Microchip Technology - ATSAM3U1CB-CU Datasheet

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	64KB (64K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	20K 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/atsam3u1cb-cu

4.2.4 100-ball TFBGA Pinout

Table 4-4. 100-ball TFBGA Pinout (SAM3U4C / SAM3U2C / SAM3U1C Devices)

A1	VBG	C6	PB22	F1	PB1	H6	PA15/PGMD7
A2	XIN	C7	TMS/SWDIO	F2	PB12	H7	PA18/PGMD10
A3	XOUT	C8	NRSTB	F3	VDDIO	H8	PA24
A4	PB17	C9	JTAGSEL	F4	PA31	H9	PA1/PGMRDY
A5	PB21	C10	VDDBU	F5	VDDIO	H10	PA2/PGMNOE
A6	PB23	D1	DFSDM	F6	GND	J1	PB6
A7	TCK/SWCLK	D2	DHSDM	F7	PB16	J2	PB8
A8	VDDIN	D3	VDDPLL	F8	PA6/PGMM2	J3	ADVREF
A9	VDDOUT	D4	VDDCORE	F9	VDDCORE	J4	PA30
A10	XIN32	D5	PB20	F10	PA7/PGMM3	J5	PB3
B1	VDDCORE	D6	ERASE	G1	PB11	J6	PA16/PGMD8
B2	GNDUTMI	D7	TST	G2	PB2	J7	PA19/PGMD11
B3	VDDUTMI	D8	FWUP	G3	PB0	J8	PA21/PGMD13
B4	PB10	D9	PA11/PGMD3	G4	PB13	J9	PA26
B5	PB18	D10	PA12/PGMD4	G5	VDDCORE	J10	PA0/PGMNCMD
B6	PB24	E1	PA29	G6	GND	K1	PB7
B7	NRST	E2	GND	G7	PB15	K2	VDDANA
B8	TDO/TRACESWO	E3	PA28	G8	PA3/PGMNVALID	K3	GNDANA
B9	TDI	E4	PB9	G9	PA5/PGMM1	K4	AD12BVREF
B10	XOUT32	E5	GNDBU	G10	PA4/PGMM0	K5	PB4
C1	DFSDP	E6	VDDIO	H1	VDDCORE	K6	PA14/PGMD6
C2	DHSDP	E7	VDDCORE	H2	PB5	K7	PA17/PGMD9
C3	GNDPLL	E8	PA10/PGMD2	H3	PA27	K8	PA20/PGMD12
C4	PB14	E9	PA9/PGMD1	H4	PA22/PGMD14	K9	PA23/PGMD15
C5	PB19	E10	PA8/PGMD0	H5	PA13/PGMD5	K10	PA25

12.11.9 CLREX

Clear Exclusive.

12.11.9.1 Syntax

```
CLREX{cond}
```

where:

cond	is an optional condition code, see “Conditional execution” on page 87.

12.11.9.2 Operation

Use CLREX to make the next STREX, STREXB, or STREXH instruction write 1 to its destination register and fail to perform the store. It is useful in exception handler code to force the failure of the store exclusive if the exception occurs between a load exclusive instruction and the matching store exclusive instruction in a synchronization operation.

See “Synchronization primitives” on page 65 for more information.

12.11.9.3 Condition flags

These instructions do not change the flags.

12.11.9.4 Examples

```
CLREX
```

12.15 Bitfield instructions

Table 12-22 shows the instructions that operate on adjacent sets of bits in registers or bitfields:

Table 12-22. Packing and unpacking instructions

Mnemonic	Brief description	See
BFC	Bit Field Clear	“BFC and BFI” on page 125
BFI	Bit Field Insert	“BFC and BFI” on page 125
SBFX	Signed Bit Field Extract	“SBFX and UBFX” on page 126
SXTB	Sign extend a byte	“SXT and UXT” on page 127
SXTH	Sign extend a halfword	“SXT and UXT” on page 127
UBFX	Unsigned Bit Field Extract	“SBFX and UBFX” on page 126
UXTB	Zero extend a byte	“SXT and UXT” on page 127
UXTH	Zero extend a halfword	“SXT and UXT” on page 127

- **USERSETMPEND**

Enables unprivileged software access to the STIR, see “Software Trigger Interrupt Register” on page 158:

0 = disable

1 = enable.

- **NONEBASETHRDENA**

Indicates how the processor enters Thread mode:

0 = processor can enter Thread mode only when no exception is active.

1 = processor can enter Thread mode from any level under the control of an EXC_RETURN value, see “Exception return” on page 72.

12.20.9.1 System Handler Priority Register 1

The bit assignments are:

31	30	29	28	27	26	25	24
PRI_7: Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRI_6							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PRI_5							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRI_4							

- **PRI_7**

Reserved

- **PRI_6**

Priority of system handler 6, usage fault

- **PRI_5**

Priority of system handler 5, bus fault

- **PRI_4**

Priority of system handler 4, memory management fault

12.22.2 MPU Control Register

The MPU CTRL register:

- enables the MPU
- enables the default memory map background region
- enables use of the MPU when in the hard fault, *Non-maskable Interrupt* (NMI), and FAULTMASK escalated handlers.

See the register summary in Table 12-35 on page 195 for the MPU CTRL attributes. The bit assignments are:

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

- **PRIVDEFENA**

Enables privileged software access to the default memory map:

0 = If the MPU is enabled, disables use of the default memory map. Any memory access to a location not covered by any enabled region causes a fault.

1 = If the MPU is enabled, enables use of the default memory map as a background region for privileged software accesses.

When enabled, the background region acts as if it is region number -1. Any region that is defined and enabled has priority over this default map.

If the MPU is disabled, the processor ignores this bit.

- **HFNMIENA**

Enables the operation of MPU during hard fault, NMI, and FAULTMASK handlers.

When the MPU is enabled:

0 = MPU is disabled during hard fault, NMI, and FAULTMASK handlers, regardless of the value of the ENABLE bit

1 = the MPU is enabled during hard fault, NMI, and FAULTMASK handlers.

When the MPU is disabled, if this bit is set to 1 the behavior is Unpredictable.

- **ENABLE**

Enables the MPU:

0 = MPU disabled

1 = MPU enabled.

When ENABLE and PRIVDEFENA are both set to 1:

For privileged accesses, the *default memory map* is as described in “Memory model” on page 58. Any access by privileged software that does not address an enabled memory region behaves as defined by the default memory map.

Any access by unprivileged software that does not address an enabled memory region causes a memory management fault.

XN and Strongly-ordered rules always apply to the System Control Space regardless of the value of the ENABLE bit.

Doubleword-aligned

A data item having a memory address that is divisible by eight.

Endianness

Byte ordering. The scheme that determines the order that successive bytes of a data word are stored in memory. An aspect of the system's memory mapping.

See also "Little-endian (LE)"

Exception

An event that interrupts program execution. When an exception occurs, the processor suspends the normal program flow and starts execution at the address indicated by the corresponding exception vector. The indicated address contains the first instruction of the handler for the exception.

An exception can be an interrupt request, a fault, or a software-generated system exception. Faults include attempting an invalid memory access, attempting to execute an instruction in an invalid processor state, and attempting to execute an undefined instruction.

Exception service routine

See "Interrupt handler".

Exception vector

See "Interrupt vector".

Flat address mapping

A system of organizing memory in which each physical address in the memory space is the same as the corresponding virtual address.

Halfword

A 16-bit data item.

Illegal instruction

An instruction that is architecturally Undefined.

Implementation-defined

The behavior is not architecturally defined, but is defined and documented by individual implementations.

Implementation-specific

The behavior is not architecturally defined, and does not have to be documented by individual implementations. Used when there are a number of implementation options available and the option chosen does not affect software compatibility.

Index register

In some load and store instruction descriptions, the value of this register is used as an offset to be added to or subtracted from the base register value to form the address that is sent to memory. Some addressing modes optionally enable the index register value to be shifted prior to the addition or subtraction.

See also "Base register"

Instruction cycle count

The number of cycles that an instruction occupies the Execute stage of the pipeline.

Interrupt handler

A program that control of the processor is passed to when an interrupt occurs.

Interrupt vector

One of a number of fixed addresses in low memory, or in high memory if high vectors are configured, that contains the first instruction of the corresponding interrupt handler.

Little-endian (LE)

Byte ordering scheme in which bytes of increasing significance in a data word are stored at increasing addresses in memory.

See also “Condition field”, “Endianness”.

Little-endian memory

Memory in which:

a byte or halfword at a word-aligned address is the least significant byte or halfword within the word at that address

a byte at a halfword-aligned address is the least significant byte within the halfword at that address.

Load/store architecture

A processor architecture where data-processing operations only operate on register contents, not directly on memory contents.

Memory Protection Unit (MPU)

Hardware that controls access permissions to blocks of memory. An MPU does not perform any address translation.

Prefetching

In pipelined processors, the process of fetching instructions from memory to fill up the pipeline before the preceding instructions have finished executing. Prefetching an instruction does not mean that the instruction has to be executed.

Read

Reads are defined as memory operations that have the semantics of a load. Reads include the Thumb instructions LDM, LDR, LDRSH, LDRH, LDRSB, LDRB, and POP.

Region

A partition of memory space.

Reserved

A field in a control register or instruction format is reserved if the field is to be defined by the implementation, or produces Unpredictable results if the contents of the field are not zero. These fields are reserved for use in future extensions of the architecture or are implementation-specific. All reserved bits not used by the implementation must be written as 0 and read as 0.

Should Be One (SBO)

Write as 1, or all 1s for bit fields, by software. Writing as 0 produces Unpredictable results.

Should Be Zero (SBZ)

Write as 0, or all 0s for bit fields, by software. Writing as 1 produces Unpredictable results.

Should Be Zero or Preserved (SBZP)

Write as 0, or all 0s for bit fields, by software, or preserved by writing the same value back that has been previously read from the same field on the same processor.

Thread-safe

In a multi-tasking environment, thread-safe functions use safeguard mechanisms when accessing shared resources, to ensure correct operation without the risk of shared access conflicts.

Thumb instruction

One or two halfwords that specify an operation for a processor to perform. Thumb instructions must be halfword-aligned.

The PMC_MCKR must not be programmed in a single write operation. The preferred programming sequence for the PMC_MCKR is as follows:

If a new value for CSS field corresponds to PLL Clock,

a. Program PMC_MCKR.PRES field
b. Wait for PMC_SR.MCKRDY bit to be set
c. Program PMC_MCKR.CSS field
d. Wait for PMC_SR.MCKRDY bit to be set

If a new value for CSS field corresponds to Main Clock or Slow Clock,

a. Program PMC_MCKR.CSS field
b. Wait for PMC_SR.MCKRDY bit to be set
c. Program PMC_MCKR.PRES field
d. Wait for PMC_SR.MCKRDY bit to be set

If at some stage one of the following parameters, CSS or PRES, is modified, the MCKRDY bit will go low to indicate that the Master Clock and the Processor Clock are not ready yet. The user must wait for MCKRDY bit to be set again before using the Master and Processor Clocks.

Note: IF PLLx clock was selected as the Master Clock and the user decides to modify it by writing in CKGR_PLLR, the MCKRDY flag will go low while PLL is unlocked. Once PLL is locked again, LOCK goes high and MCKRDY is set. While PLL is unlocked, the Master Clock selection is automatically changed to Slow Clock. For further information, see Section 27.12.2 “Clock Switching Waveforms” on page 460.

Code Example:

```
write_register(PMC_MCKR,0x00000001)
wait (MCKRDY=1)
write_register(PMC_MCKR,0x00000011)
wait (MCKRDY=1)
```

The Master Clock is main clock divided by 16.

The Processor Clock is the Master Clock.

5. Selection of Programmable Clocks

Programmable clocks are controlled via registers; PMC_SCER, PMC_SCDR and PMC_SCSR.

Programmable clocks can be enabled and/or disabled via the PMC_SCER and PMC_SCDR. Three programmable clocks can be enabled or disabled. The PMC_SCSR provides a clear indication as to which programmable clock is enabled. By default all programmable clocks are disabled.

PMC_PCKx registers are used to configure programmable clocks.

The CSS field is used to select the programmable clock divider source. Four clock options are available: main clock, slow clock, PLLACK and UPLLCK. By default, the clock source selected is main clock.

The PRES field is used to control the programmable clock prescaler. It is possible to choose between different values (1, 2, 4, 8, 16, 32, 64). Programmable clock output is prescaler input divided by PRES parameter. By default, the PRES parameter is set to 0 which means that master clock is equal to slow clock.

Once the PMC_PCKx register has been programmed, The corresponding programmable clock must be enabled and the user is constrained to wait for the PCKRDYx bit to be set in the PMC_SR. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to PCKRDYx has been enabled in the PMC_IER. All parameters in PMC_PCKx can be programmed in a single write operation.

Figure 27-6. Change PLLA Programming

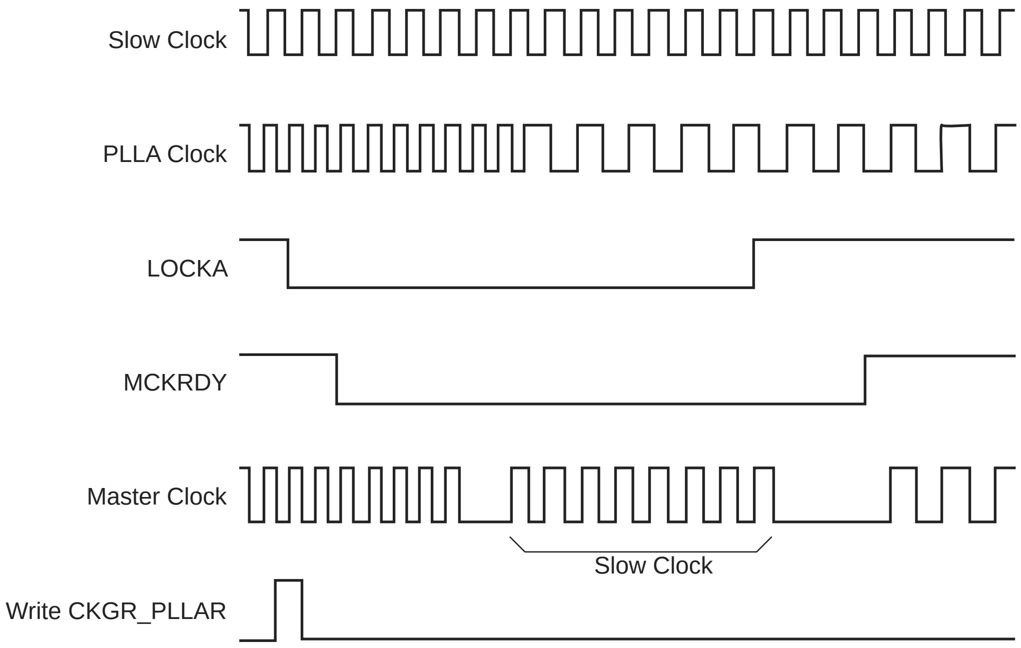
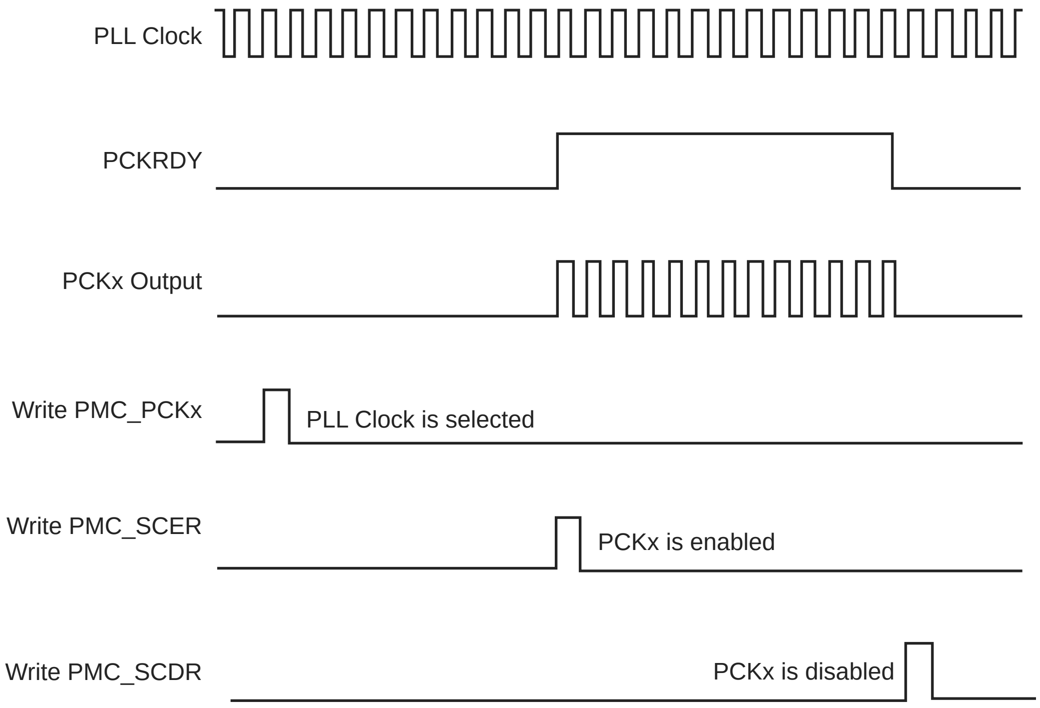


Figure 27-7. Programmable Clock Output Programming



29.5.2 I/O Line or Peripheral Function Selection

When a pin is multiplexed with one or two peripheral functions, the selection is controlled with the registers PIO_PER (PIO Enable Register) and PIO_PDR (PIO Disable Register). The register PIO_PSR (PIO Status Register) is the result of the set and clear registers and indicates whether the pin is controlled by the corresponding peripheral or by the PIO Controller. A value of 0 indicates that the pin is controlled by the corresponding on-chip peripheral selected in the PIO_ABSR (AB Select Register). A value of 1 indicates the pin is controlled by the PIO controller.

If a pin is used as a general purpose I/O line (not multiplexed with an on-chip peripheral), PIO_PER and PIO_PDR have no effect and PIO_PSR returns 1 for the corresponding bit.

After reset, most generally, the I/O lines are controlled by the PIO controller, i.e. PIO_PSR resets at 1. However, in some events, it is important that PIO lines are controlled by the peripheral (as in the case of memory chip select lines that must be driven inactive after reset or for address lines that must be driven low for booting out of an external memory). Thus, the reset value of PIO_PSR is defined at the product level, depending on the multiplexing of the device.

29.5.3 Peripheral A or B Selection

The PIO Controller provides multiplexing of up to two peripheral functions on a single pin. The selection is performed by writing PIO_ABSR (AB Select Register). For each pin, the corresponding bit at level 0 means peripheral A is selected whereas the corresponding bit at level 1 indicates that peripheral B is selected.

Note that multiplexing of peripheral lines A and B only affects the output line. The peripheral input lines are always connected to the pin input.

After reset, PIO_ABSR is 0, thus indicating that all the PIO lines are configured on peripheral A. However, peripheral A generally does not drive the pin as the PIO Controller resets in I/O line mode.

Writing in PIO_ABSR manages the multiplexing regardless of the configuration of the pin. However, assignment of a pin to a peripheral function requires a write in the peripheral selection register (PIO_ABSR) in addition to a write in PIO_PDR.

29.5.4 Output Control

When the I/0 line is assigned to a peripheral function, i.e. the corresponding bit in PIO_PSR is at 0, the drive of the I/O line is controlled by the peripheral. Peripheral A or B depending on the value in PIO_ABSR (AB Select Register) determines whether the pin is driven or not.

When the I/O line is controlled by the PIO controller, the pin can be configured to be driven. This is done by writing PIO_OER (Output Enable Register) and PIO_ODR (Output Disable Register). The results of these write operations are detected in PIO_OSR (Output Status Register). When a bit in this register is at 0, the corresponding I/O line is used as an input only. When the bit is at 1, the corresponding I/O line is driven by the PIO controller.

The level driven on an I/O line can be determined by writing in PIO_SODR (Set Output Data Register) and PIO_CODR (Clear Output Data Register). These write operations respectively set and clear PIO_ODSR (Output Data Status Register), which represents the data driven on the I/O lines. Writing in PIO_OER and PIO_ODR manages PIO_OSR whether the pin is configured to be controlled by the PIO controller or assigned to a peripheral function. This enables configuration of the I/O line prior to setting it to be managed by the PIO Controller.

Similarly, writing in PIO_SODR and PIO_CODR effects PIO_ODSR. This is important as it defines the first level driven on the I/O line.

29.7.13 PIO Controller Pin Data Status Register

Name: PIO_PDSR

Address: 0x400E0C3C (PIOA), 0x400E0E3C (PIOB), 0x400E103C (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 Data Status**

0 = The I/O line is at level 0.

1 = The I/O line is at level 1.

29.7.21 PIO Pull Up Disable Register

Name: PIO_PUDR

Address: 0x400E0C60 (PIOA), 0x400E0E60 (PIOB), 0x400E1060 (PIOC)

Access: Write-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

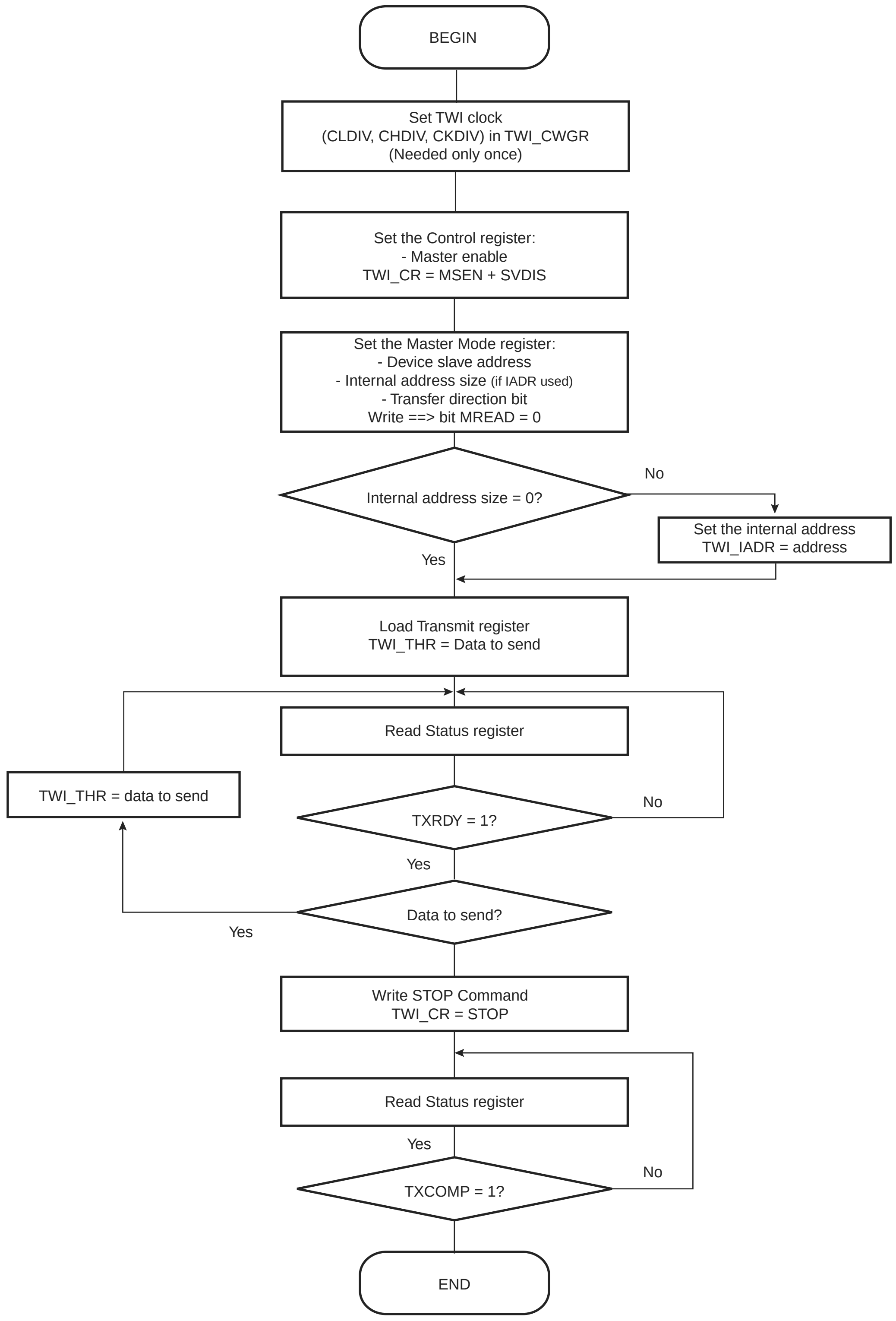
This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register”.

- **P0-P31: Pull Up Disable.**

0 = No effect.

1 = Disables the pull up resistor on the I/O line.

Figure 32-17. TWI Write Operation with Multiple Data Bytes with or without Internal Address



32.11.3 TWI Slave Mode Register

Name: TWI_SMR

Address: 0x40084008 (0), 0x40088008 (1)

Access: Read-write

Reset: 0x00000000

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

23	22	21	20	19	18	17	16
–	SADR						

15	14	13	12	11	10	9	8
–	–	–	–	–	–		

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **SADR: Slave Address**

The slave device address is used in Slave mode in order to be accessed by master devices in read or write mode.

SADR must be programmed before enabling the Slave mode or after a general call. Writes at other times have no effect.

33.3 Block Diagram

Figure 33-1. UART Functional Block Diagram

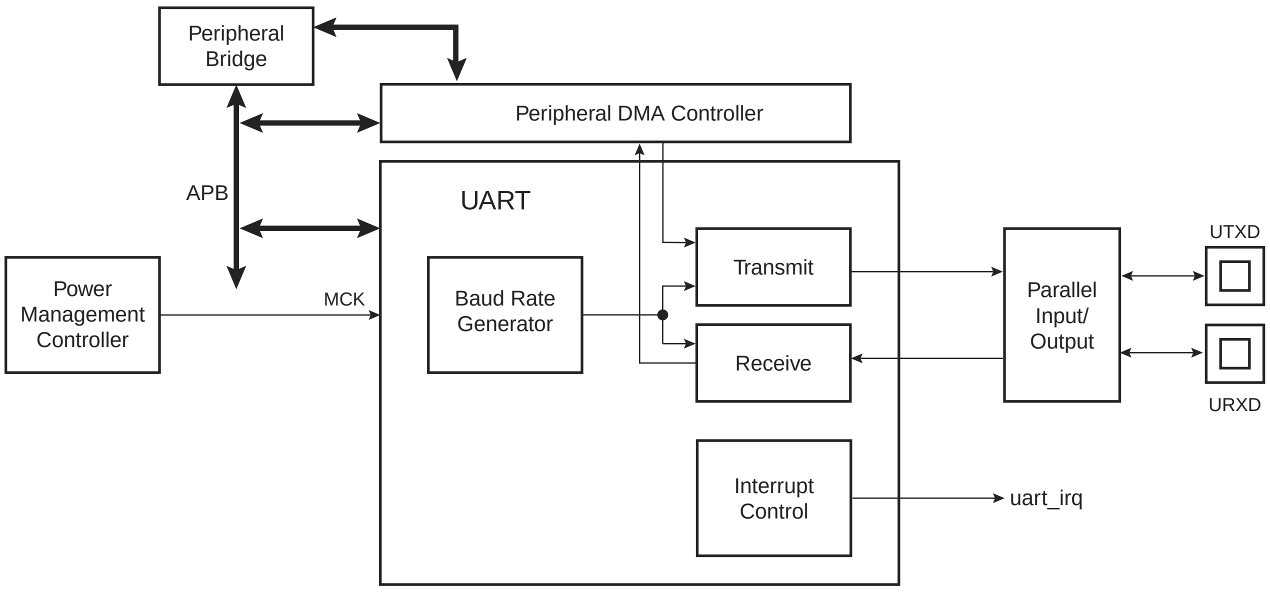


Table 33-1. UART Pin Description

Pin Name	Description	Type
URXD	UART Receive Data	Input
UTXD	UART Transmit Data	Output

33.4 Product Dependencies

33.4.1 I/O Lines

The UART pins are multiplexed with PIO lines. The programmer must first configure the corresponding PIO Controller to enable I/O line operations of the UART.

Table 33-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
UART	URXD	PA11	A
UART	UTXD	PA12	A

33.4.2 Power Management

The UART clock is controllable through the Power Management Controller. In this case, the programmer must first configure the PMC to enable the UART clock. Usually, the peripheral identifier used for this purpose is 1.

33.4.3 Interrupt Source

The UART interrupt line is connected to one of the interrupt sources of the Nested Vectored Interrupt Controller (NVIC). Interrupt handling requires programming of the NVIC before configuring the UART.

34.7.1 Baud Rate Generator

The Baud Rate Generator provides the bit period clock named the Baud Rate Clock to both the receiver and the transmitter.

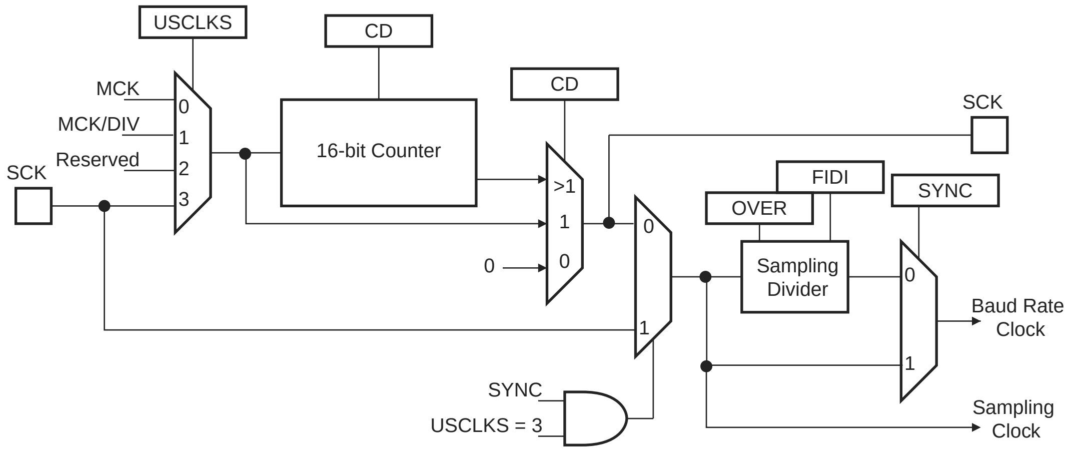
The Baud Rate Generator clock source can be selected by setting the USCLKS field in the Mode Register (US_MR) between:

- the Master Clock MCK
- a division of the Master Clock, the divider being product dependent, but generally set to 8
- the external clock, available on the SCK pin

The Baud Rate Generator is based upon a 16-bit divider, which is programmed with the CD field of the Baud Rate Generator Register (US_BRGR). If CD is programmed to 0, the Baud Rate Generator does not generate any clock. If CD is programmed to 1, the divider is bypassed and becomes inactive.

If the external SCK clock is selected, the duration of the low and high levels of the signal provided on the SCK pin must be longer than a Master Clock (MCK) period. The frequency of the signal provided on SCK must be at least 3 times lower than MCK in USART mode, or 6 in SPI mode.

Figure 34-3. Baud Rate Generator



34.7.1.1 Baud Rate in Asynchronous Mode

If the USART is programmed to operate in asynchronous mode, the selected clock is first divided by CD, which is field programmed in the Baud Rate Generator Register (US_BRGR). The resulting clock is provided to the receiver as a sampling clock and then divided by 16 or 8, depending on the programming of the OVER bit in US_MR.

If OVER is set to 1, the receiver sampling is 8 times higher than the baud rate clock. If OVER is cleared, the sampling is performed at 16 times the baud rate clock.

The following formula performs the calculation of the Baud Rate.

$$Baudrate = \frac{SelectedClock}{(8(2 - Over)CD)}$$

This gives a maximum baud rate of MCK divided by 8, assuming that MCK is the highest possible clock and that OVER is programmed to 1.

34.8.1 USART Control Register

Name:	US_CR
Address:	0x40090000 (0), 0x40094000 (1), 0x40098000 (2), 0x4009C000 (3)
Access:	Write-only

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

23	22	21	20	19	18	17	16
–	–	–	–	RTSDIS/RCS	RTSEN/FCS	DTRDIS	DTREN

15	14	13	12	11	10	9	8
RETTO	RSTNACK	RSTIT	SENDA	STTTO	STPBRK	STTBRK	RSTSTA

7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

- **RSTRX: Reset Receiver**

0: No effect.

1: Resets the receiver.

- **RSTTX: Reset Transmitter**

0: No effect.

1: Resets the transmitter.

- **RXEN: Receiver Enable**

0: No effect.

1: Enables the receiver, if RXDIS is 0.

- **RXDIS: Receiver Disable**

0: No effect.

1: Disables the receiver.

- **TXEN: Transmitter Enable**

0: No effect.

1: Enables the transmitter if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0: No effect.

1: Disables the transmitter.

- **RSTSTA: Reset Status Bits**

0: No effect.

1: Resets the status bits PARE, FRAME, OVRE, MANERR, **UNRE** and RXBRK in US_CSR.

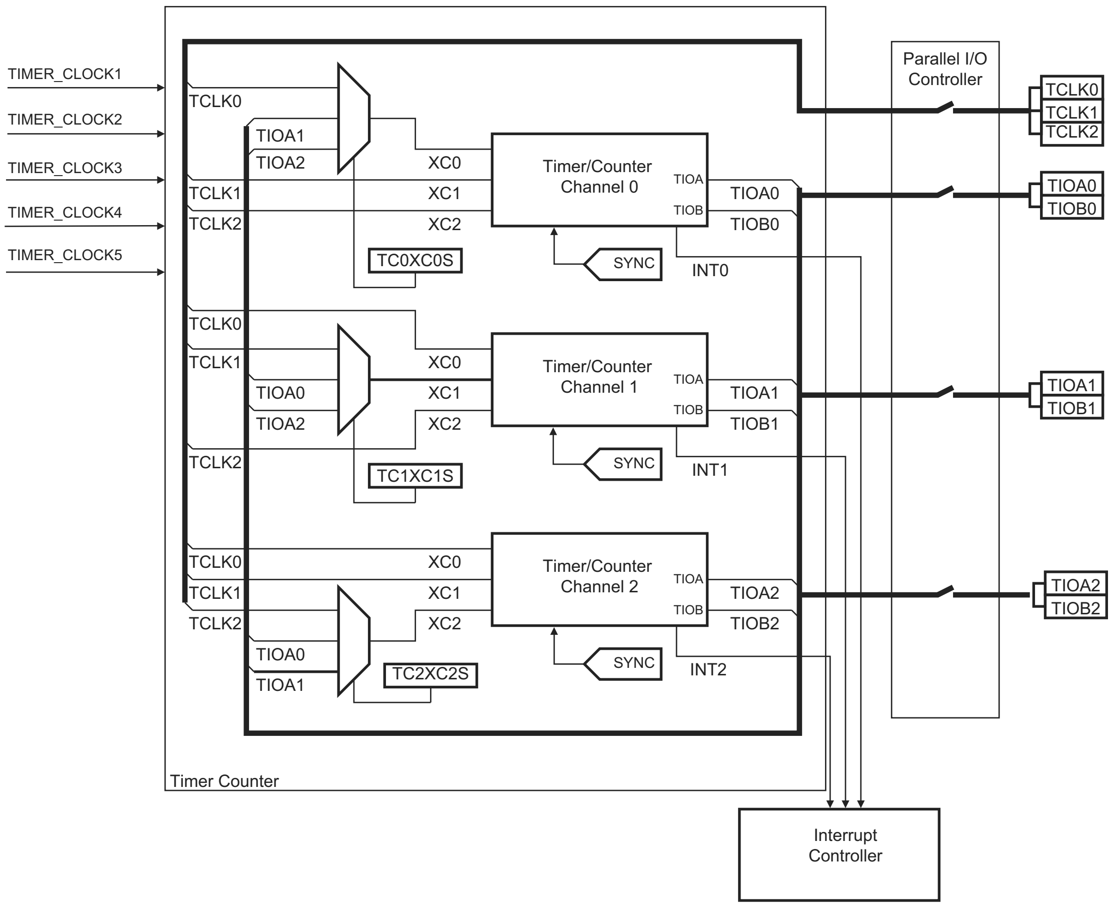
35.3 Block Diagram

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



41.6.5 ADC Channel Status Register

Name: ADC_CHSR

Address: 0x400AC018

Access: Read-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**
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Status**

0 = Corresponding channel is disabled.

1 = Corresponding channel is enabled.