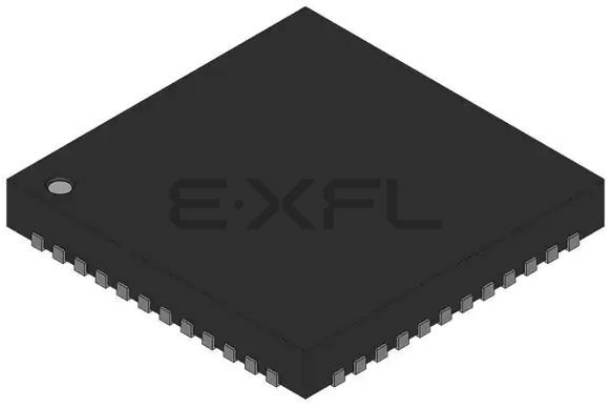


**Welcome to E-XFL.COM**

## 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 | 50MHz |
| Connectivity | I²C, SPI, UART/USART, USB |
| Peripherals | Brown-out Detect/Reset, DMA, POR, PWM, WDT |
| Number of I/O | 36 |
| Program Memory Size | 64KB (64K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 16K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.62V ~ 3.6V |
| Data Converters | A/D 8x12b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 48-VFQFN Exposed Pad |
| Supplier Device Package | 48-QFN (7x7) |
| Purchase URL | https://www.e-xfl.com/product-detail/atmel/atuc64l4u-zaur |

### 7.7.8 Memory Address Reload Register

**Name:** MARR

**Access Type:** Read/Write

**Offset:** 0x00C + n*0x040

**Reset Value:** 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MARV[31:24] | | | | | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| MARV[23:16] | | | | | | | |

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

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MARV[7:0] | | | | | | | |

- **MARV: Memory Address Reload Value**
  Reload Value for the MAR register. This value will be loaded into MAR when TCR reaches zero if the TCRR register has a non-zero value.

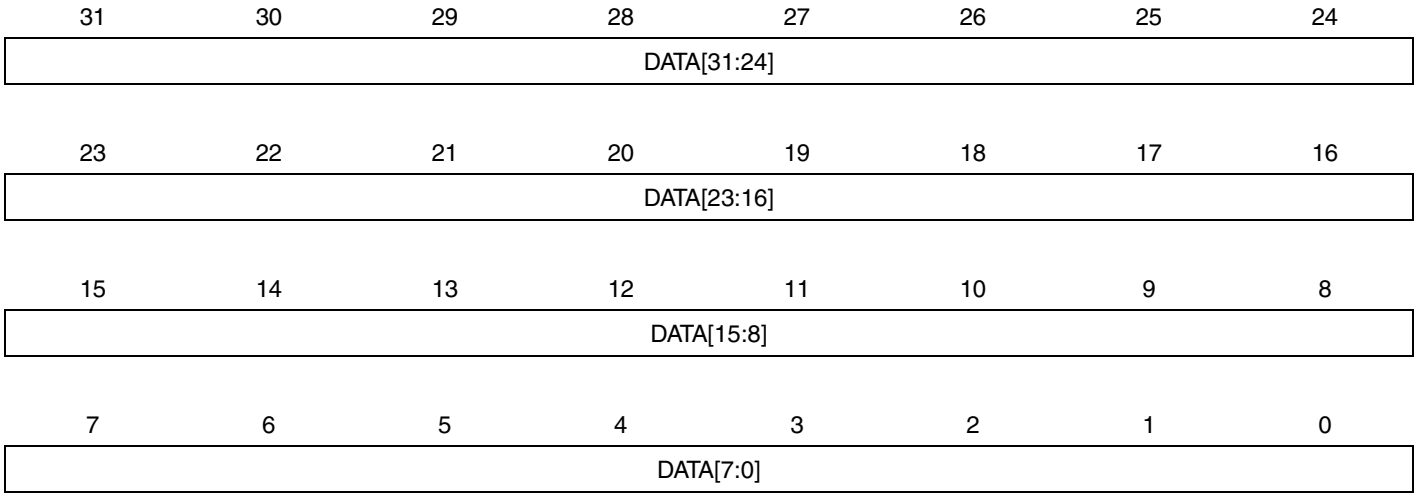
### 7.7.24 Performance Channel 1 Read Data Cycles

**Name:** PRDATA1

**Access Type:** Read-only

**Offset:** 0x81C

**Reset Value:** 0x00000000




- **DATA: Data Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock

### 8.7.2.12 *Endpoint n Status Clear Register*

**Register Name:** UESTAnCLR, n in [0..6]

**Access Type:** Write-Only

**Offset:** 0x0160 + (n * 0x04)

**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 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | RAMACERIC | - | - | - |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | STALLEDIC/ CRCERRIC | - | NAKINIC | NAKOUTIC | RXSTPIC/ ERRORFIC | RXOUTIC | TXINIC |

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 UESTA.
These bits always read as zero.

# 9. Flash Controller (FLASHCDW)

Rev: 1.2.0.0

## 9.1 Features

- **Controls on-chip flash memory**
- **Supports 0 and 1 wait state bus access**
- **Buffers reducing penalty of wait state in sequential code or loops**
- **Allows interleaved burst reads for systems with one wait state, outputting one 32-bit word per clock cycle for sequential reads**
- **Secure State for supporting FlashVault technology**
- **32-bit HSB interface for reads from flash and writes to page buffer**
- **32-bit PB interface for issuing commands to and configuration of the controller**
- **Flash memory is divided into 16 regions can be individually protected or unprotected**
- **Additional protection of the Boot Loader pages**
- **Supports reads and writes of general-purpose Non Volatile Memory (NVM) bits**
- **Supports reads and writes of additional NVM pages**
- **Supports device protection through a security bit**
- **Dedicated command for chip-erase, first erasing all on-chip volatile memories before erasing flash and clearing security bit**

## 9.2 Overview

The Flash Controller (FLASHCDW) interfaces the on-chip flash memory with the 32-bit internal HSB bus. The controller manages the reading, writing, erasing, locking, and unlocking sequences.

## 9.3 Product Dependencies

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

### 9.3.1 Power Management

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

### 9.3.2 Clocks

The FLASHCDW has two bus clocks connected: One High Speed Bus clock (CLK_FLASHCDW_HSB) and one Peripheral Bus clock (CLK_FLASHCDW_PB). These clocks are generated by the Power Manager. Both clocks are enabled at reset, and can be disabled by writing to the Power Manager. The user has to ensure that CLK_FLASHCDW_HSB is not turned off before reading the flash or writing the pagebuffer and that CLK_FLASHCDW_PB is not turned off before accessing the FLASHCDW configuration and control registers. Failing to do so may deadlock the bus.

### 9.3.3 Interrupts

The FLASHCDW interrupt request lines are connected to the interrupt controller. Using the FLASHCDW interrupts requires the interrupt controller to be programmed first.

The page buffer is not automatically reset after a page write. The programmer should do this manually by issuing the Clear Page Buffer flash command. This can be done after a page write, or before the page buffer is loaded with data to be stored to the flash page.

## 9.5 Flash Commands

The FLASHCDW offers a command set to manage programming of the flash memory, locking and unlocking of regions, and full flash erasing. See Section 9.8.2 for a complete list of commands.

To run a command, the CMD field in the Flash Command Register (FCMD) has to be written with the command number. As soon as the FCMD register is written, the FRDY bit in the Flash Status Register (FSR) is automatically cleared. Once the current command is complete, the FSR.FRDY bit is automatically set. If an interrupt has been enabled by writing a one to FCR.FRDY, the interrupt request line of the Flash Controller is activated. All flash commands except for Quick Page Read (QPR) and Quick User Page Read (QPRUP) will generate an interrupt request upon completion if FCR.FRDY is one.

Any HSB bus transfers attempting to read flash memory when the FLASHCDW is busy executing a flash command will be stalled, and allowed to continue when the flash command is complete.

After a command has been written to FCMD, the programming algorithm should wait until the command has been executed before attempting to read instructions or data from the flash or writing to the page buffer, as the flash will be busy. The waiting can be performed either by polling the Flash Status Register (FSR) or by waiting for the flash ready interrupt. The command written to FCMD is initiated on the first clock cycle where the HSB bus interface in FLASHCDW is IDLE. The user must make sure that the access pattern to the FLASHCDW HSB interface contains an IDLE cycle so that the command is allowed to start. Make sure that no bus masters such as DMA controllers are performing endless burst transfers from the flash. Also, make sure that the CPU does not perform endless burst transfers from flash. This is done by letting the CPU enter sleep mode after writing to FCMD, or by polling FSR for command completion. This polling will result in an access pattern with IDLE HSB cycles.

All the commands are protected by the same keyword, which has to be written in the eight highest bits of the FCMD register. Writing FCMD with data that does not contain the correct key and/or with an invalid command has no effect on the flash memory; however, the PROGE bit is set in the Flash Status Register (FSR). This bit is automatically cleared by a read access to the FSR register.

Writing a command to FCMD while another command is being executed has no effect on the flash memory; however, the PROGE bit is set in the Flash Status Register (FSR). This bit is automatically cleared by a read access to the FSR register.

If the current command writes or erases a page in a locked region, or a page protected by the BOOTPROT fuses, the command has no effect on the flash memory; however, the LOCKE bit is set in the FSR register. This bit is automatically cleared by a read access to the FSR register.

### 9.5.1 Write/Erase Page Operation

Flash technology requires that an erase must be done before programming. The entire flash can be erased by an Erase All command. Alternatively, pages can be individually erased by the Erase Page command.

The User page can be written and erased using the mechanisms described in this chapter.

- **FSZ: Flash Size**
  The size of the flash. Not all device families will provide all flash sizes indicated in the table.

**Table 9-10.** Flash Size

| FSZ | Flash Size | FSZ | Flash Size |
|---|---|---|---|
| 0 | 4 Kbyte | 8 | 192 Kbyte |
| 1 | 8 Kbyte | 9 | 256 Kbyte |
| 2 | 16 Kbyte | 10 | 384 Kbyte |
| 3 | 32 Kbyte | 11 | 512 Kbyte |
| 4 | 48 Kbyte | 12 | 768 Kbyte |
| 5 | 64 Kbyte | 13 | 1024 Kbyte |
| 6 | 96 Kbyte | 14 | 2048 Kbyte |
| 7 | 128 Kbyte | 15 | Reserved |

### 14.6.10 DFLLn Configuration Register

**Name:** DFLLnCONF

**Access Type:** Read/Write

**Reset Value:** 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| COARSE[7:0] | | | | | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | FINE[8] |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| FINE[7:0] | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | QLEN | CCEN | - | LLAW | DITHER | MODE | EN |

- **COARSE: Coarse Calibration Value**
  Set the value of the coarse calibration register. If in closed loop mode, this field is Read-only.
- **FINE: FINE Calibration Value**
  Set the value of the fine calibration register. If in closed loop mode, this field is Read-only.
- **QLEN: Quick Lock Enable**
  0: Quick Lock is disabled.
  1: Quick Lock is enabled.
- **CCEN: Chill Cycle Enable**
  0: Chill Cycle is disabled.
  1: Chill Cycle is enabled.
- **LLAW: Lose Lock After Wake**
  0: Locks will not be lost after waking up from sleep modes.
  1: Locks will be lost after waking up from sleep modes where the DFLL clock has been stopped.
- **DITHER: Enable Dithering**
  0: The fine LSB input to the VCO is constant.
  1: The fine LSB input to the VCO is dithered to achieve sub-LSB approximation to the correct multiplication ratio.
- **MODE: Mode Selection**
  0: The DFLL is in open loop operation.
  1: The DFLL is in closed loop operation.
- **EN: Enable**
  0: The DFLL is disabled.
  1: The DFLL is enabled.

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

### 14.6.15 DFLLn Synchronization Register

**Name:** DFLLnSYNC

**Access Type:** Write-only

**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 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | SYNC |

- **SYNC: Synchronization**
  To be able to read the current value of DFLLnCONF or DFLLnRATIO in closed-loop mode, this bit should be written to one. The updated value is available in DFLLnCONF and DFLLnRATIO when PCLKSR.DFLLnRDY is set.

### 14.6.44 Generic Clock Version Register

**Name:** GCLKVERSION

**Access Type:** Read-only

**Offset:** 0x03F8

**Reset Value:** -

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

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

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

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| VERSION[7:0] | | | | | | | |

- **VARIANT: Variant number**
  Reserved. No functionality associated.
- **VERSION: Version number**
  Version number of the module. No functionality associated.

**Figure 17-2.** Timing Diagram, Synchronous Interrupts, High Level or Rising Edge

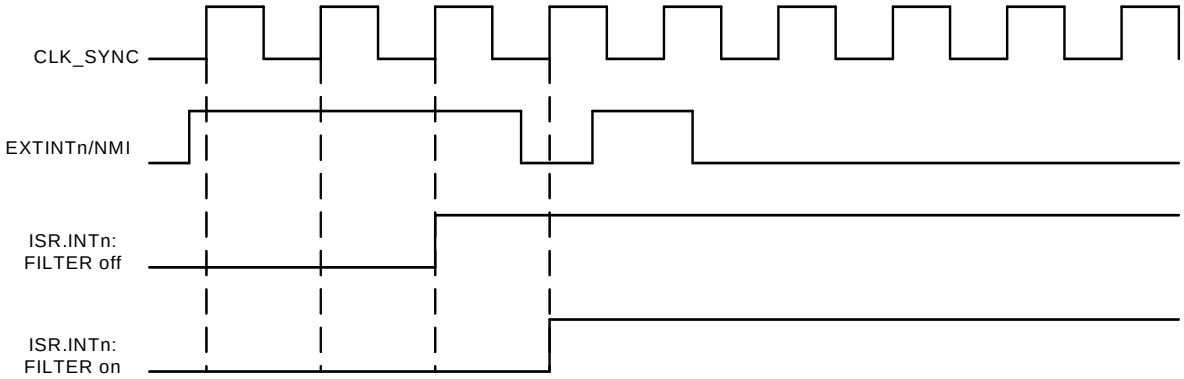
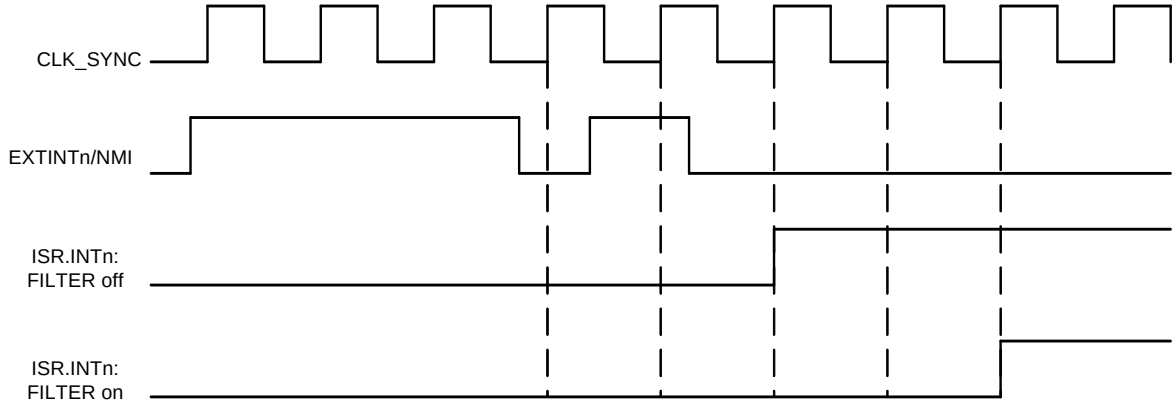



**Figure 17-3.** Timing Diagram, Synchronous Interrupts, Low Level or Falling Edge




### 17.6.3 Non-Maskable Interrupt

The NMI supports the same features as the external interrupts, and is accessed through the same registers. The description in Section 17.6.1 should be followed, accessing the NMI bit instead of the INTn bits.

The NMI is non-maskable within the CPU in the sense that it can interrupt any other execution mode. Still, as for the other external interrupts, the actual NMI input can be enabled and disabled by accessing the registers in the EIC.

### 17.6.4 Asynchronous Interrupts

Each external interrupt can be made asynchronous by writing a one to INTn in the ASYNC register. This will route the interrupt signal through the asynchronous path of the module. All edge interrupts will be interpreted as level interrupts and the filter is disabled. If an interrupt is configured as edge triggered interrupt in asynchronous mode, a zero in EDGE.INTn will be interpreted as low level, and a one in EDGE.INTn will be interpreted as high level.

EIC_WAKE will be set immediately after the source triggers the interrupt, while the corresponding bit in ISR and the interrupt to the interrupt controller will be set on the next rising edge of CLK_SYNC. Please refer to Figure 17-4 on page 369 for details.

## 17.7 User Interface

**Table 17-2.** EIC Register Memory Map

| Offset | Register | Register Name | Access | Reset |
|---|---|---|---|---|
| 0x000 | Interrupt Enable Register | IER | Write-only | 0x00000000 |
| 0x004 | Interrupt Disable Register | IDR | Write-only | 0x00000000 |
| 0x008 | Interrupt Mask Register | IMR | Read-only | 0x00000000 |
| 0x00C | Interrupt Status Register | ISR | Read-only | 0x00000000 |
| 0x010 | Interrupt Clear Register | ICR | Write-only | 0x00000000 |
| 0x014 | Mode Register | MODE | Read/Write | 0x00000000 |
| 0x018 | Edge Register | EDGE | Read/Write | 0x00000000 |
| 0x01C | Level Register | LEVEL | Read/Write | 0x00000000 |
| 0x020 | Filter Register | FILTER | Read/Write | 0x00000000 |
| 0x024 | Test Register | TEST | Read/Write | 0x00000000 |
| 0x028 | Asynchronous Register | ASYNC | Read/Write | 0x00000000 |
| 0x030 | Enable Register | EN | Write-only | 0x00000000 |
| 0x034 | Disable Register | DIS | Write-only | 0x00000000 |
| 0x038 | Control Register | CTRL | Read-only | 0x00000000 |
| 0x3FC | Version Register | VERSION | Read-only | - (1) |

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

### 17.7.5 Interrupt Clear Register

**Name:** ICR

**Access Type:** Write-only

**Offset:** 0x010

**Reset Value:** 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | INT30 | INT29 | INT28 | INT27 | INT26 | INT25 | INT24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| INT23 | INT22 | INT21 | INT20 | INT19 | INT18 | INT17 | INT16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| INT15 | INT14 | INT13 | INT12 | INT11 | INT10 | INT9 | INT8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INT7 | INT6 | INT5 | INT4 | INT3 | INT2 | INT1 | NMI |

- **INTn: External Interrupt n**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the corresponding bit in ISR.
  Please refer to the Module Configuration section for the number of external interrupts.
- **NMI: Non-Maskable Interrupt**
  Writing a zero to this bit has no effect.
  Writing a one to this bit will clear the corresponding bit in ISR.

customizable response data lengths, and requires minimal CPU resources. Writing 0xA (master) or 0xB (slave) to MR.MODE enables this mode.

#### 20.6.5.1 *Modes of operation*

Changing LIN mode after initial configuration has to be followed by a transceiver software reset in order to avoid unpredictable behavior.

#### 20.6.5.2 *Receiver and Transmitter Control*

See Section “20.6.2” on page 439.

#### 20.6.5.3 *Baud Rate Configuration*

The LIN nodes baud rate is configured in the Baud Rate Generator Register (BRGR), See Section “20.6.1.1” on page 437.

#### 20.6.5.4 *Character Transmission and Reception*

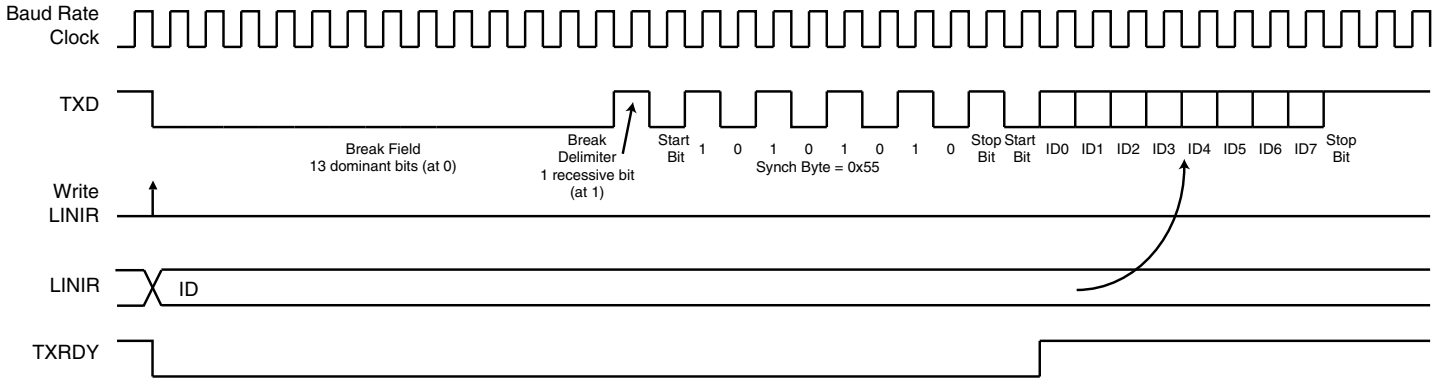
See ”Transmitter Operations” on page 440, and ”Receiver Operations” on page 442.

#### 20.6.5.5 *Header Transmission (Master Node Configuration)*

All LIN frames start with a header sent by the master. As soon as the identifier has been written to the Identifier Character field in the LIN Identifier Register (LINIR.IDCHR), TXRDY is cleared and the header is sent. The header consists of a Break, Sync, and Identifier field. TXRDY is set when the identifier has been transferred into the transmitters shift register.

The Break field consists of 13 dominant bits, the break, and one recessive bit, the break delimiter. The Sync field is the character 0x55. The Identifier field contains the Identifier as written to IDCHR. The identifier parity bits can be generated automatically (see Section 20.6.5.8).

**Figure 20-21.** Header Transmission




#### 20.6.5.6 *Header Reception (Slave Node Configuration)*

The USART stays idle until it detects a break field, consisting of at least 11 consecutive dominant bits (zeroes) on the bus. A received break will set the Lin Break bit (CSR.LINBK). The Sync field is used to synchronize the baud rate (see Section 20.6.5.7). IDCHR is updated and the LIN Identifier bit (CSR.LINID) is set when the Identifier has been received. The Identifier parity bits can be automatically checked (see Section 20.6.5.8). Writing a one to RSTSTA will clear LINBK and LINID.

- **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.

### 22.9.1 Control Register

**Name:** CR

**Access Type:** Write-only

**Offset:** 0x00

**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 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | STOP |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SWRST | - | SMDIS | SMEN | - | - | MDIS | MEN |

- **STOP: Stop the Current Transfer**
  Writing a one to this bit terminates the current transfer, sending a STOP condition after the shifter has become idle. If there are additional pending transfers, they will have to be explicitly restarted by software after the STOP condition has been successfully sent.
  Writing a zero to this bit has no effect.
- **SWRST: Software Reset**
  If the TWIM master interface is enabled, writing a one to this bit resets the TWIM. All transfers are halted immediately, possibly violating the bus semantics.
  If the TWIM master interface is not enabled, it must first be enabled before writing a one to this bit.
  Writing a zero to this bit has no effect.
- **SMDIS: SMBus Disable**
  Writing a one to this bit disables SMBus mode.
  Writing a zero to this bit has no effect.
- **SMEN: SMBus Enable**
  Writing a one to this bit enables SMBus mode.
  Writing a zero to this bit has no effect.
- **MDIS: Master Disable**
  Writing a one to this bit disables the master interface.
  Writing a zero to this bit has no effect.
- **MEN: Master Enable**
  Writing a one to this bit enables the master interface.
  Writing a zero to this bit has no effect.

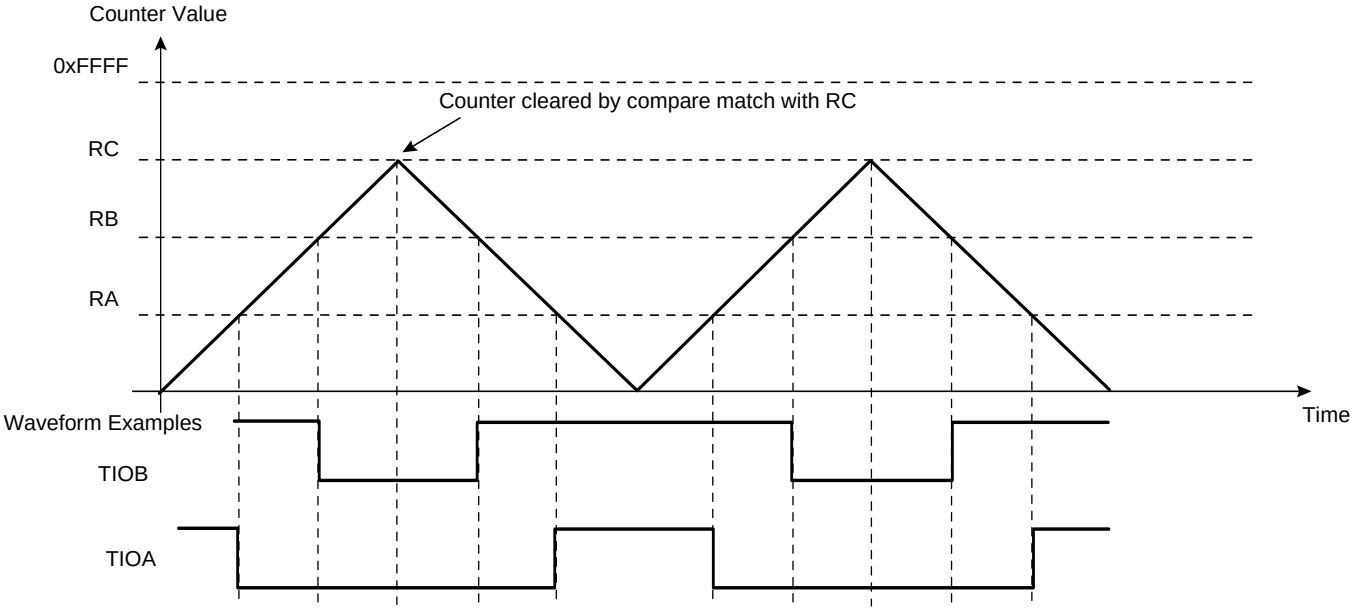
### 26.6.3.5 WAVSEL = 3

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 656.

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 657.

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



### 31.7.13 Status Register

**Name:** SR

**Access Type:** Read-only

**Offset:** 0x3C

**Reset Value:** 0x00000000

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

- **DMATSC: DMATouch Sensor State Change**
  0: No change in the DMATSS register.
  1: One or more bits have changed in the DMATSS register.
- **DMATSR: DMATouch State Read Register Ready**
  0: A new state word is not available in the DMATSR register.
  1: A new state word is available in the DMATSR register.
- **DMATSW: DMATouch State Write Register Request**
  0: The DMATouch algorithm is not requesting that a state word be written to the DMATSW register.
  1: The DMATouch algorithm is requesting that a state word be written to the DMATSW register.
- **ACQDONE: Acquisition Done**
  0: Acquisition is not done (still in progress).
  1: Acquisition is complete.
- **ACREADY: Acquired Count Data is Ready**
  0: Acquired count data is not available in the ACOUNT register.
  1: Acquired count data is available in the ACOUNT register.
- **MBLREQ: Matrix Burst Length Required**
  0: The QMatrix acquisition does not require any burst lengths.
  1: The QMatrix acquisition requires burst lengths for the current X line.
- **ATSTATE: Autonomous Touch Sensor State**
  0: The autonomous QTouch sensor is not active.
  1: The autonomous QTouch sensor is active.
- **ATSC: Autonomous Touch Sensor Status Interrupt**
  0: No status change in the autonomous QTouch sensor.
  1: Status change in the autonomous QTouch sensor.

**Figure 34-2.** JTAG-based Debugger

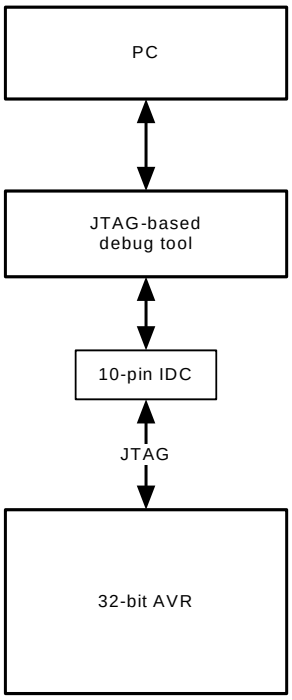
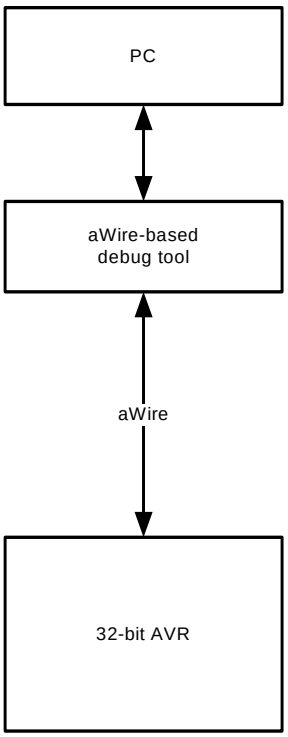



**Figure 34-3.** aWire-based Debugger




### *34.3.6.1 Debug Communication Channel*

The Debug Communication Channel (DCC) consists of a pair OCD registers with associated handshake logic, accessible to both CPU and debugger. The registers can be used to exchange data between the CPU and the debugmaster, both runtime as well as in debug mode.

1. The size of the data field: 7 (size and starting address + read length indicator) in the length field.
2. The size of the transfer: Words, halfwords, or bytes.
3. The starting address of the transfer.
4. The number of **bytes** to read (max 65532).

The 4 MSB of the 36 bit SAB address are submitted together with the size field (2 bits). The 4 remaining address bytes are submitted before the number of bytes to read. The size of the transfer is specified using the values from the following table:

**Table 34-43.** Size Field Decoding

| Size field | Description |
|---|---|
| 00 | Byte transfer |
| 01 | Halfword transfer |
| 10 | Word transfer |
| 11 | Reserved |

Below is an example read command:

1. 0x55 (sync)
2. 0x81 (command)
3. 0x00 (length MSB)
4. 0x07 (length LSB)
5. 0x25 (size and address MSB, the two MSB of this byte are unused and set to zero)
6. 0x00
7. 0x00
8. 0x00
9. 0x04 (address LSB)
10. 0x00
11. 0x04
12. 0xXX (CRC MSB)
13. 0xXX (CRC LSB)

The length field is set to 0x0007 because there are 7 bytes of additional data: 5 bytes of address and size and 2 bytes with the number of bytes to read. The address and size field indicates one word (four bytes) should be read from address 0x500000004.

**Table 34-44.** MEMORY_READ Details

| Command | Details |
|---|---|
| Command value | 0x81 |
| Additional data | Size, Address and Length |
| Possible responses | 0xC1: MEMDATA (Section 34.6.8.4)<br>0xC2: MEMORY_READWRITE_STATUS (Section 34.6.8.5)<br>0x41: NACK (Section 34.6.8.2) |

### 35.6.2 32 KHz Crystal Oscillator (OSC32K) Characteristics

Figure 35-3 and the equation above also applies to the 32 KHz oscillator connection. The user must choose a crystal oscillator where the crystal load capacitance $C_L$ is within the range given in the table. The exact value of $C_L$ can then be found in the crystal datasheet.

**Table 35-12.** 32 KHz Crystal Oscillator Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $f_{OUT}$ | Crystal oscillator frequency | | | 32 768 | | Hz |
| $t_{STARTUP}$ | Startup time | $R_S$ = 60 kOhm, $C_L$ = 9 pF | | 30 000[1] | | cycles |
| $C_L$ | Crystal load capacitance[2] | | 6 | | 12.5 | pF |
| $C_i$ | Internal equivalent load capacitance | | | 2 | | |
| $I_{OSC32}$ | Current consumption | | | 0.6 | | µA |
| $R_S$ | Equivalent series resistance[2] | 32 768 Hz | 35 | | 85 | kOhm |

Notes: 1. Nominal crystal cycles.
2. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

### 35.6.3 Phase Locked Loop (PLL) Characteristics

**Table 35-13.** Phase Locked Loop Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $f_{OUT}$ | Output frequency[1] | | 40 | | 240 | MHz |
| $f_{IN}$ | Input frequency[1] | | 4 | | 16 | |
| $I_{PLL}$ | Current consumption | | | 8 | | µA/MHz |
| $t_{STARTUP}$ | Startup time, from enabling the PLL until the PLL is locked | $f_{IN}$= 4 MHz | | 200 | | µs |
| | | $f_{IN}$= 16 MHz | | 155 | | |

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.