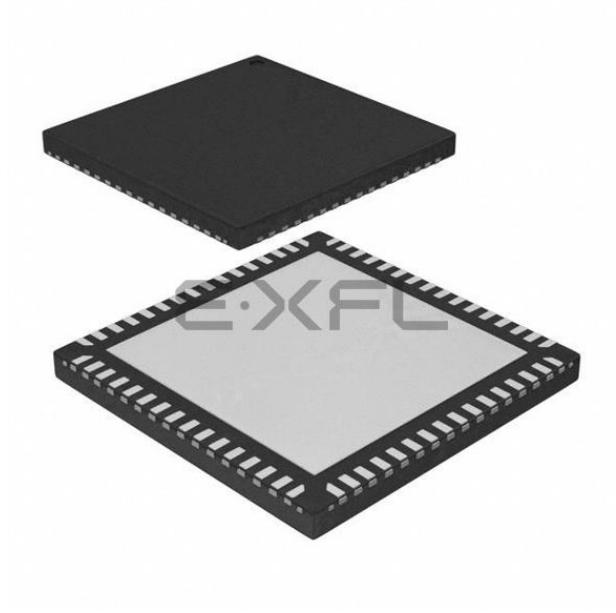


Welcome to E-XFL.COM

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	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	32K 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/microchip-technology/atuc256l3u-z3ur

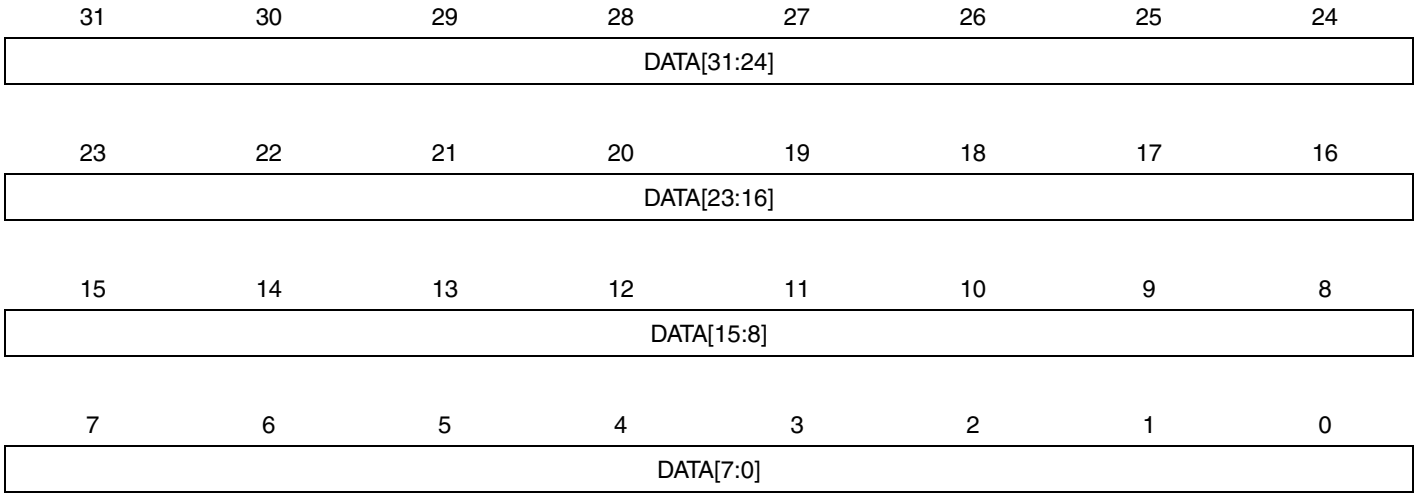
7.7.18 Performance Channel 0 Read Data Cycles

Name: PRDATA0

Access Type: Read-only

Offset: 0x804

Reset Value: 0x00000000



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

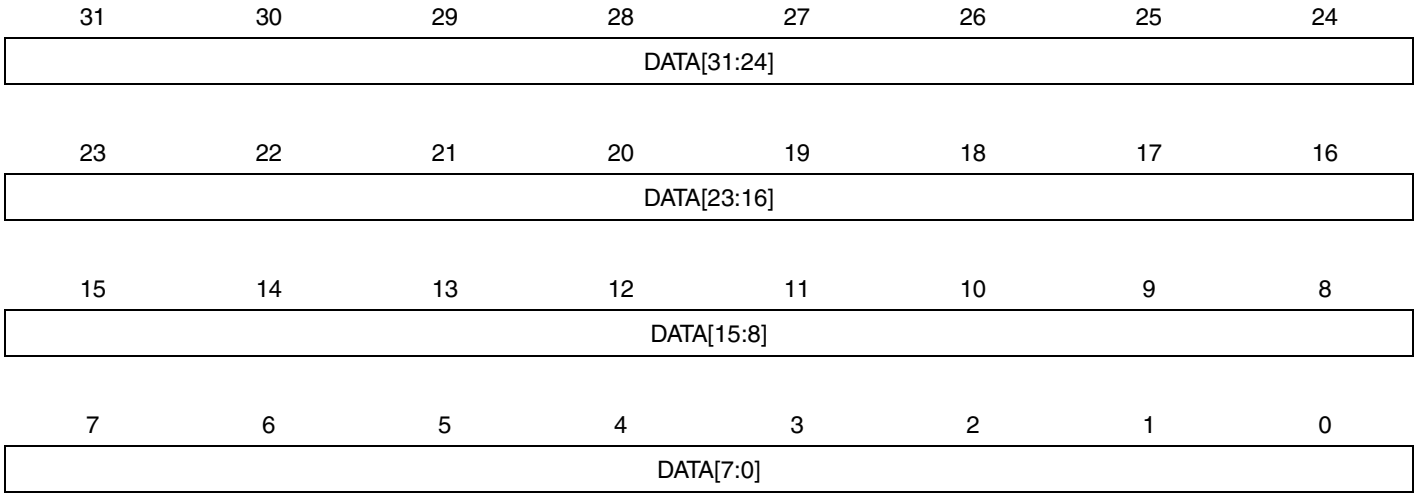
7.7.21 Performance Channel 0 Write Data Cycles

Name: PWDATA0

Access Type: Read-only

Offset: 0x810

Reset Value: 0x00000000



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

- Programming Error: A bad keyword and/or an invalid command have been written in the FCMD register.
- Lock Error: At least one lock region is protected, or BOOTPROT is different from 0. The erase command has been aborted and no page has been erased. A “Unlock region containing given page” (UP) command must be executed to unlock any locked regions.

9.5.3 Region Lock Bits

The flash memory has *p* pages, and these pages are grouped into 16 lock regions, each region containing *p*/16 pages. Each region has a dedicated lock bit preventing writing and erasing pages in the region. After production, the device may have some regions locked. These locked regions are reserved for a boot or default application. Locked regions can be unlocked to be erased and then programmed with another application or other data.

To lock or unlock a region, the commands Lock Region Containing Page (LP) and Unlock Region Containing Page (UP) are provided. Writing one of these commands, together with the number of the page whose region should be locked/unlocked, performs the desired operation.

One error can be detected in the FSR register after issuing the command:

- Programming Error: A bad keyword and/or an invalid command have been written in the FCMD register.

The lock bits are implemented using the lowest 16 general-purpose fuse bits. This means that lock bits can also be set/cleared using the commands for writing/erasing general-purpose fuse bits, see Section 9.6. The general-purpose bit being in an erased (1) state means that the region is unlocked.

The lowermost pages in the flash can additionally be protected by the BOOTPROT fuses, see Section 9.6.

9.6 General-purpose Fuse Bits

The flash memory has a number of general-purpose fuse bits that the application programmer can use freely. The fuse bits can be written and erased using dedicated commands, and read

12.6 User Interface

Table 12-1. INTC Register Memory Map

Offset	Register	Register Name	Access	Reset
0x000	Interrupt Priority Register 0	IPR0	Read/Write	0x00000000
0x004	Interrupt Priority Register 1	IPR1	Read/Write	0x00000000
...	...	...	...	...
0x0FC	Interrupt Priority Register 63	IPR63	Read/Write	0x00000000
0x100	Interrupt Request Register 0	IRR0	Read-only	N/A
0x104	Interrupt Request Register 1	IRR1	Read-only	N/A
...	...	...	...	...
0x1FC	Interrupt Request Register 63	IRR63	Read-only	N/A
0x200	Interrupt Cause Register 3	ICR3	Read-only	N/A
0x204	Interrupt Cause Register 2	ICR2	Read-only	N/A
0x208	Interrupt Cause Register 1	ICR1	Read-only	N/A
0x20C	Interrupt Cause Register 0	ICR0	Read-only	N/A

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.

Factor (PLLDIV) fields must be written with the multiplication and division factors, respectively. The PLLMUL must always be greater than 1, creating the PLL frequency:

$f_{vco} = (PLLMUL+1)/PLLDIV \bullet f_{REF}$, if PLLDIV >0

$f_{vco} = 2\bullet(PLLMUL+1) \bullet f_{REF}$, if PLLDIV = 0

The PLL Options (PLLOPT) field should be configured to proper values according to the PLL operating frequency. The PLLOPT field can also be configured to divide the output frequency of the PLL by 2 and Wide-Bandwidth mode, which allows faster startup time and out-of-lock time.

It is not possible to change any of the PLL configuration bits when the PLL is enabled, Any write to PLLn while the PLL is enabled will be discarded.

After setting up the PLL, the PLL is enabled by writing a one to the PLL Enable (PLLEN) bit in the PLLn register.

14.5.3.2 Disabling the PLL

The PLL is disabled by writing a zero to the PLL Enable (PLLEN) bit in the PLLn register. After disabling the PLL, the PLL configuration fields becomes writable.

14.5.3.3 PLL Lock

The lock signal for each PLL is available as a PLLLOCKn flag in the PCLKSR register. If the lock for some reason is lost, the PLLLOCKLOSTn flag in PCLKSR register will be set. An interrupt can be generated on a 0 to 1 transition of these bits.

14.5.4 Digital Frequency Locked Loop (DFLL) Operation

Rev: 2.1.0.1

The DFLL is controlled by the Digital Frequency Locked Loop Interface (DFLLIF). The DFLL is disabled by default, but can be enabled to provide a high-frequency source clock for synchronous and generic clocks.

Features:

- Internal oscillator with no external components
- 20-150MHz frequency in closed loop mode
- Can operate standalone as a high-frequency programmable oscillator in open loop mode
- Can operate as an accurate frequency multiplier against a known frequency in closed loop mode
- Optional spread-spectrum clock generation
- Very high-frequency multiplication supported - can generate all frequencies from a 32KHz clock

The DFLL can operate in both open loop mode and closed loop mode. In closed loop mode a low frequency clock with high accuracy can be used as reference clock to get high accuracy on the output clock (CLK_DFLL).

To prevent unexpected writes due to software bugs, write access to the configuration registers is protected by a locking mechanism. For details please refer to the UNLOCK register description.

14.6.3 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x0008

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
AE	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	PLLLOCKLOST0	PLLLOCK0	BRIFARDY
15	14	13	12	11	10	9	8
DFLL0RCS	DFLL0RDY	DFLL0LOCKLOSTA	DFLL0LOCKLOSTF	DFLL0LOCKLOSTC	DFLL0LOCKA	DFLL0LOCKF	DFLL0LOCKC
7	6	5	4	3	2	1	0
BODDET	SM33DET	VREGOK	-	-	-	OSC0RDY	OSC32RDY

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.

14.6.31 GCLK Prescaler Version Register

Name: GCLKPRESCVERSION

Access Type: Read-only

Offset: 0x03C0

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.

slave to pull it down in order to generate the acknowledge. The master polls the data line during this clock pulse.

The SR.RXRDY bit indicates that a data byte is available in the RHR. The RXRDY bit is also used as Receive Ready for the Peripheral DMA Controller receive channel.

Figure 23-10. Slave Receiver with One Data Byte

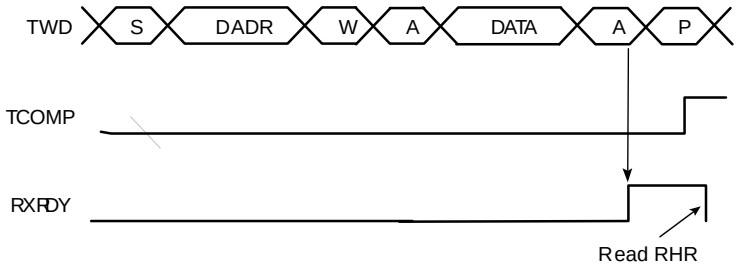
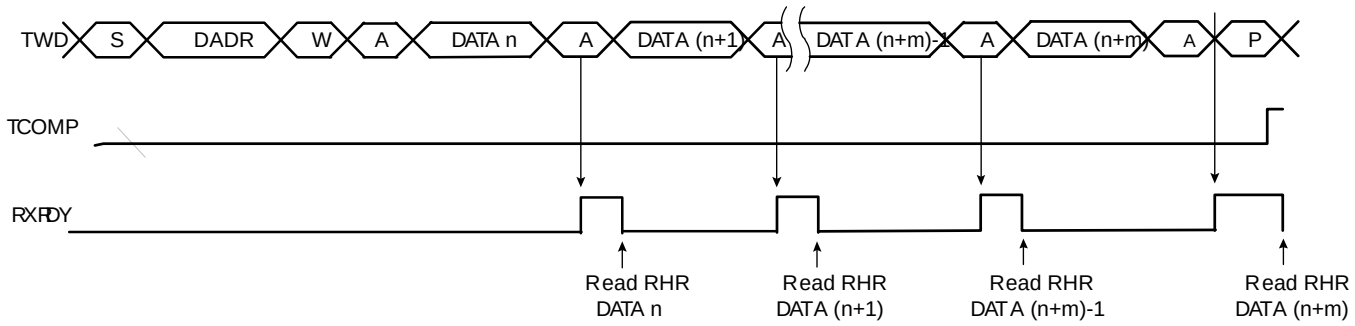


Figure 23-11. Slave Receiver with Multiple Data Bytes



23.8.5 Interactive ACKing Received Data Bytes

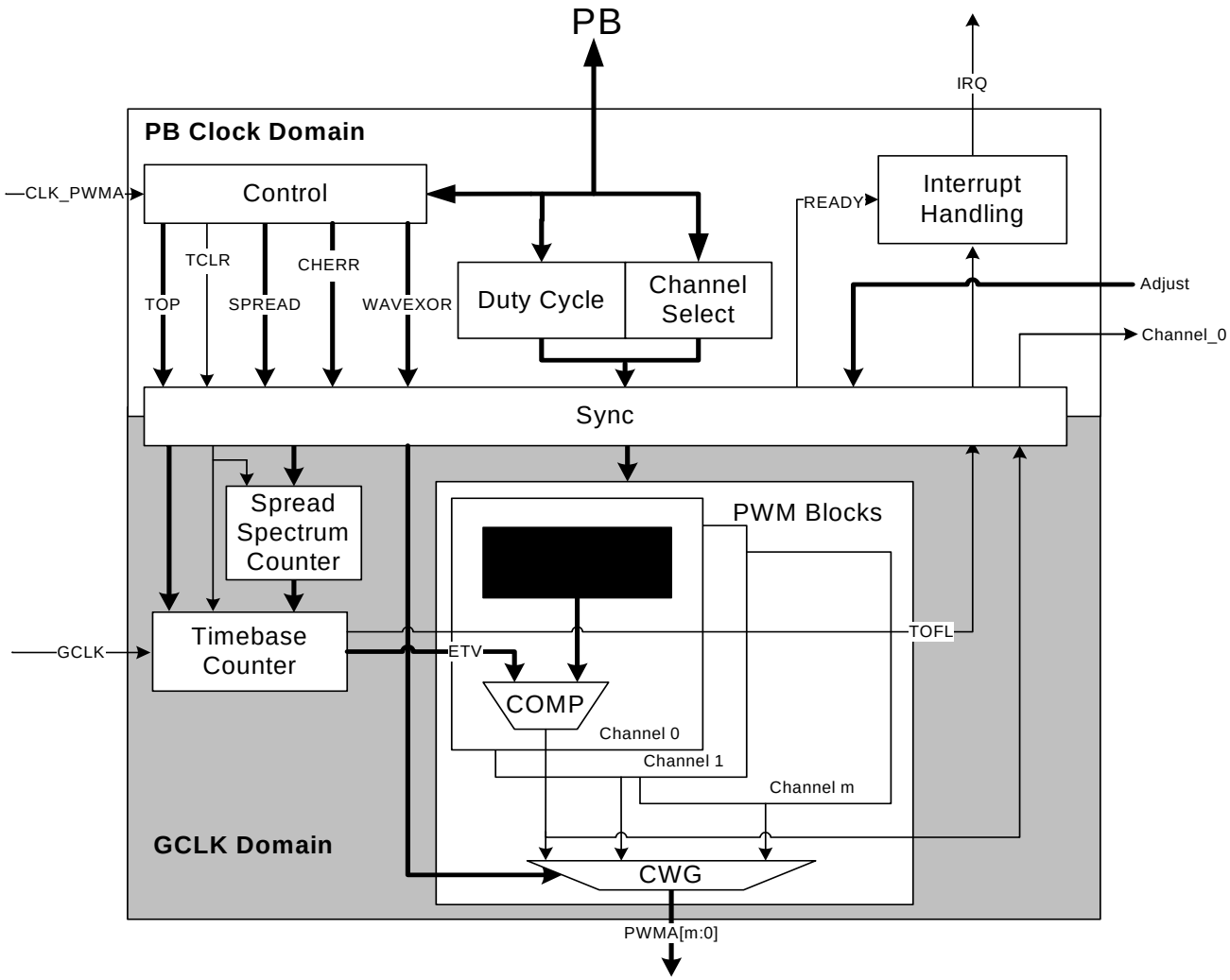
When implementing a register interface over TWI, it may sometimes be necessary or just useful to report reads and writes to invalid register addresses by sending a NAK to the host. To be able to do this, one must first receive the register address from the TWI bus, and then tell the TWIS whether to ACK or NAK it. In normal operation of the TWIS, this is not possible because the controller will automatically ACK the byte at about the same time as the RXRDY bit changes from zero to one. Writing a one to the Stretch on Data Byte Received bit (CR.SODR) will stretch the clock allowing the user to update CR.ACK bit before returning the desired value. After the last bit in the data byte is received, the TWI bus clock is stretched, the received data byte is transferred to the RHR register, and SR.BTF is set. At this time, the user can examine the received byte and write the desired ACK or NACK value to CR.ACK. When the user clears SR.BTF, the desired ACK value is transferred on the TWI bus. This makes it possible to look at the byte received, determine if it is valid, and then decide to ACK or NAK it.

23.8.6 Using the Peripheral DMA Controller

The use of the Peripheral DMA Controller significantly reduces the CPU load. The user can set up ring buffers for the Peripheral DMA Controller, containing data to transmit or free buffer space to place received data. By initializing NBYTES to zero before a transfer, and writing a one to CR.CUP, NBYTES is incremented by one each time a data has been transmitted or received. This allows the user to detect how much data was actually transferred by the DMA system.

25.3 Block Diagram

Figure 25-1. PWMA Block Diagram



25.4 I/O Lines Description

Each channel outputs one PWM waveform on one external I/O line.

Table 25-1. I/O Line Description

Pin Name	Pin Description	Type
PWMA[n]	Output PWM waveform for one channel n	Output
PWMMOD[n]	Output PWM waveform for one channel n, open drain mode	Output

25.5 Product Dependencies

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

25.7.10 Parameter Register

Name: PARAMETER

Access Type: Read-only

Offset: 0x24

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
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
CHANNELS							

- **CHANNELS: Channels Implemented**
  This field contains the number of channels implemented on the device.

27. Peripheral Event System

Rev: 1.0.0.1

27.1 Features

- **Direct peripheral to peripheral communication system**
- **Allows peripherals to receive, react to, and send peripheral events without CPU intervention**
- **Cycle deterministic event communication**
- **Asynchronous interrupts allow advanced peripheral operation in low power sleep modes**

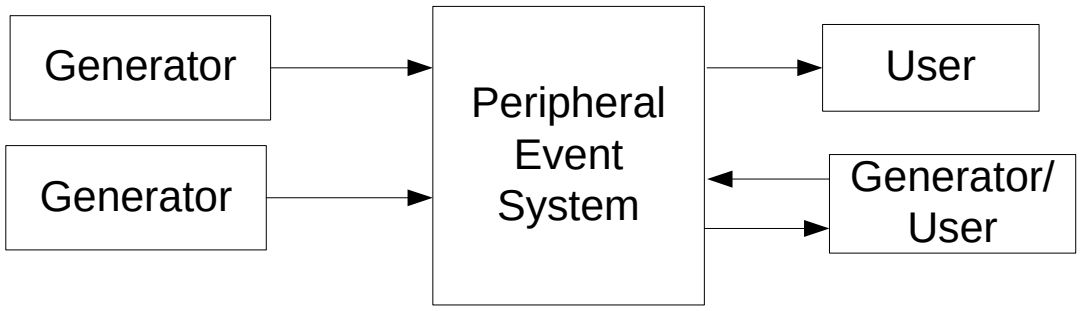
27.2 Overview

Several peripheral modules can be configured to emit or respond to signals known as peripheral events. The exact condition to trigger a peripheral event, or the action taken upon receiving a peripheral event, is specific to each module. Peripherals that respond to peripheral events are called peripheral event users and peripherals that emit peripheral events are called peripheral event generators. A single module can be both a peripheral event generator and user.

The peripheral event generators and users are interconnected by a network known as the Peripheral Event System. This allows low latency peripheral-to-peripheral signaling without CPU intervention, and without consuming system resources such as bus or RAM bandwidth. This offloads the CPU and system resources compared to a traditional interrupt-based software driven system.

27.3 Peripheral Event System Block Diagram

Figure 27-1. Peripheral Event System Block Diagram



27.4 Functional Description

27.4.1 Configuration

The Peripheral Event System in the ATUC64/128/256L3/4U has a fixed mapping of peripheral events between generators and users, as described in Table 27-1 to Table 27-4. Thus, the user does not need to configure the interconnection between the modules, although each peripheral event can be enabled or disabled at the generator or user side as described in the peripheral chapter for each module.

29. ADC Interface (ADCIFB)

Rev:1.0.1.1

29.1 Features

- **Multi-channel Analog-to-Digital Converter with up to 12-bit resolution**
- **Enhanced Resolution Mode**
  - **11-bit resolution obtained by interpolating 4 samples**
  - **12-bit resolution obtained by interpolating 16 samples**
- **Glueless interface with resistive touch screen panel, allowing**
  - **Resistive Touch Screen position measurement**
  - **Pen detection and pen loss detection**
- **Integrated enhanced sequencer**
  - **ADC Mode**
  - **Resistive Touch Screen Mode**
- **Numerous trigger sources**
  - **Software**
  - **Embedded 16-bit timer for periodic trigger**
  - **Pen detect trigger**
  - **Continuous trigger**
  - **External trigger, rising, falling, or any-edge trigger**
  - **Peripheral event trigger**
- **ADC Sleep Mode for low power ADC applications**
- **Programmable ADC timings**
  - **Programmable ADC clock**
  - **Programmable startup time**

29.2 Overview

The ADC Interface (ADCIFB) converts analog input voltages to digital values. The ADCIFB is based on a Successive Approximation Register (SAR) 10-bit Analog-to-Digital Converter (ADC). The conversions extend from 0V to ADVREFP.

The ADCIFB supports 8-bit and 10-bit resolution mode, in addition to enhanced resolution mode with 11-bit and 12-bit resolution. Conversion results are reported in a common register for all channels.

The 11-bit and 12-bit resolution modes are obtained by interpolating multiple samples to acquire better accuracy. For 11-bit mode 4 samples are used, which gives an effective sample rate of 1/4 of the actual sample frequency. For 12-bit mode 16 samples are used, giving a effective sample rate of 1/16 of actual. This arrangement allows conversion speed to be traded for better accuracy.

Conversions can be started for all enabled channels, either by a software trigger, by detection of a level change on the external trigger pin (TRIGGER), or by an integrated programmable timer.

When the Resistive Touch Screen Mode is enabled, an integrated sequencer automatically configures the pad control signals and performs resistive touch screen conversions.

The ADCIFB also integrates an ADC Sleep Mode, a Pen-Detect Mode, and an Analog Compare Mode, and connects with one Peripheral DMA Controller channel. These features reduce both power consumption and processor intervention.

29.9.17 Channel Status Register

Name: CHSR

Access Type: Read-only

Offset: 0x48

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24
23	22	21	20	19	18	17	16
CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
15	14	13	12	11	10	9	8
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHn: Channel N Status**

  0: The corresponding channel is disabled.

  1: The corresponding channel is enabled.

  A bit in this register is cleared by writing a one to the corresponding bit in Channel Disable Register (CHDR).

  A bit in this register is set by writing a one to the corresponding bit in Channel Enable Register (CHER).

  The number of available channels is device dependent. Please refer to the Module Configuration section at the end of this chapter for information regarding how many channels are implemented.

31. Capacitive Touch Module (CAT)

Rev: 4.0.0.0

31.1 Features

- **QTouch® method allows N touch sensors to be implemented using 2N physical pins**
- **QMatrix method allows X by Y matrix of sensors to be implemented using (X+2Y) physical pins**
- **One autonomous QTouch sensor operates without DMA or CPU intervention**
- **All QTouch sensors can operate in DMA-driven mode without CPU intervention**
- **External synchronization to reduce 50 or 60 Hz mains interference**
- **Spread spectrum sensor drive capability**

31.2 Overview

The Capacitive Touch Module (CAT) senses touch on external capacitive touch sensors. Capacitive touch sensors use no external mechanical components, and therefore demand less maintenance in the user application.

The module implements the QTouch method of capturing signals from capacitive touch sensors. The QTouch method is generally suitable for small numbers of sensors since it requires 2 physical pins per sensor. The module also implements the QMatrix method, which is more appropriate for large numbers of sensors since it allows an X by Y matrix of sensors to be implemented using only (X+2Y) physical pins. The module allows methods to function together, so N touch sensors and an X by Y matrix of sensors can be implemented using (2N+X+2Y) physical pins.

In addition, the module allows sensors using the QTouch method to be divided into two groups. Each QTouch group can be configured with different properties. This eases the implementation of multiple kinds of controls such as push buttons, wheels, and sliders.

All of the QTouch sensors can operate in a DMA-driven mode, known as DMATouch, that allows detection of touch without CPU intervention. The module also implements one autonomous QTouch sensor that is capable of detecting touch without DMA or CPU intervention. This allows proximity or activation detection in low-power sleep modes.

31.7.23 CSA Resistor Control Register

Name: CSARES

Access Type: Read/Write

Offset: 0x64

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-							

23	22	21	20	19	18	17	16
-							RES[16]

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

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

- **RES: Resistive Drive Enable**
  When RES[n] is 0, CSA[n] has the same drive properties as normal I/O pads.
  When RES[n] is 1, CSA[n] has a nominal output resistance of 1kOhm during the burst phase.

33.7 User Interface

Table 33-2. aWire UART user interface Register Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CTRL	Read/Write	0x00000000
0x04	Status Register	SR	Read-only	0x00000000
0x08	Status Clear Register	SCR	Write-only	-
0x0C	Interrupt Enable Register	IER	Write-only	-
0x10	Interrupt Disable Register	IDR	Write-only	-
0x14	Interrupt Mask Register	IMR	Read-only	0x00000000
0x18	Receive Holding Register	RHR	Read-only	0x00000000
0x1C	Transmit Holding Register	THR	Read/Write	0x00000000
0x20	Baud Rate Register	BRR	Read/Write	0x00000000
0x24	Version Register	VERSION	Read-only	-(1)
0x28	Clock Request Register	CLKR	Read/Write	0x00000000

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

1. 0x55 (sync)
2. 0xC1 (command)
3. 0x00 (length MSB)
4. 0x07 (length LSB)
5. 0xCA (Data MSB)
6. 0xFE
7. 0xBA
8. 0xBE (Data LSB)
9. 0x00 (Status byte)
10. 0x00 (Bytes remaining MSB)
11. 0x00 (Bytes remaining LSB)
12. 0xXX (CRC MSB)
13. 0xXX (CRC LSB)

The status is 0x00 and all data read are valid. An unsuccessful four byte read can look like this:

1. 0x55 (sync)
2. 0xC1 (command)
3. 0x00 (length MSB)
4. 0x07 (length LSB)
5. 0xCA (Data MSB)
6. 0xFE
7. 0xXX (An error has occurred. Data read is undefined. 5 bytes remaining of the Data field)
8. 0xXX (More undefined data)
9. 0x02 (Status byte)
10. 0x00 (Bytes remaining MSB)
11. 0x05 (Bytes remaining LSB)
12. 0xXX (CRC MSB)
13. 0xXX (CRC LSB)

The error occurred after reading 2 bytes on the SAB. The rest of the bytes read are undefined. The status byte indicates the error and the bytes remaining indicates how many bytes were remaining to be sent of the data field of the packet when the error occurred.

Table 34-52. MEMDATA Status Byte

status byte	Description
0x00	Read successful
0x01	SAB busy
0x02	Bus error (wrong address)
Other	Reserved

Table 34-53. MEMDATA Details

Response	Details
Response value	0xC1
Additional data	Data read, status byte, and byte count (2 bytes)

39. Datasheet Revision History

Please note that the referring page numbers in this section are referred to this document. The referring revision in this section are referring to the document revision.

39.1 Rev. D – 06/2013

1. Updated the datasheet with a new ATmel blue logo and the last page.
2. Added Flash errata.

39.2 Rev. C – 01/2012

1. Description: DFLL frequency is 20 to 150MHz, not 40 to 150MHz.
2. Block Diagram: GCLK_IN is input, not output. CAT SMP corrected from I/O to output. SPI NPCS corrected from output to I/O.
3, Package and Pinout: EXTINT0 in Signal Descriptions table is NMI.
4, Supply and Startup Considerations: In 1.8V single supply mode figure, the input voltage is 1.62-1.98V, not 1.98-3.6V. “On system start-up, the DFLL is disabled” is replaced by “On system start-up, all high-speed clocks are disabled”.
5, ADCIFB: PRND signal removed from block diagram.
6, Electrical Charateristics: Added 64-pin package information to I/O Pin Characteristics tables and Digital Clock Characteristics table.
7, Mechanical Characteristics: QFN48 Package Drawing updated. Note that the package drawing for QFN48 is correct in datasheet rev A, but wrong in rev B. Added notes to package drawings.
8. Summary: Removed Programming and Debugging chapter, added Processor and Architecture chapter.

39.3 Rev. B – 12/2011

1. JTAG Data Registers subchapter added in the Programming and Debugging chapter, containing JTAG IDs.

39.4 Rev. A – 12/2011

1. Initial revision.