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	ARM® Cortex®-M3
Core Size	32-Bit Single-Core
Speed	96MHz
Connectivity	EBI/EMI, I²C, Memory Card, SPI, SSC, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, I²S, POR, PWM, WDT
Number of I/O	57
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	52K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 4x10b, 4x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-TFBGA
Supplier Device Package	100-TFBGA (9x9)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam3u4ca-cu

4. Package and Pinout

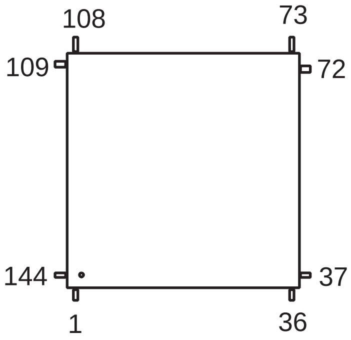
SAM3U4E / SAM3U2E / SAM3U1E devices are available in 144-lead LQFP and 144-ball LFBGA packages.

SAM3U4C / SAM3U2C / SAM3U1C devices are available in 100-lead LQFP and 100-ball TFBGA packages.

4.1 Package and Pinout (SAM3U4E / SAM3U2E / SAM3U1E Devices)

4.1.1 144-lead LQFP Package Outline

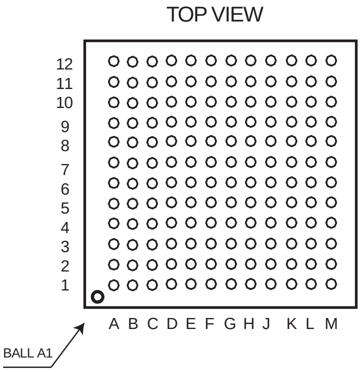
Figure 4-1. Orientation of the 144-lead LQFP Package



See Section 43.3 “144-lead LQFP Package” for mechanical drawings and specifications.

4.1.2 144-ball LFBGA Package Outline

Figure 4-2. Orientation of the 144-ball LFBGA Package



See Section 43.4 “144-ball LFBGA Package” for mechanical drawings and specifications.

The PSR bit assignments are:

31	30	29	28	27	26	25	24
N	Z	C	V	Q	ICI/IT		T

23	22	21	20	19	18	17	16
Reserved							

15	14	13	12	11	10	9	8
ICI/IT						Reserved	ISR_NUMBER

7	6	5	4	3	2	1	0
ISR_NUMBER							

Access these registers individually or as a combination of any two or all three registers, using the register name as an argument to the MSR or MRS instructions. For example:

- read all of the registers using PSR with the MRS instruction
- write to the APSR using APSR with the MSR instruction.

The PSR combinations and attributes are:

Table 12-3. PSR register combinations

Register	Type	Combination
PSR	RW [1], [2]	APSR, EPSR, and IPSR
IEPSR	RO	EPSR and IPSR
IAPSR	RW[1]	APSR and IPSR
EAPSR	RW[2]	APSR and EPSR

[1] The processor ignores writes to the IPSR bits.

[2] Reads of the EPSR bits return zero, and the processor ignores writes to the these bits.

See the instruction descriptions “MRS” on page 141 and “MSR” on page 142 for more information about how to access the program status registers.

12.3.3.6 Application Program Status Register

The APSR contains the current state of the condition flags from previous instruction executions. See the register summary in Table 12-2 on page 48 for its attributes. The bit assignments are:

- **N**

Negative or less than flag:

0 = operation result was positive, zero, greater than, or equal

1 = operation result was negative or less than.

- **Z**

Zero flag:

0 = operation result was not zero

1 = operation result was zero.

12.4.3.1 Additional memory access constraints for shared memory

When a system includes shared memory, some memory regions have additional access constraints, and some regions are subdivided, as Table 12-5 shows:

Table 12-5. Memory region share ability policies

Address range	Memory region	Memory type	Shareability	
0x00000000- 0x1FFFFFFF	Code	Normal [(1)]	-	
0x20000000- 0x3FFFFFFF	SRAM	Normal[(1)]	-	
0x40000000- 0x5FFFFFFF	Peripheral [(2)]	Device[(1)]	-	
0x60000000- 0x7FFFFFFF	External RAM	Normal[(1)]	-	WBWA[(2)]
0x80000000- 0x9FFFFFFF				WT[(2)]
0xA0000000- 0xBFFFFFFF	External device	Device[(1)]	Shareable[(1)]	-
0xC0000000- 0xDFFFFFFF			Non-shareable[(1)]	
0xE0000000- 0xE00FFFFF	Private Peripheral Bus	Strongly- ordered[(1)]	Shareable[(1)]	-
0xE0100000- 0xFFFFFFFF	Vendor-specific device[(2)]	Device[(1)]	-	-

1. See “Memory regions, types and attributes” on page 58 for more information.
2. The Peripheral and Vendor-specific device regions have no additional access constraints.

12.4.4 Software ordering of memory accesses

The order of instructions in the program flow does not always guarantee the order of the corresponding memory transactions. This is because:

- the processor can reorder some memory accesses to improve efficiency, providing this does not affect the behavior of the instruction sequence.
- the processor has multiple bus interfaces
- memory or devices in the memory map have different wait states
- some memory accesses are buffered or speculative.

“Memory system ordering of memory accesses” on page 59 describes the cases where the memory system guarantees the order of memory accesses. Otherwise, if the order of memory accesses is critical, software must include memory barrier instructions to force that ordering. The processor provides the following memory barrier instructions:

12.4.4.1 DMB

The *Data Memory Barrier* (DMB) instruction ensures that outstanding memory transactions complete before subsequent memory transactions. See “DMB” on page 138.

- do not branch to any instruction inside an IT block, except when returning from an exception handler
- all conditional instructions except B*cond* must be inside an IT block. B*cond* can be either outside or inside an IT block but has a larger branch range if it is inside one
- each instruction inside the IT block must specify a condition code suffix that is either the same or logical inverse as for the other instructions in the block.

Your assembler might place extra restrictions on the use of IT blocks, such as prohibiting the use of assembler directives within them.

12.16.3.4 Condition flags

This instruction does not change the flags.

12.16.3.5 Example

```
ITTE   NE           ; Next 3 instructions are conditional
ANDNE  R0, R0, R1   ; ANDNE does not update condition flags
ADDSNE R2, R2, #1   ; ADDSNE updates condition flags
MOVEQ  R2, R3       ; Conditional move


CMP    R0, #9       ; Convert R0 hex value (0 to 15) into ASCII
                    ; ('0'-'9', 'A'-'F')
ITE    GT           ; Next 2 instructions are conditional
ADDGT  R1, R0, #55  ; Convert 0xA -> 'A'
ADDLE  R1, R0, #48  ; Convert 0x0 -> '0'


IT     GT           ; IT block with only one conditional instruction
ADDGT  R1, R1, #1   ; Increment R1 conditionally


ITTEE  EQ           ; Next 4 instructions are conditional
MOVEQ  R0, R1       ; Conditional move
ADDEQ  R2, R2, #10  ; Conditional add
ANDNE  R3, R3, #1   ; Conditional AND
BNE.W  dloop        ; Branch instruction can only be used in the last
                    ; instruction of an IT block


IT     NE           ; Next instruction is conditional
ADD    R0, R0, R1   ; Syntax error: no condition code used in IT block
```

12.21.4 SysTick Calibration Value Register

The CALIB register indicates the SysTick calibration properties. See the register summary in Table 12-33 on page 189 for its attributes. The bit assignments are:

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

- **NOREF**

Reads as zero.

- **SKEW**

Reads as zero

- **TENMS**

Read as 0x0002904. The SysTick calibration value is fixed at 0x0002904 (10500), which allows the generation of a time base of 1 ms with SysTick clock at 10.5 MHz (84/8 = 10.5 MHz)

12.21.5 SysTick design hints and tips

The SysTick counter runs on the processor clock. If this clock signal is stopped for low power mode, the SysTick counter stops.

Ensure software uses aligned word accesses to access the SysTick registers.

17.5.6 RTC Calendar Alarm Register

Name: RTC_CALALR

Address: 0x400E1274

Access: Read-write

31	30	29	28	27	26	25	24
DATEEN	–	DATE					

23	22	21	20	19	18	17	16
MTHEN	–	–	MONTH				

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

This register can only be written if the WPEN bit is cleared in “RTC Write Protect Mode Register” on page 263.

- **MONTH: Month Alarm**

This field is the alarm field corresponding to the BCD-coded month counter.

- **MTHEN: Month Alarm Enable**

0 = The month-matching alarm is disabled.

1 = The month-matching alarm is enabled.

- **DATE: Date Alarm**

This field is the alarm field corresponding to the BCD-coded date counter.

- **DATEEN: Date Alarm Enable**

0 = The date-matching alarm is disabled.

1 = The date-matching alarm is enabled.

18.4.6.2 NRSTB Asynchronous Reset Pin

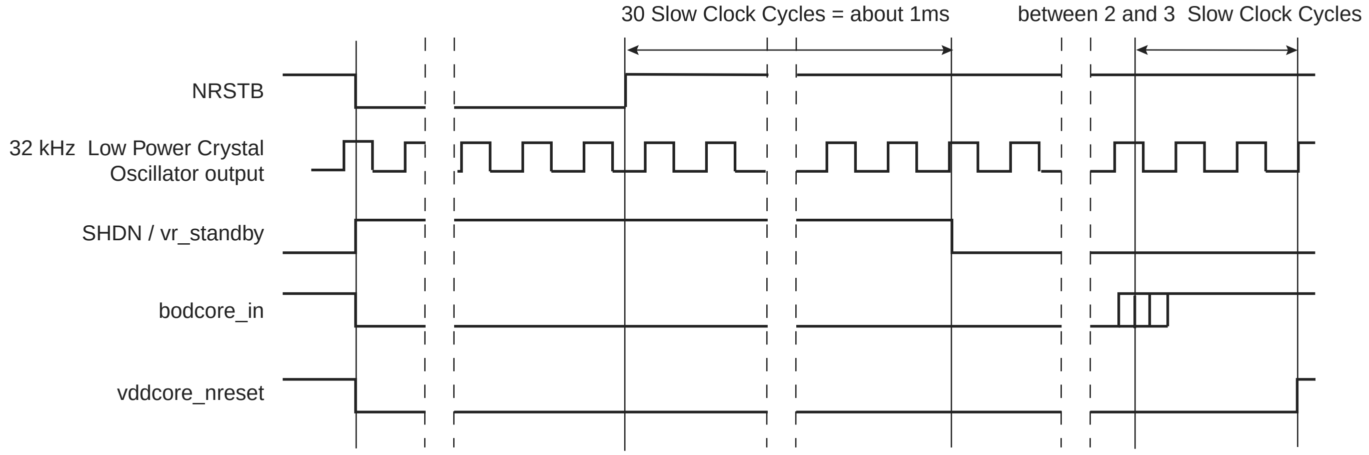
The NRSTB pin is an asynchronous reset input, which acts exactly like the zero-power power-on reset cell.

As soon as NRSTB is tied to GND, the supply controller is reset generating in turn, a reset of the whole system.

When NRSTB is released, the system can start as described in Section 18.4.6.1 ”Raising the Backup Power Supply”.

The NRSTB pin does not need to be driven during power-up phase to allow a reset of the system, it is done by the zero-power power-on cell.

Figure 18-6. NRSTB Reset


30 Slow Clock Cycles = about 1ms
between 2 and 3 Slow Clock Cycles
NRSTB
32 kHz Low Power Crystal Oscillator output
SHDN / vr_standby
bodcore_in
vddcore_nreset

Note: periph_nreset, ice_reset and proc_nreset are not shown, but are asserted low thanks to the vddcore_nreset signal controlling the Reset controller.

18.4.6.3 SHDN output pin

As shown in Figure 18-6, the SHDN pin acts like the vr_standby signal making it possible to use the SHDN pin to control external voltage regulator with shutdown capabilities.

18.4.7 Core Reset

The Supply Controller manages the vddcore_nreset signal to the Reset Controller, as described previously in Section 18.4.6 ”Backup Power Supply Reset”. The vddcore_nreset signal is normally asserted before shutting down the core power supply and released as soon as the core power supply is correctly regulated.

There are two additional sources which can be programmed to activate vddcore_nreset:

- a supply monitor detection
- a brownout detection

18.4.7.1 Supply Monitor Reset

The supply monitor is capable of generating a reset of the system. This can be enabled by setting the SMRSTEN bit in the Supply Controller Supply Monitor Mode Register (SUPC_SMMR).

If SMRSTEN is set and if a supply monitor detection occurs, the vddcore_nreset signal is immediately activated for a minimum of 1 slow clock cycle.

18.4.7.2 Brownout Detector Reset

The brownout detector provides the bodcore_in signal to the SUPC which indicates that the voltage regulation is operating as programmed. If this signal is lost for longer than 1 slow clock period while the voltage regulator is enabled, the Supply Controller can assert vddcore_nreset. This feature is enabled by writing the bit, BODRSTEN (Brownout Detector Reset Enable) to 1 in the Supply Controller Mode Register (SUPC_MR).

23.6 Bus Matrix (MATRIX) User Interface

Table 23-1. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	Master Configuration Register 0	MATRIX_MCFG0	Read-write	0x00000000
0x0004	Master Configuration Register 1	MATRIX_MCFG1	Read-write	0x00000000
0x0008	Master Configuration Register 2	MATRIX_MCFG2	Read-write	0x00000000
0x000C	Master Configuration Register 3	MATRIX_MCFG3	Read-write	0x00000000
0x0010	Master Configuration Register 4	MATRIX_MCFG4	Read-write	0x00000000
0x0014 - 0x003C	Reserved	–	–	–
0x0040	Slave Configuration Register 0	MATRIX_SCFG0	Read-write	0x00010010
0x0044	Slave Configuration Register 1	MATRIX_SCFG1	Read-write	0x00050010
0x0048	Slave Configuration Register 2	MATRIX_SCFG2	Read-write	0x00000010
0x004C	Slave Configuration Register 3	MATRIX_SCFG3	Read-write	0x00000010
0x0050	Slave Configuration Register 4	MATRIX_SCFG4	Read-write	0x00000010
0x0054	Slave Configuration Register 5	MATRIX_SCFG5	Read-write	0x00000010
0x0058	Slave Configuration Register 6	MATRIX_SCFG6	Read-write	0x00000010
0x005C	Slave Configuration Register 7	MATRIX_SCFG7	Read-write	0x00000010
0x0060	Slave Configuration Register 8	MATRIX_SCFG8	Read-write	0x00000010
0x0064	Slave Configuration Register 9	MATRIX_SCFG9	Read-write	0x00000010
0x0068 - 0x007C	Reserved	–	–	–
0x0080	Priority Register A for Slave 0	MATRIX_PRAS0	Read-write	0x00000000
0x0084	Reserved	–	–	–
0x0088	Priority Register A for Slave 1	MATRIX_PRAS1	Read-write	0x00000000
0x008C	Reserved	–	–	–
0x0090	Priority Register A for Slave 2	MATRIX_PRAS2	Read-write	0x00000000
0x0094	Reserved	–	–	–
0x0098	Priority Register A for Slave 3	MATRIX_PRAS3	Read-write	0x00000000
0x009C	Reserved	–	–	–
0x00A0	Priority Register A for Slave 4	MATRIX_PRAS4	Read-write	0x00000000
0x00A4	Reserved	–	–	–
0x00A8	Priority Register A for Slave 5	MATRIX_PRAS5	Read-write	0x00000000
0x00AC	Reserved	–	–	–
0x00B0	Priority Register A for Slave 6	MATRIX_PRAS6	Read-write	0x00000000
0x00B4	Reserved	–	–	–
0x00B8	Priority Register A for Slave 7	MATRIX_PRAS7	Read-write	0x00000000
0x00BC	Reserved	–	–	–
0x00C0	Priority Register A for Slave 8	MATRIX_PRAS8	Read-write	0x00000000

27.14.4 PMC Peripheral Clock Enable Register

Name: PMC_PCER

Address: 0x400E0410

Access: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24

23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16

15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8

7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	-	-

This register can only be written if the WPEN bit is cleared in the “PMC Write Protection Mode Register”.

- **PIDx: Peripheral Clock x Enable**

0: No effect.

1: Enables the corresponding peripheral clock.

Note: PID2 to PID31 refer to identifiers as defined in Section 11.1 ”Peripheral Identifiers”.

Note: Programming the control bits of the Peripheral ID that are not implemented has no effect on the behavior of the PMC.

27.14.20 PMC Write Protection Mode Register

Name: PMC_WPMR

Address: 0x400E04E4

Access: Read/Write

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

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x504D43 (“PMC” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x504D43 (“PMC” in ASCII).

See Section 27.13 ”Register Write Protection” for the list of registers that can be write-protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x504D43	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

29.7.6 PIO Controller Output Status Register

Name: PIO_OSR

Address: 0x400E0C18 (PIOA), 0x400E0E18 (PIOB), 0x400E1018 (PIOC)

Access: Read-only

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-P31: Output Status**

0 = The I/O line is a pure input.

1 = The I/O line is enabled in output.

- **DCDIC: Data Carrier Detect Input Change Interrupt Mask**

- **CTSIC: Clear to Send Input Change Interrupt Mask**

- **MANE: Manchester Error Interrupt Mask**

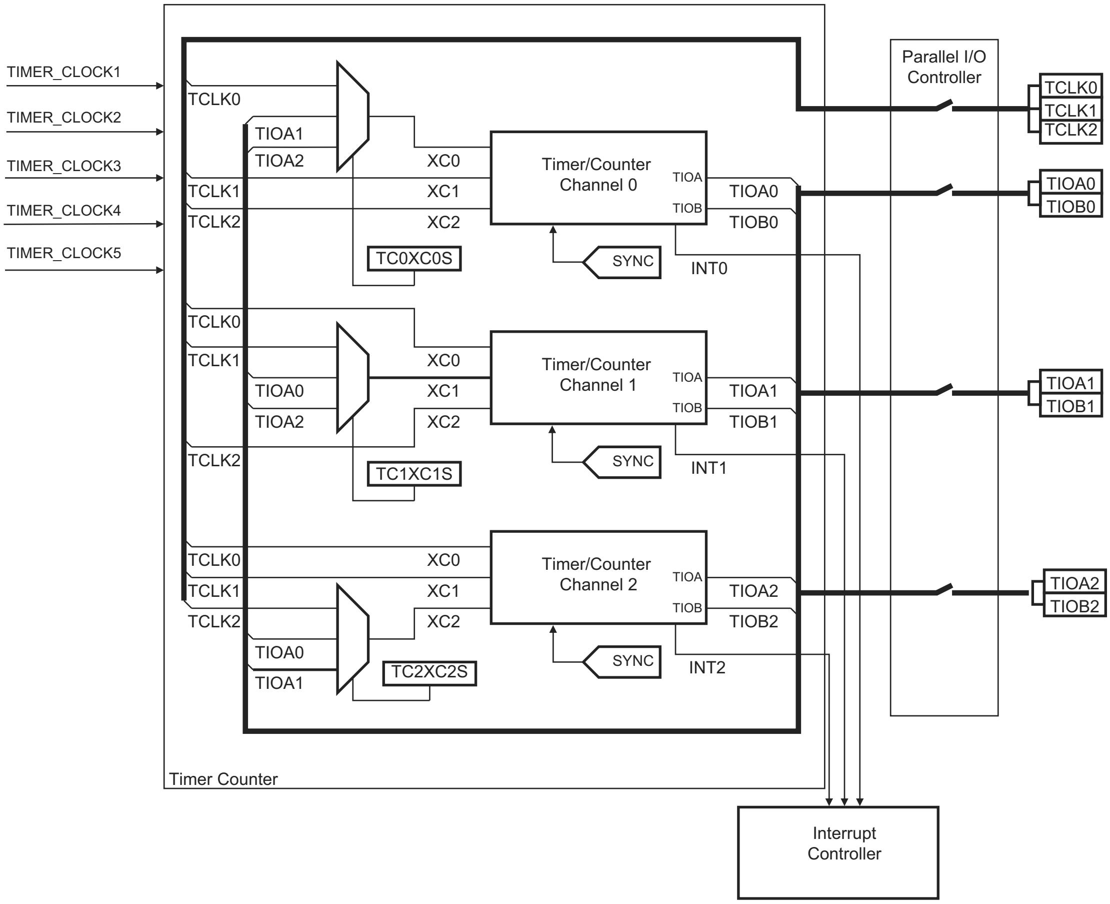
35.3 Block Diagram

Table 35-1. Timer Counter Clock Assignment

Name	Definition
TIMER_CLOCK1	MCK/2
TIMER_CLOCK2	MCK/8
TIMER_CLOCK3	MCK/32
TIMER_CLOCK4	MCK/128
TIMER_CLOCK5	SLCK

Note: 1. When SLCK is selected for Peripheral Clock (CSS = 0 in PMC Master Clock Register), SLCK input is equivalent to Peripheral Clock.

Figure 35-1. Timer Counter Block Diagram



i. If LLI(n) is the last descriptor, then LLI(n).DSCR points to 0 else LLI(n) points to the start address of LLI(n+1).

j. Program DMAC_CTRLBx for channel register x with 0. Its content is updated with the LLI fetch operation.

k. Program DMAC_DSCRx for channel register x with the address of the first descriptor LLI(0).

l. Enable Channel x writing one to DMAC_CHER[x]. The DMA is ready and waiting for request.

7. Poll CBTC[x] bit in the DMAC_EBCISR Register.
8. If a new list of buffers shall be transferred, repeat step 6. Check and handle HSMCI errors.
9. Poll FIFOEMPTY field in the HSMCI_SR.
10. Send The STOP_TRANSMISSION command writing HSMCI_ARG then HSMCI_CMDR.
11. Wait for XFRDONE in HSMCI_SR register.

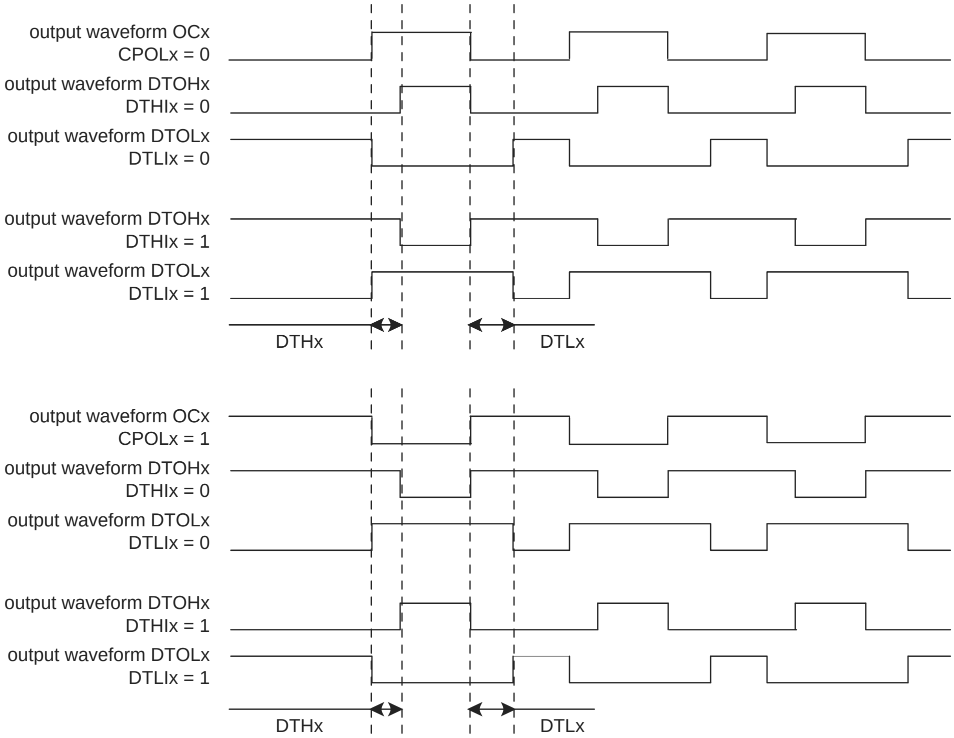
37.6.2.3 Dead-Time Generator

The dead-time generator uses the comparator output OCx to provide the two complementary outputs DTOHx and DTOLx, which allows the PWM macrocell to drive external power control switches safely. When the dead-time generator is enabled by setting the bit DTE to 1 or 0 in the “PWM Channel Mode Register” (PWM_CMRx), dead-times (also called dead-bands or non-overlapping times) are inserted between the edges of the two complementary outputs DTOHx and DTOLx. Note that enabling or disabling the dead-time generator is allowed only if the channel is disabled.

The dead-time is adjustable by the “PWM Channel Dead Time Register” (PWM_DTx). Both outputs of the dead-time generator can be adjusted separately by DTH and DTL. The dead-time values can be updated synchronously to the PWM period by using the “PWM Channel Dead Time Update Register” (PWM_DTUPDx).

The dead-time is based on a specific counter which uses the same selected clock that feeds the channel counter of the comparator. Depending on the edge and the configuration of the dead-time, DTOHx and DTOLx are delayed until the counter has reached the value defined by DTH or DTL. An inverted configuration bit (DTHI and DTLI bit in the PWM_CMRx register) is provided for each output to invert the dead-time outputs. The following figure shows the waveform of the dead-time generator.

Figure 37-6. Complementary Output Waveforms



Method 3: Automatic write of duty-cycle values and automatic trigger of the update

In this mode, the update of the duty cycle values is made automatically by the Peripheral DMA Controller (PDC). The update of the period value, the dead-time values and the update period value must be done by writing in their respective update registers with the CPU (respectively PWM_CPRDUPDx, PWM_DTUPDx and PWM_SCUPUPD).

To trigger the update of the period value and the dead-time values, the user must use the bit UPDULOCK which allows to update synchronously (at the same PWM period) the synchronous channels:

- If the bit UPDULOCK is set to 1, the update is done at the next PWM period of the synchronous channels.
- If the UPDULOCK bit is not set to 1, the update is locked and cannot be performed.

After writing the UPDULOCK bit to 1, it is held at this value until the update occurs, then it is read 0.

The update of the duty-cycle values and the update period value is triggered automatically after an update period.

To configure the automatic update, the user must define a value for the Update Period by the field UPR in the "PWM Sync Channels Update Period Register" (PWM_SCUP). The PWM controller waits UPR+1 periods of synchronous channels before updating automatically the duty values and the update period value.

Using the PDC removes processor overhead by reducing its intervention during the transfer. This significantly reduces the number of clock cycles required for a data transfer, which improves microcontroller performance.

The PDC must write the duty-cycle values in the synchronous channels index order. For example if the channels 0, 1 and 3 are synchronous channels, the PDC must write the duty-cycle of the channel 0 first, then the duty-cycle of the channel 1, and finally the duty-cycle of the channel 3.

The status of the PDC transfer is reported in the "PWM Interrupt Status Register 2" (PWM_ISR2) by the following flags:

- WRDY: this flag is set to 1 when the PWM Controller is ready to receive new duty-cycle values and a new update period value. It is reset to 0 when the PWM_ISR2 register is read. The user can choose to synchronize the WRDY flag and the PDC transfer request with a comparison match (see Section 37.6.3 "PWM Comparison Units"), by the fields PTRM and PTRCS in the PWM_SCM register.
- ENDTX: this flag is set to 1 when a PDC transfer is completed
- TXBUFE: this flag is set to 1 when the PDC buffer is empty (no pending PDC transfers)
- UNRE: this flag is set to 1 when the update period defined by the UPR field has elapsed while the whole data has not been written by the PDC. It is reset to 0 when the PWM_ISR2 register is read.

Depending on the interrupt mask in the PWM_IMR2 register, an interrupt can be generated by these flags.

37.7.6 PWM Interrupt Disable Register 1

Name: PWM_IDR1

Address: 0x4008C014

Access: Write-only

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

23	22	21	20	19	18	17	16
–	–	–	–	FCHID3	FCHID2	FCHID1	FCHID0

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Counter Event on Channel x Interrupt Disable**

- **FCHIDx: Fault Protection Trigger on Channel x Interrupt Disable**

38.7.10 UDPHS Features Register

Name: UDPHS_IPFEATURES

Address: 0x400A40F8

Access: Read-only

31	30	29	28	27	26	25	24
ISO_EPT_15	ISO_EPT_14	ISO_EPT_13	ISO_EPT_12	ISO_EPT_11	ISO_EPT_10	ISO_EPT_9	ISO_EPT_8

23	22	21	20	19	18	17	16
ISO_EPT_7	ISO_EPT_6	ISO_EPT_5	ISO_EPT_4	ISO_EPT_3	ISO_EPT_2	ISO_EPT_1	DATAB16_8

15	14	13	12	11	10	9	8
BW_DPRAM	FIFO_MAX_SIZE			DMA_FIFO_WORD_DEPTH			

7	6	5	4	3	2	1	0
DMA_B_SIZ	DMA_CHANNEL_NBR			EPT_NBR_MAX			

- **EPT_NBR_MAX: Max Number of Endpoints**

Give the max number of endpoints.

0 = if 16 endpoints are hardware implemented.

1 = if 1 endpoint is hardware implemented.

2 = if 2 endpoints are hardware implemented.

...

15 = if 15 endpoints are hardware implemented.

- **DMA_CHANNEL_NBR: Number of DMA Channels**

Give the number of DMA channels.

1 = if 1 DMA channel is hardware implemented.

2 = if 2 DMA channels are hardware implemented.

...

7 = if 7 DMA channels are hardware implemented.

- **DMA_B_SIZ: DMA Buffer Size**

0 = if the DMA Buffer size is 16 bits.

1 = if the DMA Buffer size is 24 bits.

- **DMA_FIFO_WORD_DEPTH: DMA FIFO Depth in Words**

0 = if FIFO is 16 words deep.

1 = if FIFO is 1 word deep.

2 = if FIFO is 2 words deep.

...

15 = if FIFO is 15 words deep.

38.7.17 UDPHS Endpoint Status Register

Name: UDPHS_EPTSTAx [x=0..]

Address: 0x400A411C [0], 0x400A413C [1], 0x400A415C [2], 0x400A417C [3], 0x400A419C [4], 0x400A41BC [5], 0x400A41DC [6]

Access: Read-only

31	30	29	28	27	26	25	24
SHRT_PCKT	BYTE_COUNT						

23	22	21	20	19	18	17	16
BYTE_COUNT				BUSY_BANK_STA		CURRENT_BANK/ CONTROL_DIR	

15	14	13	12	11	10	9	8
NAK_OUT	NAK_IN/ ERR_FLUSH	STALL_SNT/ ERR_CRISO/ ERR_NBTRA	RX_SETUP/ ERR_FL_ISO	TX_PK_RDY/ ERR_TRANS	TX_COMPLT	RX_BK_RDY/ KILL_BANK	ERR_OVFLW

7	6	5	4	3	2	1	0
TOGGLESQ_STA		FRCESTALL	–	–	–	–	–

- **FRCESTALL: Stall Handshake Request**

0 = no effect.

1= If set a STALL answer will be done to the host for the next handshake.

This bit is reset by hardware upon received SETUP.

- **TOGGLESQ_STA: Toggle Sequencing**

Toggle Sequencing:

– **IN endpoint:** it indicates the PID Data Toggle that will be used for the next packet sent. This is not relative to the current bank.

– CONTROL and OUT endpoint:

These bits are set by hardware to indicate the PID data of the current bank:

Value	Name	Description
0	DATA0	DATA0
1	DATA1	DATA1
2	DATA2	Data2 (only for High Bandwidth Isochronous Endpoint)
3	MDATA	MData (only for High Bandwidth Isochronous Endpoint)

Notes: 1. In OUT transfer, the Toggle information is meaningful only when the current bank is busy (Received OUT Data = 1).

2. These bits are updated for OUT transfer:
- a new data has been written into the current bank.
- the user has just cleared the Received OUT Data bit to switch to the next bank.

3. For High Bandwidth Isochronous Out endpoint, it is recommended to check the UDPHS_EPTSTAx/ERR_TRANS bit to know if the toggle sequencing is correct or not.

4. This field is reset to DATA1 by the UDPHS_EPTCLRSTAx register TOGGLESQ bit, and by UDPHS_EPTCTLDISx (disable endpoint).

40.6.1 ADC12B Control Register

Name: ADC12B_CR

Address: 0x400A8000

Access: Write-only

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
–	–	–	–	–	–	START	SWRST

- **SWRST: Software Reset**

0 = No effect.

1 = Resets the ADC12B simulating a hardware reset.

- **START: Start Conversion**

0 = No effect.

1 = Begins analog-to-digital conversion.