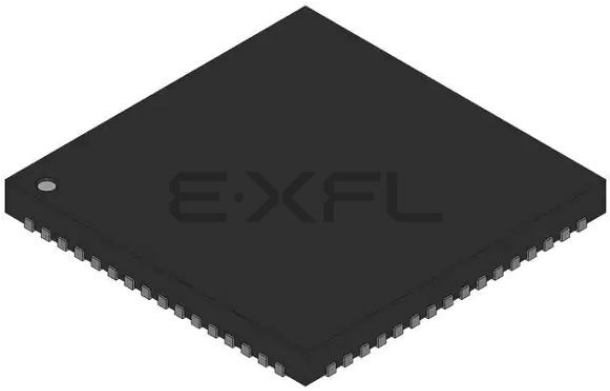

Welcome to E-XFL.COM

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-z3ut

Debug state can be entered as described in the *AVR32UC Technical Reference Manual*.

Debug state is exited by the *retd* instruction.

4.4.3.3 *Secure State*

The AVR32 can be set in a secure state, that allows a part of the code to execute in a state with higher security levels. The rest of the code can not access resources reserved for this secure code. Secure State is used to implement FlashVault technology. Refer to the *AVR32UC Technical Reference Manual* for details.

4.4.4 System Registers

The system registers are placed outside of the virtual memory space, and are only accessible using the privileged *mfsr* and *mtsr* instructions. The table below lists the system registers specified in the AVR32 architecture, some of which are unused in AVR32UC. The programmer is responsible for maintaining correct sequencing of any instructions following a *mtsr* instruction. For detail on the system registers, refer to the *AVR32UC Technical Reference Manual*.

Table 4-3. System Registers

Reg #	Address	Name	Function
0	0	SR	Status Register
1	4	EVBA	Exception Vector Base Address
2	8	ACBA	Application Call Base Address
3	12	CPUCR	CPU Control Register
4	16	ECR	Exception Cause Register
5	20	RSR_SUP	Unused in AVR32UC
6	24	RSR_INT0	Unused in AVR32UC
7	28	RSR_INT1	Unused in AVR32UC
8	32	RSR_INT2	Unused in AVR32UC
9	36	RSR_INT3	Unused in AVR32UC
10	40	RSR_EX	Unused in AVR32UC
11	44	RSR_NMI	Unused in AVR32UC
12	48	RSR_DBG	Return Status Register for Debug mode
13	52	RAR_SUP	Unused in AVR32UC
14	56	RAR_INT0	Unused in AVR32UC
15	60	RAR_INT1	Unused in AVR32UC
16	64	RAR_INT2	Unused in AVR32UC
17	68	RAR_INT3	Unused in AVR32UC
18	72	RAR_EX	Unused in AVR32UC
19	76	RAR_NMI	Unused in AVR32UC
20	80	RAR_DBG	Return Address Register for Debug mode
21	84	JECR	Unused in AVR32UC
22	88	JOSP	Unused in AVR32UC
23	92	JAVA_LV0	Unused in AVR32UC

If TCR is zero when writing to TCRR, the TCR and MAR are automatically updated with the value written in TCRR and MARR.

7.5.5 Ring Buffer

When Ring Buffer mode is enabled the TCRR and MARR registers will not be cleared when TCR and MAR registers reload. This allows the PDCA to read or write to the same memory region over and over again until the transfer is actively stopped by the user. Ring Buffer mode is enabled by writing a one to the Ring Buffer bit in the Mode Register (MR.RING).

7.5.6 Peripheral Selection

The Peripheral Select Register (PSR) decides which peripheral should be connected to the PDCA channel. A peripheral is selected by writing the corresponding Peripheral Identity (PID) to the PID field in the PSR register. Writing the PID will both select the direction of the transfer (memory to peripheral or peripheral to memory), which handshake interface to use, and the address of the peripheral holding register. Refer to the Peripheral Identity (PID) table in the Module Configuration section for the peripheral PID values.

7.5.7 Transfer Size

The transfer size can be set individually for each channel to be either byte, halfword or word (8-bit, 16-bit or 32-bit respectively). Transfer size is set by writing the desired value to the Transfer Size field in the Mode Register (MR.SIZE).

When the PDCA moves data between peripherals and memory, data is automatically sized and aligned. When memory is accessed, the size specified in MR.SIZE and system alignment is used. When a peripheral register is accessed the data to be transferred is converted to a word where bit n in the data corresponds to bit n in the peripheral register. If the transfer size is byte or halfword, bits greater than 8 and16 respectively are set to zero.

Refer to the Module Configuration section for information regarding what peripheral registers are used for the different peripherals and then to the peripheral specific chapter for information about the size option available for the different registers.

7.5.8 Enabling and Disabling

Each DMA channel is enabled by writing a one to the Transfer Enable bit in the Control Register (CR.TEN) and disabled by writing a one to the Transfer Disable bit (CR.TDIS). The current status can be read from the Status Register (SR).

While the PDCA channel is enabled all DMA request will be handled as long the TCR and TCRR is not zero.

7.5.9 Interrupts

Interrupts can be enabled by writing a one to the corresponding bit in the Interrupt Enable Register (IER) and disabled by writing a one to the corresponding bit in the Interrupt Disable Register (IDR). The Interrupt Mask Register (IMR) can be read to see whether an interrupt is enabled or not. The current status of an interrupt source can be read through the Interrupt Status Register (ISR).

The PDCA has three interrupt sources:

- Reload Counter Zero - The TCRR register is zero.
- Transfer Finished - Both the TCR and TCRR registers are zero.
- Transfer Error - An error has occurred in accessing memory.

7.7.19 Performance Channel 0 Read Stall Cycles

Name: PRSTALL0

Access Type: Read-only

Offset: 0x808

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
STALL[31:24]							

23	22	21	20	19	18	17	16
STALL[23:16]							

15	14	13	12	11	10	9	8
STALL[15:8]							

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

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

EPSIZE			Endpoint Size
1	0	1	256 bytes
1	1	0	512 bytes
1	1	1	1024 bytes

This field is cleared upon receiving a USB reset (except for the endpoint 0).

- **EPBK: Endpoint Banks**

  This bit selects the number of banks for the endpoint:

  0: single-bank endpoint

  1: double-bank endpoint

  For control endpoints, a single-bank endpoint shall be selected.

  This field is cleared upon receiving a USB reset (except for the endpoint 0).

8.8 Module Configuration

The specific configuration for each USBC instance is listed in the following tables. The module bus clocks listed here are connected to the system bus clocks. Please refer to the Power Manager chapter for details.

Table 8-6. USBC Clocks

Clock Name	Description
CLK_USBC_PB	Clock for the USBC PB interface
CLK_USBC_HSB	Clock for the USBC HSB interface
GCLK_USBC	The generic clock used for the USBC is GCLK7

Table 8-7. Register Reset Values

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

- **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

10.6.9 Interrupt Clear Register

Name: ICR

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
-	-	-	-	-	-	-	-

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

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 SR and any corresponding interrupt request.

13.7.6 PBA Divided Mask

Name: PBADIVMASK

Access Type: Read/Write

Offset: 0x040

Reset Value: 0x0000007F

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
-	MASK[6:0]						

- **MASK: Clock Mask**
  If bit n is written to zero, the clock divided by $2^{(n+1)}$ is stopped. If bit n is written to one, the clock divided by $2^{(n+1)}$ is enabled according to the current power mode. Table 13-10 shows what clocks are affected by the different MASK bits.

Table 13-10. Divided Clock Mask

Bit	USART0	USART1	USART2	USART3	TC0	TC1
0	-	-	-	-	TIMER_CLOCK2	TIMER_CLOCK2
1	-	-	-	-	-	-
2	CLK_USART/ DIV	CLK_USART/ DIV	CLK_USART/ DIV	CLK_USART/ DIV	TIMER_CLOCK3	TIMER_CLOCK3
3	-	-	-	-	-	-
4	-	-	-	-	TIMER_CLOCK4	TIMER_CLOCK4
5	-	-	-	-	-	-
6	-	-	-	-	TIMER_CLOCK5	TIMER_CLOCK5

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.

14.6.20 Temperature Sensor Configuration Register

Name: TSENS

Access Type: Read/Write

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
-	-	-			-	-	EN

- **EN: Temperature Sensor Enable**
  0: The Temperature Sensor is disabled.
  1: The Temperature Sensor 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.

14.6.27 Fractional Prescaler Control Register

Name: FPCR

Access Type: Read/Write

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
-	-	-	-	CKSEL			FPEN

- **CKSEL: Clock input selection**
  This field selects the Clock input for the prescaler. See the “FP clock sources” table in the SCIF Module Configuration section for details. It must not be changed if the FPEN is one.
- **FPEN: High Resolution Prescaler Enable**
  0: The Fractional Prescaler is disabled.
  1: The Fractional Prescaler is enabled.

14.6.32 PLL Version Register

Name: PLLVERSION

Access Type: Read-only

Offset: 0x03C4

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.

14.6.40 Temperature Sensor Version Register

Name: TSENSIFAVERSION

Access Type: Read-only

Offset: 0x03E4

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.

16.5.3 Disabling the WDT

The WDT is disabled by writing a zero to the CTRL.EN bit. When disabling the WDT no other bits in the CTRL Register should be changed until the CTRL.EN bit reads back as zero. If the CTRL.CEN bit is written to zero, the CTRL.EN bit will never read back as zero if changing the value from one to zero.

16.5.4 Flash Calibration

The WDT can be enabled at reset. This is controlled by the WDTAUTO fuse. The WDT will be set in basic mode, RCSYS is set as source for CLK_CNT, and PSEL will be set to a value giving T_{psel} above 100 ms. Please refer to the Fuse Settings chapter for details about WDTAUTO and how to program the fuses.

If the Flash Calibration Done (FCD) bit in the CTRL Register is zero at a watchdog reset the flash calibration will be redone, and the CTRL.FCD bit will be set when the calibration is done. If CTRL.FCD is one at a watchdog reset, the configuration of the WDT will not be changed during flash calibration. After any other reset the flash calibration will always be done, and the CTRL.FCD bit will be set when the calibration is done.

16.5.5 Special Considerations

Care must be taken when selecting the PSEL/TBAN values so that the timeout period is greater than the startup time of the device. Otherwise a watchdog reset will reset the device before any code has been run. This can also be avoided by writing the CTRL.DAR bit to one when configuring the WDT.

If the Store Final Value (SFV) bit in the CTRL Register is one, the CTRL Register is locked for further write accesses. All writes to the CTRL Register will be ignored. Once the CTRL Register is locked, it can only be unlocked by a reset (e.g. POR, OCD, and WDT).

The CTRL.MODE bit can only be changed when the WDT is disabled (CTRL.EN=0).

16.6 User Interface

Table 16-1. WDT Register Memory Map

Offset	Register	Register Name	Access	Reset
0x000	Control Register	CTRL	Read/Write	0x00010080
0x004	Clear Register	CLR	Write-only	0x00000000
0x008	Status Register	SR	Read-only	0x00000003
0x3FC	Version Register	VERSION	Read-only	-(1)

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

19.7.7 Peripheral Mux Register 2

Name: PMR2

Access: Read/Write, Set, Clear, Toggle

Offset: 0x030, 0x034, 0x038, 0x03C

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Peripheral Multiplexer Select bit 2**

{PMR2, PMR1, PMR0}	Selected Peripheral Function
000	A
001	B
010	C
011	D
100	E
101	F
110	G
111	H

- TFrame_Maximum = 1.4 x (THeader_Nominal + TResponse_Nominal + 1)(Note:)

Note: The term “+1” leads to an integer result for TFrame_Max (LIN Specification 1.3)

If the Checksum is sent (CHKDIS=0):

- TResponse_Nominal = 10 x (NData + 1)
- TFrame_Maximum = 1.4 x (34 + 10 x (DLC + 1 + 1) + 1)
- TFrame_Maximum = 77 + 14 x DLC

If the Checksum is not sent (CHKDIS=1):

- TResponse_Nominal = 10 x NData
- TFrame_Maximum = 1.4 x (34 + 10 x (DLC + 1) + 1)
- TFrame_Maximum = 63 + 14 x DLC

20.6.6 LIN Errors

These error bits are cleared by writing a one to CSR.RSTSTA.

20.6.6.1 Slave Not Responding Error (CSR.LINSNRE)

This error is generated if no valid message appears within the TFrame_Maximum time frame slot, while the USART is expecting a response from another node (NACT=SUBSCRIBE).

20.6.6.2 Checksum Error (CSR.LINCE)

This error is generated if the received checksum is wrong. This error can only be generated if the checksum feature is enabled (CHKDIS=0).

20.6.6.3 Identifier Parity Error (CSR.LINIPE)

This error is generated if the identifier parity is wrong. This error can only be generated if parity is enabled (PARDIS=0).

20.6.6.4 Inconsistent Sync Field Error (CSR.LINISFE)

This error is generated in slave mode if the Sync Field character received is not 0x55. Synchronization procedure is aborted.

20.6.6.5 Bit Error (CSR.LINBE)

This error is generated if the value transmitted by the USART on Tx differs from the value sampled on Rx. If a bit error is detected, the transmission is aborted at the next byte border.

20.6.7 LIN Frame Handling

20.6.7.1 Master Node Configuration

- Write a one to CR.TXEN and CR.RXEN to enable both transmitter and receiver
- Select LIN mode and master node by writing to MR.MODE
- Configure the baud rate by writing to CD and FP in BRGR
- Configure the frame transfer by writing to NACT, PARDIS, CHKDIS, CHKTYPE, DLCM, FSDIS, and DLC in LINMR
- Check that CSR.TXRDY is one
- Send the header by writing to LINIR.IDCHR

The following procedure depends on the NACT setting:

- **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.

23.9.2 NBYTES Register

Name: NBYTES

Access Type: Read/Write

Offset: 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
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
NBYTES							

- **NBYTES: Number of Bytes to Transfer**

  Writing to this field updates the NBYTES counter. The field can also be read to learn the progress of the transfer. NBYTES can be incremented or decremented automatically by hardware.

29.9.14 Version Register

Name: VERSION

Access Type: Read-only

Offset: 0x34

Reset Value: 0x00000000

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.

which is linked to the JTAG through a bus master module, which also handles synchronization between the TCK and SAB clocks.

For more information about the SAB and a list of SAB slaves see the Service Access Bus chapter.

34.4.11.1 SAB Address Mode

The MEMORY_SIZED_ACCESS instruction allows a sized read or write to any 36-bit address on the bus. MEMORY_WORD_ACCESS is a shorthand instruction for 32-bit accesses to any 36-bit address, while the NEXUS_ACCESS instruction is a Nexus-compliant shorthand instruction for accessing the 32-bit OCD registers in the 7-bit address space reserved for these. These instructions require two passes through the Shift-DR TAP state: one for the address and control information, and one for data.

34.4.11.2 Block Transfer

To increase the transfer rate, consecutive memory accesses can be accomplished by the MEMORY_BLOCK_ACCESS instruction, which only requires a single pass through Shift-DR for data transfer only. The address is automatically incremented according to the size of the last SAB transfer.

34.4.11.3 Canceling a SAB Access

It is possible to abort an ongoing SAB access by the CANCEL_ACCESS instruction, to avoid hanging the bus due to an extremely slow slave.

34.4.11.4 Busy Reporting

As the time taken to perform an access may vary depending on system activity and current chip frequency, all the SAB access JTAG instructions can return a busy indicator. This indicates whether a delay needs to be inserted, or an operation needs to be repeated in order to be successful. If a new access is requested while the SAB is busy, the request is ignored.

The SAB becomes busy when:

- Entering Update-DR in the address phase of any read operation, e.g., after scanning in a NEXUS_ACCESS address with the read bit set.
- Entering Update-DR in the data phase of any write operation, e.g., after scanning in data for a NEXUS_ACCESS write.
- Entering Update-DR during a MEMORY_BLOCK_ACCESS.
- Entering Update-DR after scanning in a counter value for SYNC.
- Entering Update-IR after scanning in a MEMORY_BLOCK_ACCESS if the previous access was a read and data was scanned after scanning the address.

The SAB becomes ready again when:

- A read or write operation completes.
- A SYNC countdown completed.
- A operation is cancelled by the CANCEL_ACCESS instruction.

What to do if the busy bit is set:

- During Shift-IR: The new instruction is selected, but the previous operation has not yet completed and will continue (unless the new instruction is CANCEL_ACCESS). You may