Atmel - ATUC64L3U-Z3UR Datasheet

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	51
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	64-VFQFN Exposed Pad
Supplier Device Package	64-QFN (9x9)
Purchase URL	https://www.e-xfl.com/product-detail/atmel/atuc64l3u-z3ur

6.1.4 Power-up Sequence

6.1.4.1 Maximum Rise Rate

To avoid risk of latch-up, the rise rate of the power supplies must not exceed the values described in Table 35-3 on page 898.

Recommended order for power supplies is also described in this chapter.

6.1.4.2 Minimum Rise Rate

The integrated Power-on Reset (POR33) circuitry monitoring the VDDIN powering supply requires a minimum rise rate for the VDDIN power supply.

See Table 35-3 on page 898 for the minimum rise rate value.

If the application can not ensure that the minimum rise rate condition for the VDDIN power supply is met, one of the following configurations can be used:

- A logic “0” value is applied during power-up on pin PA11 (WAKE_N) until VDDIN rises above 1.2V.
- A logic “0” value is applied during power-up on pin RESET_N until VDDIN rises above 1.2V.

6.2 Startup Considerations

This chapter summarizes the boot sequence of the ATUC64/128/256L3/4U. The behavior after power-up is controlled by the Power Manager. For specific details, refer to the Power Manager chapter.

6.2.1 Starting of Clocks

After power-up, the device will be held in a reset state by the Power-on Reset (POR18 and POR33) circuitry for a short time to allow the power to stabilize throughout the device. After reset, the device will use the System RC Oscillator (RCSYS) as clock source. Please refer to Table 35-17 on page 910 for the frequency for this oscillator.

On system start-up, all high-speed clocks are disabled. All clocks to all modules are running. No clocks have a divided frequency; all parts of the system receive a clock with the same frequency as the System RC Oscillator.

When powering up the device, there may be a delay before the voltage has stabilized, depending on the rise time of the supply used. The CPU can start executing code as soon as the supply is above the POR18 and POR33 thresholds, and before the supply is stable. Before switching to a high-speed clock source, the user should use the BOD to make sure the VDDCORE is above the minimum level (1.62V).

6.2.2 Fetching of Initial Instructions

After reset has been released, the AVR32 UC CPU starts fetching instructions from the reset address, which is 0x80000000. This address points to the first address in the internal Flash.

The code read from the internal flash is free to configure the clock system and clock sources. Please refer to the PM and SCIF chapters for more details.

7.5.10 Priority

If more than one PDCA channel is requesting transfer at a given time, the PDCA channels are prioritized by their channel number. Channels with lower numbers have priority over channels with higher numbers, giving channel zero the highest priority.

7.5.11 Error Handling

If the Memory Address Register (MAR) is set to point to an invalid location in memory, an error will occur when the PDCA tries to perform a transfer. When an error occurs, the Transfer Error bit in the Interrupt Status Register (ISR.TERR) will be set and the DMA channel that caused the error will be stopped. In order to restart the channel, the user must program the Memory Address Register to a valid address and then write a one to the Error Clear bit in the Control Register (CR.ECLR). If the Transfer Error interrupt is enabled, an interrupt request will be generated when a transfer error occurs.

7.5.12 Peripheral Event Trigger

Peripheral events can be used to trigger PDCA channel transfers. Peripheral Event synchronizations are enabled by writing a one to the Event Trigger bit in the Mode Register (MR.ETRIG). When set, all DMA requests will be blocked until a peripheral event is received. For each peripheral event received, only one data item is transferred. If no DMA requests are pending when a peripheral event is received, the PDCA will start a transfer as soon as a peripheral event is detected. If multiple events are received while the PDCA channel is busy transferring data, an overflow condition will be signaled in the Peripheral Event System. Refer to the Peripheral Event System chapter for more information.

7.6 Performance Monitors

Up to two performance monitors allow the user to measure the activity and stall cycles for PDCA transfers. To monitor a PDCA channel, the corresponding channel number must be written to one of the MON0/1CH fields in the Performance Control Register (PCONTROL) and a one must be written to the corresponding CH0/1EN bit in the same register.

Due to performance monitor hardware resource sharing, the two monitor channels should NOT be programmed to monitor the same PDCA channel. This may result in UNDEFINED performance monitor behavior.

7.6.1 Measuring mechanisms

Three different parameters can be measured by each channel:

- The number of data transfer cycles since last channel reset, both for read and write
- The number of stall cycles since last channel reset, both for read and write
- The maximum latency since last channel reset, both for read and write

These measurements can be extracted by software and used to generate indicators for bus latency, bus load, and maximum bus latency.

Each of the counters has a fixed width, and may therefore overflow. When an overflow is encountered in either the Performance Channel Data Read/Write Cycle registers (PRDATA0/1 and PWDATA0/1) or the Performance Channel Read/Write Stall Cycles registers (PRSTALL0/1 and PWSTALL0/1) of a channel, all registers in the channel are reset. This behavior is altered if the Channel Overflow Freeze bit is one in the Performance Control register (PCONTROL.CH0/1OVF). If this bit is one, the channel registers are frozen when either DATA or STALL reaches its maximum value. This simplifies one-shot readout of the counter values.

set, or if the total byte count is not an integral multiple of EPSIZE, whereby the last packet should be short.

To enable the multi packet mode, the user should configure the endpoint descriptor (EPn_PCKSIZE_BK0/1.BYTE_COUNT) to the total size of the multi packet, which should be larger than the endpoint size (EPSIZE).

Since the EPn_PCKSIZE_BK0/1.MULTI_PACKET_SIZE is incremented (by the transmitted packet size) after each successful transaction, it should be set to zero when setting up a new multi packet transfer.

The EPn_PCKSIZE_BK0/1.MULTI_PACKET_SIZE is cleared by hardware when all the bank contents have been sent. The bank is considered as ready and the TX_IN flag is set when:

- A short packet (smaller than EPSIZE) has been transmitted.
- A packet has been successfully transmitted, the updated MULTI_PACKET_SIZE equals the BYTE_COUNT, and the AUTO_ZLP field is not set.
- An extra zero length packet has been automatically sent for the last transfer of the current bank, if BYTE_COUNT is a multiple of EPSIZE and AUTO_ZLP is set.

8.6.2.15 Management of OUT endpoints

- *Overview*

The endpoint and its descriptor in RAM must be pre configured, see section "RAM management" on page 90 for more details.

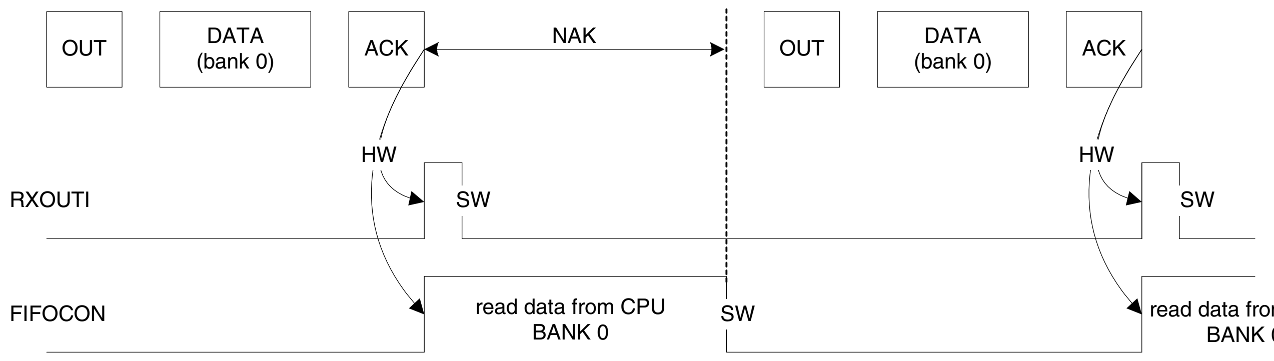
When the current bank is full, the RXOUTI and FIFO Control (UECONn.FIFOCON) bits will be set simultaneously. This triggers an EPnINT interrupt if the Received OUT Data Interrupt Enable (RXOUTE) bit in UECONn is one.

RXOUTI shall be cleared by software (by writing a one to the Received OUT Data Interrupt Clear (RXOUTIC) bit) to acknowledge the interrupt. This has no effect on the endpoint FIFO.

The user reads the OUT data from the RAM and clears the FIFOCON bit to free the bank. This will also cause a switch to the next bank if the OUT endpoint is composed of multiple banks.

RXOUTI should always be cleared before clearing FIFOCON to avoid missing an RXOUTI event.

Figure 8-11. Example of an OUT endpoint with one data bank



10.6.5 Status Register

Name:	SR
Access Type:	Read-only
Offset:	0x10
Reset Value:	0x00000400

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

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

15	14	13	12	11	10	9	8
-	-	-	-	-	IDLE	SEN	EN

7	6	5	4	3	2	1	0
RTRADR	MBERROR	URES	URKEY	URREAD	CAU	CAS	EXP

- **IDLE**

  This bit is cleared when a read or write operation to the SAU channel is started.

  This bit is set when the operation is completed and no SAU bus operations are pending.
- **SEN: SAU Setup Mode Enable**

  This bit is cleared when the SAU exits setup mode.

  This bit is set when the SAU enters setup mode.
- **EN: SAU Enabled**

  This bit is cleared when the SAU is disabled.

  This bit is set when the SAU is enabled.
- **RTRADR: RTR Address Error**

  This bit is cleared when the corresponding bit in ICR is written to one.

  This bit is set if, in the configuration phase, an RTR was written with an illegal address, i.e. the upper 16 bits in the address were different from 0xFFFC, 0xFFFD, 0xFFFE or 0xFFFF.
- **MBERROR: Master Interface Bus Error**

  This bit is cleared when the corresponding bit in ICR is written to one.

  This bit is set if a channel access generated a transfer on the master interface that received a bus error response from the addressed slave.
- **URES: Unlock Register Error Status**

  This bit is cleared when the corresponding bit in ICR is written to one.

  This bit is set if an attempt was made to unlock a channel by writing to the Unlock Register while one or more error bits were set in SR. The unlock operation was aborted.
- **URKEY: Unlock Register Key Error**

  This bit is cleared when the corresponding bit in ICR is written to one.

  This bit is set if the Unlock Register was attempted written with an invalid key.
- **URREAD: Unlock Register Read**

  This bit is cleared when the corresponding bit in ICR is written to one.

  This bit is set if the Unlock Register was read.

14.6.9 32KHz Oscillator Control Register

Name: OSCCTRL32

Access Type: Read/Write

Reset Value: 0x00000004

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

23	22	21	20	19	18	17	16
-	-	-	-	-	STARTUP[2:0]		

15	14	13	12	11	10	9	8
-	-	-	-	-	MODE[2:0]		

7	6	5	4	3	2	1	0
-	-	-	-	EN1K	EN32K	PINSEL	OSC32EN

Note: This register is only reset by Power-On Reset

- **RESERVED**
  This bit must always be written to zero.
- **STARTUP: Oscillator Start-up Time**
  Select start-up time for 32 KHz oscillator

Table 14-3. Start-up Time for 32 KHz Oscillator

STARTUP	Number of RCSYS Clock Cycle	Approximative Equivalent Time (RCOSC = 115 kHz)
0	0	0
1	128	1.1 ms
2	8192	72.3 ms
3	16384	143 ms
4	65536	570 ms
5	131072	1.1 s
6	262144	2.3 s
7	524288	4.6 s

14.6.43 32kHz RC Oscillator Version Register

Name:	RC32KIFAVERSION
Access Type:	Read-only
Offset:	0x03F4
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

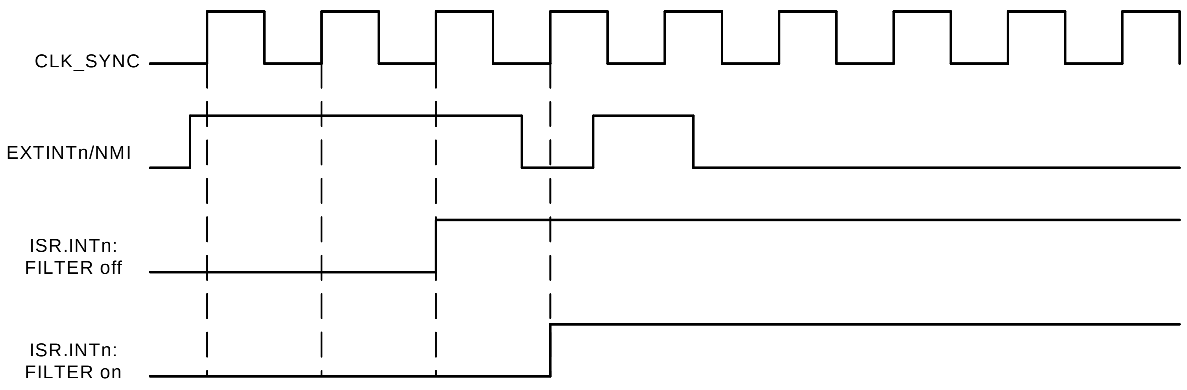
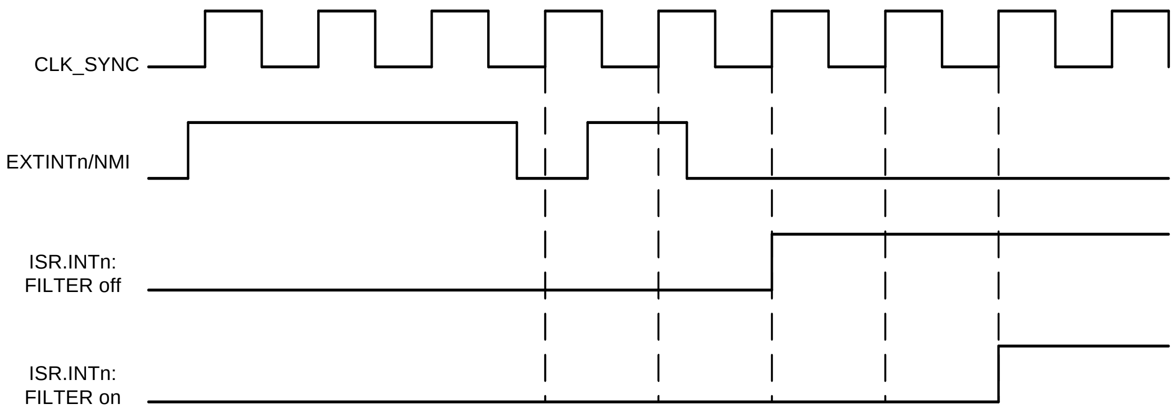


Figure 17-3. Timing Diagram, Synchronous Interrupts, Low Level or Falling Edge



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 controller will be set on the next rising edge of CLK_SYNC. Please refer to Figure 17-4 on page 369 for details.

17.7.7 Edge Register

Name:	EDGE
Access Type:	Read/Write
Offset:	0x018
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**

  0: The external interrupt triggers on falling edge.

  1: The external interrupt triggers on rising edge.

  Please refer to the Module Configuration section for the number of external interrupts.
- **NMI: Non-Maskable Interrupt**

  0: The Non-Maskable Interrupt triggers on falling edge.

  1: The Non-Maskable Interrupt triggers on rising edge.

18.6.7 Interrupt Mask Register

Name:	IMR
Access Type:	Read-only
Offset:	0x018
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
-	-	-	-	-	-	RCLKRDY	DONE

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.

19.7.22 Version Register

Name:	VERSION
Access Type:	Read-only
Offset:	0x1FC
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.

20.7 User Interface

Table 20-10. USART Register Memory Map

Offset	Register	Name	Access	Reset
0x0000	Control Register	CR	Write-only	0x00000000
0x0004	Mode Register	MR	Read-write	0x00000000
0x0008	Interrupt Enable Register	IER	Write-only	0x00000000
0x000C	Interrupt Disable Register	IDR	Write-only	0x00000000
0x0010	Interrupt Mask Register	IMR	Read-only	0x00000000
0x0014	Channel Status Register	CSR	Read-only	0x00000000
0x0018	Receiver Holding Register	RHR	Read-only	0x00000000
0x001C	Transmitter Holding Register	THR	Write-only	0x00000000
0x0020	Baud Rate Generator Register	BRGR	Read-write	0x00000000
0x0024	Receiver Time-out Register	RTOR	Read-write	0x00000000
0x0028	Transmitter Timeguard Register	TTGR	Read-write	0x00000000
0x0054	LIN Mode Register	LINMR	Read-write	0x00000000
0x0058	LIN Identifier Register	LINIR	Read-write	0x00000000
0x00E4	Write Protect Mode Register	WPMR	Read-write	0x00000000
0x00E8	Write Protect Status Register	WPSR	Read-only	0x00000000
0x00FC	Version Register	VERSION	Read-only	0x–(1)

Note: 1. Values in the Version Register vary with the version of the IP block implementation.

20.7.7 Receiver Holding Register

Name:	RHR
Access Type:	Read-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**
–	–	–	–	–	–	–	RXCHR[8]
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXCHR[7:0]							

- **RXCHR: Received Character**
  Last received character.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce the required clock/data relationship between master and slave devices.

The RA Loading Selection field in CMRn (CMRn.LDRA) defines the TIOA edge for the loading of the RA register, and the RB Loading Selection field in CMRn (CMRn.LDRB) defines the TIOA edge for the loading of the RB register.

RA is loaded only if it has not been loaded since the last trigger or if RB has been loaded since the last loading of RA.

RB is loaded only if RA has been loaded since the last trigger or the last loading of RB.

Loading RA or RB before the read of the last value loaded sets the Load Overrun Status bit in SRn (SRn.LOVRS). In this case, the old value is overwritten.

26.6.2.2 Trigger conditions

In addition to the SYNC signal, the software trigger and the RC compare trigger, an external trigger can be defined.

The TIOA or TIOB External Trigger Selection bit in CMRn (CMRn.ABETRG) selects TIOA or TIOB input signal as an external trigger. The External Trigger Edge Selection bit in CMRn (CMRn.ETREDG) defines the edge (rising, falling or both) detected to generate an external trigger. If CMRn.ETRGEDG is zero (none), the external trigger is disabled.

28.6.3 Basic operation

To convert audio data to a digital bitstream the user must first initialize the ABDACB as described in Section 28.6.2. When the ABDACB is initialized and enabled it will indicate that it is ready to receive new data by setting the Transmit Ready bit in the Status Register (SR.TXRDY). When the TXRDY bit is set in the Status Register the user has to write new samples to Sample Data Register 0 (SDR0) and Sample Data Register 1 (SDR1). If the Mono Mode (MONO) bit in the Control Register (CR) is set, or one of the compact stereo formats are used by configuring the Data Word Format (DATAFORMAT) in the Control Register, only SDR0 has to be written. Failing to write to the sample data registers will result in an underrun indicated by the Transmit Underrun (TXUR) bit in the Status Register (SR.TXUR). When new samples are written to the sample data registers the TXRDY bit will be cleared.

To increase performance of the system an interrupt handler or DMA transfer can be used to write new samples to the sample data registers. See Section 28.6.10 for details on DMA, and Section 28.6.11 for details on interrupt.

28.6.4 Data Format

The input data type is two's complement. The Audio Bitstream DAC can be configured to accept different audio formats. The format must be configured in the Data Word Format field in the Control Register. In regular operation data for the two channels are written to the sample data registers SDR0 and SDR1. If the data format field specifies a format using less than 32 bits, data must be written right-justified in SDR0 and SDR1. Sign extension into the unused bits is not necessary. Only the 16 most significant bits in the data will be used by the ABDACB. For data formats larger than 16 bits the least significant bits are ignored. For 8-bit data formats the 8 bits will be used as the most significant bits in the 16-bit samples, the additional bits will be zeros.

The ABDACB also supports compact data formats for 16- and 8-bit samples. For 16-bit samples the sample for channel 0 must be written to bits 15 through 0 and the sample for channel 1 must be written to bits 31 through 16 in SDR0. For 8-bit samples the sample for channel 0 must be written to bits 7 through 0 and the sample for channel 1 must be written to bits 15 through 8 in SDR0. SDR1 is not used in this mode. See Table 28-5 on page 699.

28.6.5 Data Swapping

When the Swap Channels (SWAP) bit in the Control Register (CR.SWAP) is one, writing to the Sample Data Register 0 (SDR0) will put the data in Sample Data Register 1 (SDR1). Writing SDR1 will put the data in SDR0. If one of the two compact stereo formats is used the lower and upper halfword of SDR0 will be swapped when writing to SDR0.

28.6.6 Common Mode Offset Control

When the Common Mode Offset Control (CMOC) bit in the Control Register is one the input data will get a DC value applied to it and the amplitude will be scaled. This will make the common mode offset of the two corresponding outputs, DAC and DACN, to move away from each other so that the output signals are not overlapping. The result is that the two signals can be applied to a differential analog filter, and the difference will always be a positive value, removing the need for a negative voltage supply for the filter. The cost of doing this a 3dB loss in dynamic range. On the left side of Figure 28-2 one can see the filtered output from the DAC and DACN pins when a sine wave is played when CR.CMOC is zero. The waveform on the right side shows the output of the differential filter when the two outputs on the left side are used as inputs to the differential filter. Figure 28-3 show the corresponding outputs when CR.CMOC is one.

29.9.9 Interrupt Enable Register

Name:	IER
Access Type:	Write-only
Offset:	0x20
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
-	CELSE	CGT	CLT	-	-	BUSY	READY

7	6	5	4	3	2	1	0
-	-	NOCNT	PENCNT	-	-	OVRE	DRDY

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

33.6.1 How to Initialize The Module

To initialize the aWire UART user interface the user must first enable the clock by writing a one to the Clock Enable bit in the Clock Request Register (CLKR.CLKEN) and wait for the Clock Enable bit in the Status Register (SR.CENABLED) to be set. After doing this either receive, transmit or receive with resync must be selected by writing the corresponding value into the Mode field of the Control (CTRL.MODE) Register. Due to the RC120M being asynchronous with the system clock values must be allowed to propagate in the system. During this time the aWire master will set the Busy bit in the Status Register (SR.BUSY).

After the SR.BUSY bit is cleared the Baud Rate field in the Baud Rate Register (BRR.BR) can be written with the wanted baudrate (f_{br}) according to the following formula (f_{aw} is the RC120M clock frequency):

$$f_{br} = \frac{8f_{aw}}{BR}$$

After this operation the user must wait until the SR.BUSY is cleared. The interface is now ready to be used.

33.6.2 Basic Asynchronous Receiver Operation

The aWire UART user interface must be initialized according to the sequence above, but the CTRL.MODE field must be written to one (Receive mode).

When a data byte arrives the aWire UART user interface will indicate this by setting the Data Ready Interrupt bit in the Status Register (SR.DREADYINT). The user must read the Data in the Receive Holding Register (RHR.RXDATA) and clear the Interrupt bit by writing a one to the Data Ready Interrupt Clear bit in the Status Clear Register (SCR.DREADYINT). The interface is now ready to receive another byte.

33.6.3 Basic Asynchronous Transmitter Operation

The aWire UART user interface must be initialized according to the sequence above, but the CTRL.MODE field must be written to two (Transmit mode).

To transmit a data byte the user must write the data to the Transmit Holding Register (THE.TXDATA). Before the next byte can be written the SR.BUSY must be cleared.

33.6.4 Basic Asynchronous Receiver with Resynchronization

By writing three into CTRL.MODE the aWire UART user interface will assume that the first byte it receives is a sync byte (0x55) and set BRR.BR according to this. All subsequent transfers will assume this baudrate, unless BRR.BR is rewritten by the user.

To make the aWire UART user interface accept a new sync resynchronization the aWire UART user interface must be disabled by writing zero to CTRL.MODE and then reenable the interface.

33.6.5 Overrun

In Receive mode an overrun can occur if the user has not read the previous received data from the RHR.RXDATA when the newest data should be placed there. Such a condition is flagged by setting the Overrun bit in the Status Register (SR.OVERRUN). If SR.OVERRUN is set the newest data received is placed in RHR.RXDATA and the data that was there before is overwritten.

33.6.6 Interrupts

To make the CPU able to do other things while waiting for the aWire UART user interface to finish its operations the aWire UART user interface supports generating interrupts. All status bits in the Status Register can be used as interrupt sources, except the SR.BUSY and SR.CENABLED bits.

To enable an interrupt the user must write a one to the corresponding bit in the Interrupt Enable Register (IER). Upon the next zero to one transition of this SR bit the aWire UART user interface will flag this interrupt to the CPU. To clear the interrupt the user must write a one to the corresponding bit in the Status Clear Register (SCR).

Interrupts can be disabled by writing a one to the corresponding bit in the Interrupt Disable Register (IDR). The interrupt Mask Register (IMR) can be read to check if an interrupt is enabled or disabled.

33.6.7 Using the Peripheral DMA Controller

To relieve the CPU of data transfers the aWire UART user interface support using the Peripheral DMA controller.

To transmit using the Peripheral DMA Controller do the following:

1. Setup the aWire UART user interface in transmit mode.
2. Setup the Peripheral DMA Controller with buffer address and length, use byte as transfer size.
3. Enable the Peripheral DMA Controller.
4. Wait until the Peripheral DMA Controller is done.

To receive using the Peripheral DMA Controller do the following:

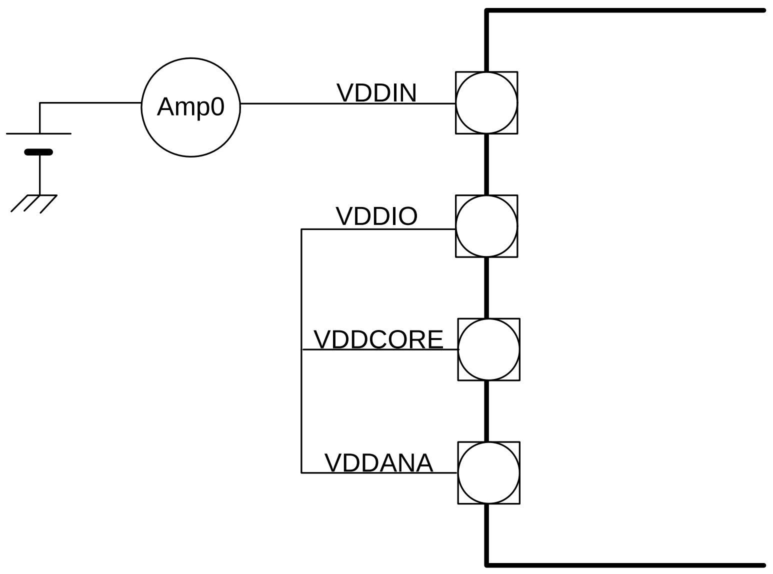
1. Setup the aWire UART user interface in receive mode
2. Setup the Peripheral DMA Controller with buffer address and length, use byte as transfer size.
3. Enable the Peripheral DMA Controller.
4. Wait until the Peripheral DMA Controller is ready.

Table 35-5. Power Consumption for Different Operating Modes

Mode	Conditions	Measured on	Consumption Typ	Unit
Active[(1)]	CPU running a recursive Fibonacci algorithm	Amp0	300	µA/MHz
	CPU running a division algorithm		174	
Idle[(1)]			96	
Frozen[(1)]			57	
Standby[(1)]			46	
Stop			38	µA
DeepStop			25	
Static	-OSC32K and AST stopped -Internal core supply		14	
	-OSC32K running -AST running at 1 KHz -External core supply (Figure 35-2)		7.3	
	-OSC32K and AST stopped -External core supply (Figure 35-2)		6.7	
Shutdown	-OSC32K running -AST running at 1 KHz		800	nA
	AST and OSC32K stopped		220	

Note: 1. These numbers are valid for the measured condition only and must not be extrapolated to other frequencies.

Figure 35-1. Measurement Schematic, Internal Core Supply



Under certain circumstances, the lock signal from the Phase Locked Loop (PLL) oscillator may not go back to zero after the PLL oscillator has been disabled. This can cause the propagation of clock signals with the wrong frequency to parts of the system that use the PLL clock.
Fix/Workaround
PLL must be turned off before entering STOP, DEEPSTOP or STATIC sleep modes. If PLL has been turned off, a delay of 30us must be observed after the PLL has been enabled again before the SCIF.PLL0LOCK bit can be used as a valid indication that the PLL is locked.

3. **PLLCOUNT value larger than zero can cause PLLEN glitch**
Initializing the PLLCOUNT with a value greater than zero creates a glitch on the PLLEN signal during asynchronous wake up.
Fix/Workaround
The lock-masking mechanism for the PLL should not be used.
The PLLCOUNT field of the PLL Control Register should always be written to zero.

4. **RCSYS is not calibrated**
The RCSYS is not calibrated and will run faster than 115.2kHz. Frequencies around 150kHz can be expected.
Fix/Workaround
If a known clock source is available the RCSYS can be runtime calibrated by using the frequency meter (FREQM) and tuning the RCSYS by writing to the RCCR register in SCIF.

5. **Writing 0x5A5A5A5A to the SCIF memory range will enable the SCIF UNLOCK feature**
The SCIF UNLOCK feature will be enabled if the value 0x5A5A5A5A is written to any location in the SCIF memory range.
Fix/Workaround
None.

38.5.4 WDT

1. **Clearing the Watchdog Timer (WDT) counter in second half of timeout period will issue a Watchdog reset**
If the WDT counter is cleared in the second half of the timeout period, the WDT will immediately issue a Watchdog reset.
Fix/Workaround
Use twice as long timeout period as needed and clear the WDT counter within the first half of the timeout period. If the WDT counter is cleared after the first half of the timeout period, you will get a Watchdog reset immediately. If the WDT counter is not cleared at all, the time before the reset will be twice as long as needed.

2. **WDT Control Register does not have synchronization feedback**
When writing to the Timeout Prescale Select (PSEL), Time Ban Prescale Select (TBAN), Enable (EN), or WDT Mode (MODE) fieldss of the WDT Control Register (CTRL), a synchronizer is started to propagate the values to the WDT clcok domain. This synchronization takes a finite amount of time, but only the status of the synchronization of the EN bit is reflected back to the user. Writing to the synchronized fields during synchronization can lead to undefined behavior.
Fix/Workaround
-When writing to the affected fields, the user must ensure a wait corresponding to 2 clock cycles of both the WDT peripheral bus clock and the selected WDT clock source.
-When doing writes that changes the EN bit, the EN bit can be read back until it reflects the written value.