION | 1 | #define AVR32_SPI1_MOSI_0_0_FUNCTION | 1 |
| #define AVR32_SPI1_NPCS_0_PIN | 14 | #define AVR32_SPI1_NPCS_0_0_PIN | 14 |
| #define AVR32_SPI1_NPCS_0_FUNCTION | 1 | #define AVR32_SPI1_NPCS_0_0_FUNCTION | 1 |
| #define AVR32_SPI1_NPCS_1_PIN | 18 | #define AVR32_SPI1_NPCS_1_0_PIN | 18 |
| #define AVR32_SPI1_NPCS_1_FUNCTION | 1 | #define AVR32_SPI1_NPCS_1_0_FUNCTION | 1 |
| #define AVR32_SPI1_NPCS_2_PIN | 19 | #define AVR32_SPI1_NPCS_2_0_PIN | 19 |
| #define AVR32_SPI1_NPCS_2_FUNCTION | 1 | #define AVR32_SPI1_NPCS_2_0_FUNCTION | 1 |
| #define AVR32_SPI1_SCK_0_PIN | 15 | #define AVR32_SPI1_SCK_0_0_PIN | 15 |
| #define AVR32_SPI1_SCK_0_FUNCTION | 1 | #define AVR32_SPI1_SCK_0_0_FUNCTION | 1 |
| USART0: | | USART0: | |

| | |
|---|---|
| #define AVR32_USART0_CTS_0_PIN 4<br>#define AVR32_USART0_CTS_0_FUNCTION 0<br>#define AVR32_USART0_RTS_0_PIN 3<br>#define AVR32_USART0_RTS_0_FUNCTION 0<br>#define AVR32_USART0_RXD_0_PIN 0<br>#define AVR32_USART0_RXD_0_FUNCTION 0<br>#define AVR32_USART0_TXD_0_PIN 1<br>#define AVR32_USART0_TXD_0_FUNCTION 0 | #define AVR32_USART0_CTS_0_0_PIN 4<br>#define AVR32_USART0_CTS_0_0_FUNCTION 0<br>#define AVR32_USART0_RTS_0_0_PIN 3<br>#define AVR32_USART0_RTS_0_0_FUNCTION 0<br>#define AVR32_USART0_RXD_0_0_PIN 0<br>#define AVR32_USART0_RXD_0_0_FUNCTION 0<br>#define AVR32_USART0_TXD_0_0_PIN 1<br>#define AVR32_USART0_TXD_0_0_FUNCTION 0 |
| USART1:<br><br>#define AVR32_USART1_CTS_0_PIN 9<br>#define AVR32_USART1_CTS_0_FUNCTION 0<br>#define AVR32_USART1_RTS_0_PIN 8<br>#define AVR32_USART1_RTS_0_FUNCTION 0<br>#define AVR32_USART1_RXD_0_PIN 5<br>#define AVR32_USART1_RXD_0_FUNCTION 0<br>#define AVR32_USART1_TXD_0_PIN 6<br>#define AVR32_USART1_TXD_0_FUNCTION 0 | USART1:<br><br>#define AVR32_USART1_CTS_0_0_PIN 9<br>#define AVR32_USART1_CTS_0_0_FUNCTION 0<br>#define AVR32_USART1_RTS_0_0_PIN 8<br>#define AVR32_USART1_RTS_0_0_FUNCTION 0<br>#define AVR32_USART1_RXD_0_0_PIN 5<br>#define AVR32_USART1_RXD_0_0_FUNCTION 0<br>#define AVR32_USART1_TXD_0_0_PIN 6<br>#define AVR32_USART1_TXD_0_0_FUNCTION 0 |
| USART3:<br><br>#define AVR32_USART3_RXD_0_PIN 42<br>#define AVR32_USART3_RXD_0_FUNCTION 1<br>#define AVR32_USART3_TXD_0_PIN 43<br>#define AVR32_USART3_TXD_0_FUNCTION 1 | USART3:<br><br>#define AVR32_USART3_RXD_0_0_PIN 42<br>#define AVR32_USART3_RXD_0_0_FUNCTION 1<br>#define AVR32_USART3_TXD_0_0_PIN 43<br>#define AVR32_USART3_TXD_0_0_FUNCTION 1 |
| Timer Counter (TC):<br><br>#define AVR32_TC_A0_0_PIN 55<br>#define AVR32_TC_A0_0_FUNCTION 0<br>#define AVR32_TC_A1_0_PIN 57<br>#define AVR32_TC_A1_0_FUNCTION 0<br>#define AVR32_TC_A2_0_PIN 59<br>#define AVR32_TC_A2_0_FUNCTION 0<br>#define AVR32_TC_B0_0_PIN 56<br>#define AVR32_TC_B0_0_FUNCTION 0<br>#define AVR32_TC_B1_0_PIN 58<br>#define AVR32_TC_B1_0_FUNCTION 0<br>#define AVR32_TC_B2_0_PIN 60<br>#define AVR32_TC_B2_0_FUNCTION 0 | Timer Counter (TC):<br><br>#define AVR32_TC_A0_0_0_PIN 55<br>#define AVR32_TC_A0_0_0_FUNCTION 0<br>#define AVR32_TC_A1_0_0_PIN 57<br>#define AVR32_TC_A1_0_0_FUNCTION 0<br>#define AVR32_TC_A2_0_0_PIN 59<br>#define AVR32_TC_A2_0_0_FUNCTION 0<br>#define AVR32_TC_B0_0_0_PIN 56<br>#define AVR32_TC_B0_0_0_FUNCTION 0<br>#define AVR32_TC_B1_0_0_PIN 58<br>#define AVR32_TC_B1_0_0_FUNCTION 0<br>#define AVR32_TC_B2_0_0_PIN 60<br>#define AVR32_TC_B2_0_0_FUNCTION 0 |
| Interrupt Controller header file:<br><br>AVR32_INTC_IPR0_INTLEV_OFFSET<br>AVR32_INTC_IPR0_INTLEV_MASK | <br><br>AVR32_INTC_IPR_INTLEV_OFFSET<br>AVR32_INTC_IPR_INTLEV_MASK |
| Flashc header file:<br><br>AVR32_FLASHC_FGPFR_x | <br><br>AVR32_FLASHC_FGPFRLO_x |

To migrate a project from UC3Ax512ES (Rev E) target to UC3Ax512 (Rev H or later), the definitions described above must be updated.

## 4.2 AVR32 UC3 Software Framework

### 4.2.1 Overview

**Table 4-3. Supported Frameworks per Parts**

| UC3Ax Rev E | | UC3Ax Rev H or later | |
|---|---|---|---|
| **Toolchain 1.y.z** | **Toolchain 2.0 or later** | **Toolchain 1.y.z** | **Toolchain 2.0 or later** |
| Software framework 1.1.1 | Software framework 1.2.0ES | Not Available | Software framework 1.3.0 |

### 4.2.2 Version 1.1.1-AT32UC3A

The AVR32 software framework version 1.1.1 was dedicated to ES parts.

All drivers provided in this framework are implemented for the Rev E parts, and include workarounds for the Rev E errata.

This framework only supports the Toolchain version 1.y.z.

Since the Toolchain 2.0 or later introduced header files definition changes, the software framework 1.1.1 does not straight away compile with these toolchains.

### 4.2.3 Version 1.2.zES-AT32UC3A

The AVR32 software framework version 1.2.zES is dedicated to ES parts. All drivers provided in this framework are implemented for the Rev E parts, and include workarounds for the Rev E errata.

This framework only supports the Toolchain version 2.0 or later.

### 4.2.4 Version 1.3.z-AT32UC3A

The AVR32 software frameworks 1.3.z and later versions support UC3A Rev H and later revisions. It does not support the Rev E version (ES).

All drivers provided in this framework are implemented for the Rev H parts, and include workarounds for the Rev H errata.

This framework only supports the Toolchain version 2.0 or later.

## 4.3 USB DFU Bootloader

The pre-programmed Atmel USB DFU Bootloader implementation is different for Rev E and Rev H, the interface remains the same.

The bootloader binary and its programming script can be found:

- On Rev E, in software framework 1.2.zES :
  \SERVICES\USB\CLASS\DFU\EXAMPLES\ISP\AT32UC3AES\Releases\

- On Rev H and later, in software framework 1.3.z: \SERVICES\USB\CLASS\DFU\EXAMPLES\ISP\AT32UC3A\Releases\

# 5 Suggested reading

## 5.1 Device datasheet

The device datasheet contains block diagrams of the peripherals and details about implementing firmware for the device. The datasheet is available on http://www.atmel.com/AVR32 in the *Datasheets* section.

## 5.2 AVR32 Software Framework

This framework provides software drivers and libraries to build any application for AVR32 UC3 devices. It is available on http://www.atmel.com/AVR32 in the *Tools & Software* section.

## 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

***Technical Support***
avr32@atmel.com

***Sales Contact***
www.atmel.com/contacts

***Literature Request***
www.atmel.com/literature