Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	AVR
Core Size	32-Bit Single-Core
Speed	48MHz
Connectivity	I²C, SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, I²S, POR, PWM, WDT
Number of I/O	35
Program Memory Size	64KB (64K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	16K x 8
Voltage - Supply (Vcc/Vdd)	1.65V ~ 3.6V
Data Converters	A/D 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	48-TQFP
Supplier Device Package	48-TQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/atmel/atuc64d4-aut

9.5.4 Priority Registers B For Slaves

Name: PRBS0...PRBS15

Access Type: Read/Write

Offset: -

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	M15PR		-	-	M14PR	

23	22	21	20	19	18	17	16
-	-	M13PR		-	-	M12PR	

15	14	13	12	11	10	9	8
-	-	M11PR		-	-	M10PR	

7	6	5	4	3	2	1	0
-	-	M9PR		-	-	M8PR	

- **MxPR: Master x Priority**

  Fixed priority of Master x for accessing the selected slave. The higher the number, the higher the priority.

13.6.7 Unlock Register

Name: UNLOCK

Access Type: Write-only

Offset: 0x0018

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	ADDR[9:8]	

7	6	5	4	3	2	1	0
ADDR[7:0]							

To unlock a write protected register, first write to the UNLOCK register with the address of the register to unlock in the ADDR field and 0xAA in the KEY field. Then, in the next PB access write to the register specified in the ADDR field.

- **KEY: Unlock Key**
  Write this bit field to 0xAA to enable unlock.
- **ADDR: Unlock Address**
  Write the address of the register to unlock to this field.

- **CEN: Clock Enable**
  0: Clock is stopped.
  1: Clock is running.

14. Asynchronous Timer (AST)

Rev: 3.1.0.1

14.1 Features

- **32-bit counter with 32-bit prescaler**
- **Clocked Source**
  - **System RC oscillator (RCSYS)**
  - **32KHz crystal oscillator (OSC32K)**
  - **PB clock**
  - **Generic clock (GCLK)**
- **Optional calendar mode supported**
- **Periodic interrupt(s) supported**
- **Alarm interrupt(s) supported**
  - **Optional clear on alarm**

14.2 Overview

The Asynchronous Timer (AST) enables periodic interrupts, as well as interrupts at a specified time in the future. The AST consists of a 32-bit prescaler which feeds a 32-bit up-counter. The prescaler can be clocked from five different clock sources, including the low-power 32KHz oscillator, which allows the AST to be used as a real-time timer with a maximum timeout of more than 100 years. Also, the PB clock or a generic clock can be used for high-speed operation, allowing the AST to be used as a general timer.

The AST can generate periodic interrupts from output from the prescaler, as well as alarm interrupts, which can trigger at any counter value. Additionally, the timer can trigger an overflow interrupt, and be reset on the occurrence of any alarm. This allows periodic interrupts at very long and accurate intervals.

The AST has been designed to meet the system tick and Real Time Clock requirements of most embedded operating systems.

19.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

19.5.1 I/O Lines

The USBC pins may be multiplexed with the I/O Controller lines. The user must first configure the I/O Controller to assign the desired USBC pins to their peripheral functions.

19.5.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the USBC, the USBC will stop functioning and resume operation after the system wakes up from sleep mode.

19.5.3 Clocks

The USBC has two bus clocks connected: One High Speed Bus clock (CLK_USBC_HSB) and one Peripheral Bus clock (CLK_USBC_PB). These clocks are generated by the Power Manager. Both clocks are enabled at reset, and can be disabled by the Power Manager. It is recommended to disable the USBC before disabling the clocks, to avoid freezing the USBC in an undefined state.

The 48 MHz USB clock is generated by a dedicated generic clock from the SCIF module. Before using the USB, the user must ensure that the USB generic clock (GCLK_USBC) is enabled at 48 MHz in the SCIF module.

19.5.4 Interrupts

The USBC interrupt request line is connected to the interrupt controller. Using the USBC interrupt requires the interrupt controller to be programmed first.

The USBC asynchronous interrupts can wake the CPU from any sleep mode:

- The VBUS Transition Interrupt (VBUSTI)
- The Wakeup Interrupt (WAKEUP)

19.7.1 USB General Registers

19.7.1.1 General Control Register

Name: USBCON

Access Type: Read/Write

Offset: 0x0800

Reset Value: 0x00004000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-		-	-	-	
15	14	13	12	11	10	9	8
USBE	FRZCLK	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	-	VBUSTE	-

- **USBE: USBC Enable**
  Writing a zero to this bit will disable the USBC, USB transceiver, and USB clock inputs. This will over-ride FRZCLK settings but not affect the value. Unless explicitly stated, all registers will become reset and read-only.
  Writing a one to this bit will enable the USBC.
  0: The USBC is disabled.
  1: The USBC is enabled.
  This bit can be written to even if FRZCLK is one.
- **FRZCLK: Freeze USB Clock**
  Writing a zero to this bit will enable USB clock inputs.
  Writing a one to this bit will disable USB clock inputs. The resume detection will remain active. Unless explicitly stated, all registers will become read-only.
  0: The clock inputs are enabled.
  1: The clock inputs are disabled.
  This bit can be written to even if USBE is zero.
- **VBUSTE: VBUS Transition Interrupt Enable**
  0: The VBUS Transition Interrupt (VBUSTI) is disabled.
  1: The VBUS Transition Interrupt (VBUSTI) is enabled.

19.8 Module Configuration

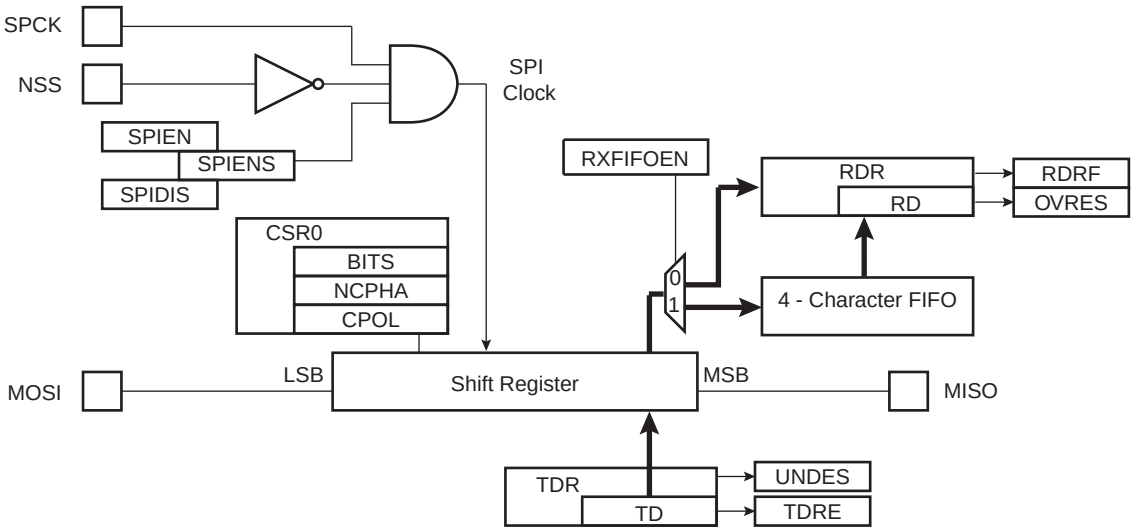
Table 19-6. USBC Clocks

Clock Name	Description
CLK_USBC	Clock for the USBC bus interface
GCLK_USBC	48Mhz USB clock. This clock frequency must be configured to 48MHz. The generic clock used for the USBC is GCLK3

Table 19-7. Register Reset Values

Register	Reset Value
UVERS	0x00000200
UFEATURES	0x00000007
UADDRSIZE	0x00001000
UNAME1	0x48555342
UNAME2	0x00000000

Figure 21-9. Slave Mode Functional Block Diagram



- **BITS: Bits Per Transfer**
  The BITS field determines the number of data bits transferred. Reserved values should not be used.

BITS	Bits Per Transfer
0000	8
0001	9
0010	10
0011	11
0100	12
0101	13
0110	14
0111	15
1000	16
1001	4
1010	5
1011	6
1100	7
1101	Reserved
1110	Reserved
1111	Reserved

- **CSAAT: Chip Select Active After Transfer**
  1: The Peripheral Chip Select does not rise after the last transfer is achieved. It remains active until a new transfer is requested on a different chip select.
  0: The Peripheral Chip Select Line rises as soon as the last transfer is achieved.
- **CSNAAT: Chip Select Not Active After Transfer (Ignored if CSAAT = 1)**
  0: The Peripheral Chip Select does not rise between two transfers if the TDR is reloaded before the end of the first transfer and if the two transfers occur on the same Chip Select.
  1: The Peripheral Chip Select rises systematically between each transfer performed on the same slave for a minimal duration of:

  $\frac{DLYBCS}{CLKSPI}$ (if DLYBCT field is different from 0)

  $\frac{DLYBCS + 1}{CLK\overline{SPI}}$ (if DLYBCT field equals 0)
- **NCPHA: Clock Phase**
  1: Data is captured after the leading (inactive-to-active) edge of SPCK and changed on the trailing (active-to-inactive) edge of SPCK.
  0: Data is changed on the leading (inactive-to-active) edge of SPCK and captured after the trailing (active-to-inactive) edge of SPCK.
  NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. NCPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.
- **CPOL: Clock Polarity**
  1: The inactive state value of SPCK is logic level one.
  0: The inactive state value of SPCK is logic level zero.

22.8.7 Interrupt Disable Register

Name: IDR

Access Type: Write-only

Offset: 0x18

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	TXUR	TXRDY	-	-	RXOR	RXRDY	-

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

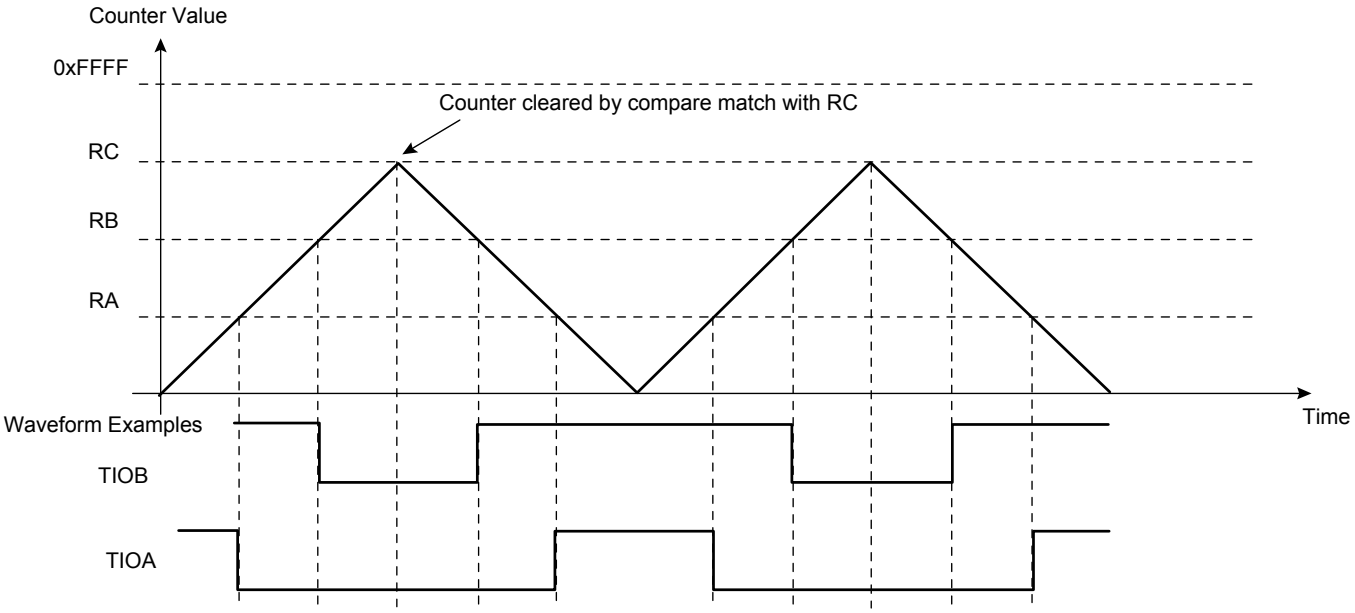
26.6.3.5 WAVSEL = 3

When CMRn.WAVSEL is three, the value of CVn is incremented from zero to RC. Once RC is reached, the value of CVn is decremented to zero, then re-incremented to RC and so on. See Figure 26-12 on page 542.

A trigger such as an external event or a software trigger can modify CVn at any time. If a trigger occurs while CVn is incrementing, CVn then decrements. If a trigger is received while CVn is decrementing, CVn then increments. See Figure 26-13 on page 543.

RC Compare can stop the counter clock (CMRn.CPCSTOP = 1) and/or disable the counter clock (CMRn.CPCDIS = 1).

Figure 26-12. WAVSEL = 3 Without Trigger



- RB Compare Effect on TIOB (CMRn.BCPB)
- RC Compare Effect on TIOA (CMRn.ACPC)
- RA Compare Effect on TIOA (CMRn.ACPA)

26.7.14 Features Register

Name: FEATURES

Access Type: Read-only

Offset: 0xF8

Reset Value: -

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	

15	14	13	12	11	10	9	8
-	-	-	-	-	-	BRPBHSB	UPDNIMPL

7	6	5	4	3	2	1	0
CTRSIZE							

- **BRPBHSB: Bridge type is PB to HSB**
  1: Bridge type is PB to HSB.
  0: Bridge type is not PB to HSB.
- **UPDNIMPL: Up/down is implemented**
  1: Up/down counter capability is implemented.
  0: Up/down counter capability is not implemented.
- **CTRSIZE: Counter size**
  This field indicates the size of the counter in bits.

27.7.14 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x4C

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
DMATSC	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	ACQDONE	ACREADY
7	6	5	4	3	2	1	0
-	-	-	-	-	ATSC	ATCAL	-

0: The corresponding interrupt is disabled.
1: The corresponding interrupt is enabled.
A bit in this register is cleared when the corresponding bit in IDR is written to one.
A bit in this register is set when the corresponding bit in IER is written to one.

28.7.15 Interrupt disable register

Name : IDR

Access Type : Write-Only

Offset : 0x38

Reset Value : 0x00000000

31	30	29	28	27	26	25	24
	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
TTO	SUTD	SMTRG	WM	LOVR	SEOC	SEOS	SSOS

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in IMR.

29.7 User Interface

Table 29-3. GLOC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00+n*0x08	Control Register n	CRn	Read/Write	0x00000000
0x04+n*0x08	Truth Table Register n	TRUTHn	Read/Write	0x00000000
0x38	Parameter Register	PARAMETER	Read-only	- (1)
0x3C	Version Register	VERSION	Read-only	- (1)

Note: 1. The reset values are device specific. Please refer to the Module Configuration section at the end of this chapter.

31.3 On-Chip Debug

Rev: 2.1.2.0

31.3.1 Features

- **Debug interface in compliance with IEEE-ISTO 5001-2003 (Nexus 2.0) Class 2+**
- **JTAG or aWire access to all on-chip debug functions**
- **Advanced Program, Data, Ownership, and Watchpoint trace supported**
- **NanoTrace aWire- or JTAG-based trace access**
- **Auxiliary port for high-speed trace information**
- **Hardware support for 6 Program and 2 Data breakpoints**
- **Unlimited number of software breakpoints supported**
- **Automatic CRC check of memory regions**

31.3.2 Overview

Debugging on the UC3D is facilitated by a powerful On-Chip Debug (OCD) system. The user accesses this through an external debug tool which connects to the JTAG or aWire port and the Auxiliary (AUX) port if implemented. The AUX port is primarily used for trace functions, and an aWire- or JTAG-based debugger is sufficient for basic debugging.

The debug system is based on the Nexus 2.0 standard, class 2+, which includes:

- Basic run-time control
- Program breakpoints
- Data breakpoints
- Program trace
- Ownership trace
- Data trace

In addition to the mandatory Nexus debug features, the UC3D implements several useful OCD features, such as:

- Debug Communication Channel between CPU and debugger
- Run-time PC monitoring
- CRC checking
- NanoTrace
- Software Quality Assurance (SQA) support

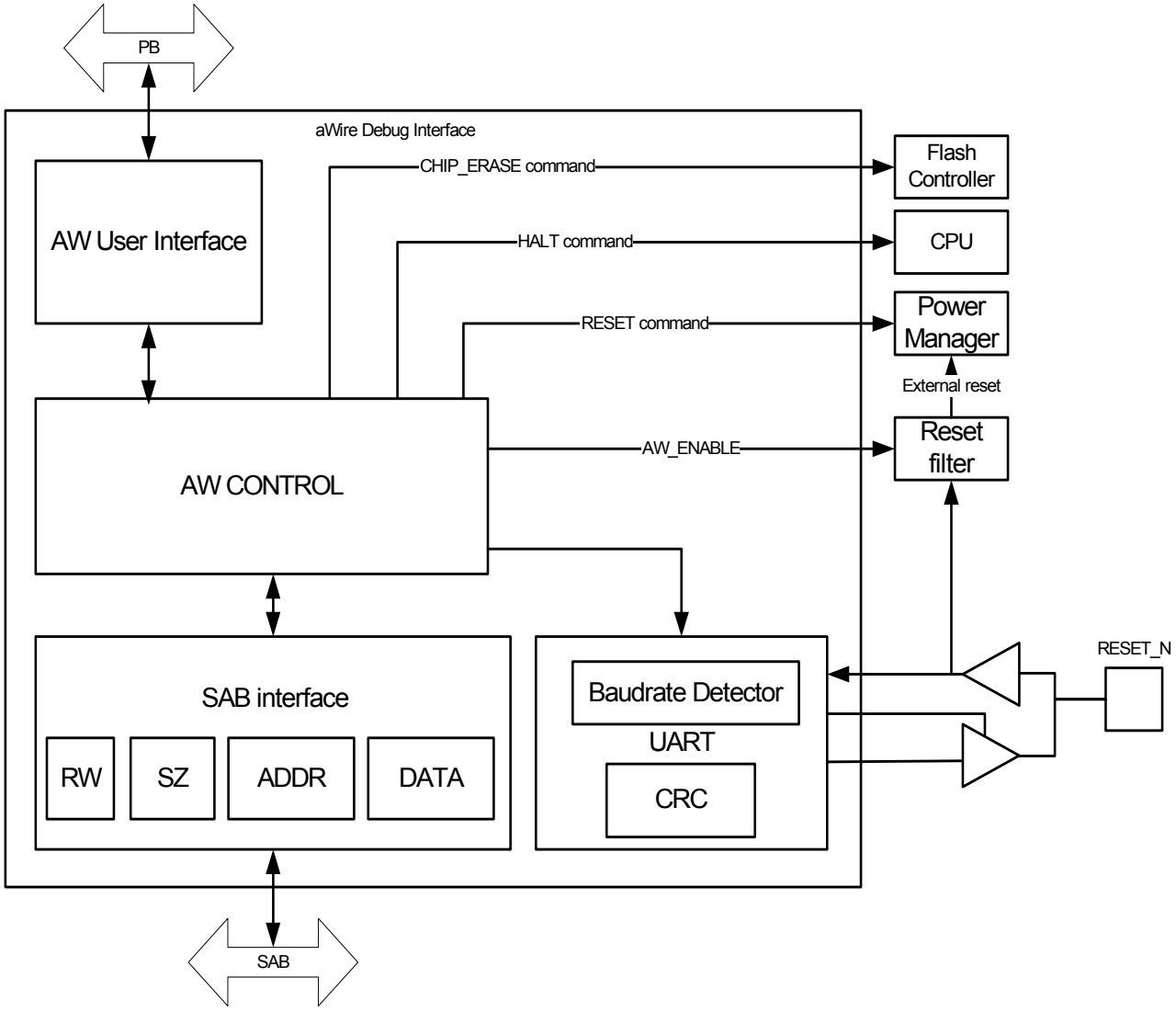
The OCD features are controlled by OCD registers, which can be accessed by the debugger, for instance when the NEXUS_ACCESS JTAG instruction is loaded. The CPU can also access OCD registers directly using mtdr/mfdr instructions in any privileged mode. The OCD registers are implemented based on the recommendations in the Nexus 2.0 standard, and are detailed in the AVR32UC Technical Reference Manual.

31.3.3 I/O Lines Description

The OCD AUX trace port contains a number of pins, as shown in Table 31-5 on page 668. These are multiplexed with I/O Controller lines, and must explicitly be enabled by writing OCD registers before the debug session starts. The AUX port is mapped to two different locations,

31.4.3 Block Diagram

Figure 31-5. aWire Debug Interface Block Diagram



31.4.4 I/O Lines Description

Table 31-6. I/O Lines Description

Name	Description	Type
DATA	aWire data multiplexed with the RESET_N pin.	Input/Output
DATAOUT	aWire data output in 2-pin mode.	Output

31.4.5 Product Dependencies

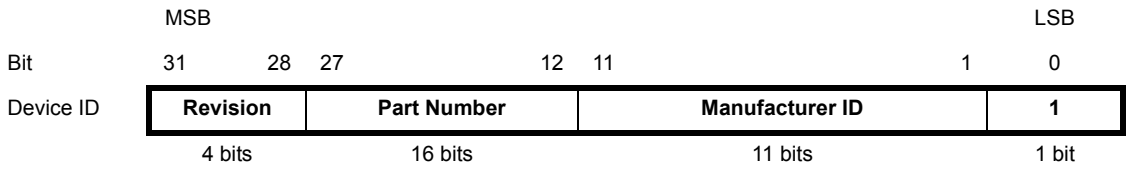
In order to use this module, other parts of the system must be configured correctly, as described below.

31.6.4 JTAG Data Registers

The following device specific registers can be selected as JTAG scan chain depending on the instruction loaded in the JTAG Instruction Register. Additional registers exist, but are implicitly described in the functional description of the relevant instructions.

31.6.4.1 Device Identification Register

The Device Identification Register contains a unique identifier for each product. The register is selected by the IDCODE instruction, which is the default instruction after a JTAG reset.



Revision This is a 4 bit number identifying the revision of the component. Rev A = 0x0, B = 0x1, etc.

Part Number The part number is a 16 bit code identifying the component.

Manufacturer ID The Manufacturer ID is a 11 bit code identifying the manufacturer. The JTAG manufacturer ID for ATMEL is 0x01F.

Device specific ID codes

The different device configurations have different JTAG ID codes, as shown in Table 31-58. Note that if the flash controller is statically reset, the ID code will be undefined.

Table 31-58. Device and JTAG ID

Device Name	JTAG ID Code (R is the revision number)
ATUC256D3	0xr212303F
ATUC128D3	0xr212003F
ATUC64D3	0xr212103F
ATUC256D4	0xr213303F
ATUC128D4	0xr213003F
ATUC64D4	0xr213103F

31.6.4.2 Reset Register

The reset register is selected by the AVR_RESET instruction and contains one bit for each reset domain in the device. Setting each bit to one will keep that domain reset until the bit is cleared.

Bit	Reset Domain	Description
0	System	Resets the whole chip, except the JTAG itself.

31.6.4.3 Boundary-Scan Chain

The Boundary-Scan Chain has the capability of driving and observing the logic levels on the digital I/O pins, as well as driving and observing the logic levels between the digital I/O pins and the

35. Errata

35.1 Rev. C

35.1.1 SPI

1. **SPI disable does not work in SLAVE mode**
   SPI disable does not work in SLAVE mode.
   Fix/Workaround
   Read the last received data, then perform a software reset by writing a one to the Software Reset bit in the Control Register (CR.SWRST).

2. **PCS field in receive data register is inaccurate**
   The PCS field in the SPI_RDR register does not accurately indicate from which slave the received data is read.
   Fix/Workaround
   None.

3. **SPI data transfer hangs with CSR0.CSAAT==1 and MR.MODFDIS==0**
   When CSR0.CSAAT==1 and mode fault detection is enabled (MR.MODFDIS==0), the SPI module will not start a data transfer.
   Fix/Workaround
   Disable mode fault detection by writing a one to MR.MODFDIS.

4. **Disabling SPI has no effect on the SR.TDRE bit**
   Disabling SPI has no effect on the SR.TDRE bit whereas the write data command is filtered when SPI is disabled. Writing to TDR when SPI is disabled will not clear SR.TDRE. If SPI is disabled during a PDCA transfer, the PDCA will continue to write data to TDR until its buffer is empty, and this data will be lost.
   Fix/Workaround
   Disable the PDCA, add two NOPs, and disable the SPI. To continue the transfer, enable the SPI and PDCA.

5. **SPI bad serial clock generation on 2nd chip_select when SCBR=1, CPOL=1, and NCPHA=0**
   When multiple chip selects (CS) are in use, if one of the baudrates equal 1 while one (CSRn.SCBR=1) of the others do not equal 1, and CSRn.CPOL=1 and CSRn.NCPHA=0, then an additional pulse will be generated on SCK.
   Fix/Workaround
   When multiple CS are in use, if one of the baudrates equals 1, the others must also equal 1 if CSRn.CPOL=1 and CSRn.NCPHA=0.

6. **Timer Counter**

7. **Channel chaining skips first pulse for upper channel**
   When chaining two channels using the Block Mode Register, the first pulse of the clock between the channels is skipped.
   Fix/Workaround
   Configure the lower channel with RA = 0x1 and RC = 0x2 to produce a dummy clock cycle for the upper channel. After the dummy cycle has been generated, indicated by the SR.CPCS bit, reconfigure the RA and RC registers for the lower channel with the real values.