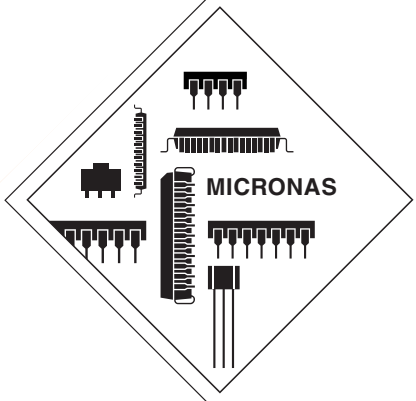


ADVANCE INFORMATION

PUC 303xA
Programmable Universal Controller “Zenon”

Edition Aug. 13, 2001
6251-565-1AI

MICRONAS

Contents, continued

Contents, continued

Contents, continued

Programmable Universal Controller “Zenon”

1. Introduction

The Programmable Universal Controller PUC 303xA is ideally suited for use in portable consumer devices and is equipped with an embedded ARM7TDMI processor, a highly flexible power management scheme, and embedded Flash.

Fig. 3–1 shows a simplified functional block diagram of the PUC 303xA.

2. Application Overview

The PUC 303xA meets the requirements of digital audio players, specific mobile network appliances, digital audio recorders (including water-marking, DRM), and wireless applications such as Bluetooth™.

The PUC 303xA device offers appropriate security features, embedded Flash and a variety of communication interfaces, including USB connectivity.

3. Features

- 64 MHz ARM7TDMI
- 48 kByte embedded SRAM
- 256 kByte embedded Flash
- Memory Protection unit
- Secure Mode controller
- Power Management Unit with integrated PLL
- 12 Mbit/s USB 1.1 function core with integrated transceiver (PUC 3030A only) and dedicated 8 kByte SRAM
- I^2C Master/Slave interface
- 66 General-purpose I/Os
- 5 x Synchronous Serial Port (including I^2S master), 24 Mbit/s max. speed as master, 16 Mbit/s max. speed as slave.
- 2 x UARTs / IrDA
- 3 x timer / counter
- Watchdog
- Interrupt Controller with hardware prioritization
- Real Time Clock (not available for all packages)
- Especially suited to interface with Micronas’ MAS 35x9F / MAS 3587F devices

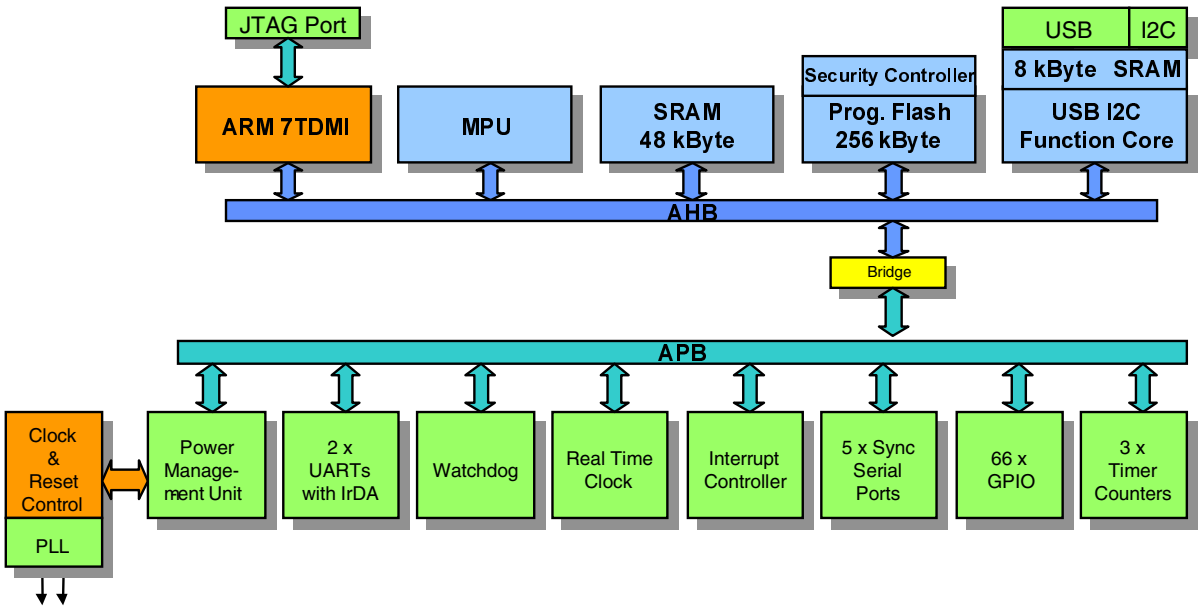


Fig. 3–1: Simplified functional block diagram of the PUC 303xA

4. Device Overview

4.1. 64-MHz ARM7TDMI

The embedded ARM7TDMI RISC processor runs at frequencies of up to 64 MHz.

The ARM processor and associated bus structures are configured for **little endian** operation.

4.2. 48 + 8 kByte SRAM

The embedded SRAM is divided into 48 kBytes zero-wait-state memory directly connected to the AMBA bus via an SRAM interface, and 8 kBytes contained within the USB space. The latter is accessible directly from both the USB core and the ARM7TDMI processor.

4.3. 256 kByte Embedded Flash

The embedded Flash is connected to the AHB via a Flash interface and allows storage of boot and application code. The Flash interface also implements protection modes to allow secure storage of application code and data.

4.4. Memory Protection Unit

A Memory Protection Unit (MPU) is provided for managing read and write access control for the ARM7TDMI memory map. It is also used to control the Flash / SRAM remapping functions.

4.5. Power Management Unit

The Power Management Unit (PMU) provides full control of the devices power modes, including 'operating', 'idle', 'stand-by', and 'off' modes.

4.6. 12-Mbit/s USB 1.1 function core

The USB 1.1 compliant function allows communication with a USB host-PC. Included is a built-in USB transceiver cell ensuring that minimal external components are required.

4.7. 66 General-purpose I/Os

Up to 66 General Purpose I/O (GPIO) pins can be used under software control. Many of these pins implement shared functionality to reduce pin count of the device.

4.8. 5 x Synchronous Serial Port

All of the five provided Synchronous Serial Ports can implement Motorola SPI, National Semiconductor Microwire, Texas Instrument Synchronous Serial and I^2S modes at speeds of up to 24 Mbit/s as a master and up to 16 Mbit/s as a slave.

4.9. 2 x UARTs / IrDA

Two fully independent UARTs are available running at a maximum of 115.2 kBaud. Each UART can also be configured for IrDA mode.

4.10. 3 x Timer / Counters

Three independent counter timers are provided for timing control during software routines, by using the various interrupts which can be generated. Two of the timers can also be clocked off an external clock, allowing edge counting and pulse width monitoring. In addition these two timers can also be used to generate periodic waveforms to an external device.

4.11. Watchdog

The Watchdog Timer provides a method to monitor software activity, and reset the device if a lock up occurs. A programmable timeout range is provided to allow for flexibility.

4.12. Interrupt Controller

The interrupt controller incorporates hardware prioritization of the 32 interrupt sources. This reduces the load on the processor when dealing with interrupts. All interrupts are maskable and can also be triggered by software.

4.13. Real Time Clock

The real time clock provides accurate reporting of time of day (hours, minutes and seconds) and calendar functions (day of week, date of month, month, year, century). It also has an alarm function (month, date, hour, minute or second resolution).

The real time clock is only available where application and package permit.

5. Functional Description

5.1. ARM7TDMI Processor and AHB/APB Buses

The Programmable Universal Controller PUC 303xA contains an embedded ARM7TDMI processor, which operates at a maximum frequency of 64 MHz.

The AMBA AHB bus connects the following peripherals to the ARM7TDMI processor:

– MPU
– Flash memory
– SRAM
– USB (Universal Serial Bus) core.

The AMBA APB bus connects the following peripherals to the AHB via the AHB/APB bridge:

– UARTs
– SSPs
– GPIO
– Interrupt Controller
– Timers
– Watchdog Timer
– Real Time Clock
– Power Management Unit

The ARM7TDMI processor and associated bus structures are configured for **little endian** operation.

5.1.1. Memory Map

The following table, Table 5–1, states the base addresses of memory mapped Flash, SRAM and peripherals. For detailed register offsets, relative to base addresses, refer to the functional descriptions of specific components.

The memory map may be dynamically changed by remapping the SRAM and Flash memory areas. Details of this operation can be found in the Memory Protection description (Section 5.4.4.).

Table 5–1: List of ARM7TDMI Processor Address Space

Base Address (Boot)	Base Address (Remapped)	Comment
0000.0000_{hex} 0008.0000_{hex}	0008.0000_{hex}	Embedded Flash (256KBytes) base address
0010.0000_{hex}	0000.0000_{hex}	Embedded SRAM (48KBytes) base address
0012.0000_{hex}	0012.0000_{hex}	USB base address
$001A.0000_{hex}$	$001A.0000_{hex}$	MPU base address
$001B.0000_{hex}$	$001B.0000_{hex}$	TPC (Test Port Controller) base address
$001C.0000_{hex}$	$001C.0000_{hex}$	Programmable Interrupt Controller base address
$001C.4000_{hex}$	$001C.4000_{hex}$	UART 1 base address
$001C.8000_{hex}$	$001C.8000_{hex}$	UART 2 base address
$001C.C000_{hex}$	$001C.C000_{hex}$	General Purpose I/O base address
$001D.0000_{hex}$	$001D.0000_{hex}$	Serial interface 1 base address (SSP1)
$001D.4000_{hex}$	$001D.4000_{hex}$	Serial interface 2 base address (SSP2)
$001D.8000_{hex}$	$001D.8000_{hex}$	Serial interface 3 base address (SSP3)
$001D.C000_{hex}$	$001D.C000_{hex}$	Serial interface 4 base address (SSP4)
$001E.0000_{hex}$	$001E.0000_{hex}$	Serial interface 5 base address (SSP5, I^2S)
$001E.4000_{hex}$	$001E.4000_{hex}$	Timers/Counters base address
$001E.8000_{hex}$	$001E.8000_{hex}$	Watchdog Timer base address
$001E.C000_{hex}$	$001E.C000_{hex}$	Real Time Clock base address
$001F.0000_{hex}$	$001F.0000_{hex}$	Power Management Unit base address

5.2. Clock Generation

The PUC 303xA implements a powerful and flexible clocking scheme, in order to provide maximum flexibility in power management, and to provide suitable clocks for the integrated peripherals. The features of the clock generation scheme are:

– Two oscillator clock pad inputs, either allowing direct connection to a crystal oscillator circuit or clock source.

– Software-selectable Phase Locked Loop (PLL) device on the main system oscillator clock generation logic.

– Programmable PLL dividers, allowing dynamic, glitch-free configuration of the PLL and a wide range of system clock frequencies.

– Automatic relock timing when the PLL is powered up, or when divider values are changed.

– Individually programmable clock dividers, on all but dedicated USB clocks, providing dynamic glitch-free frequency changing.

– Individually programmable clock enables on all clocks to stop them in their low phases.

– Maskable wake-up condition selection.

– Real Time Clock oscillator, with programmable enable.

– Real Time Clock support, by generating a 1 Hz clock from the Real Time Clock oscillator clock frequency ranging from 32768 Hz to 8.372224 MHz.

All of the features in the clock generation can be controlled by means of the addressable registers in the Power Management Unit.

Figure 5–1 shows the major blocks in the clock generation logic.

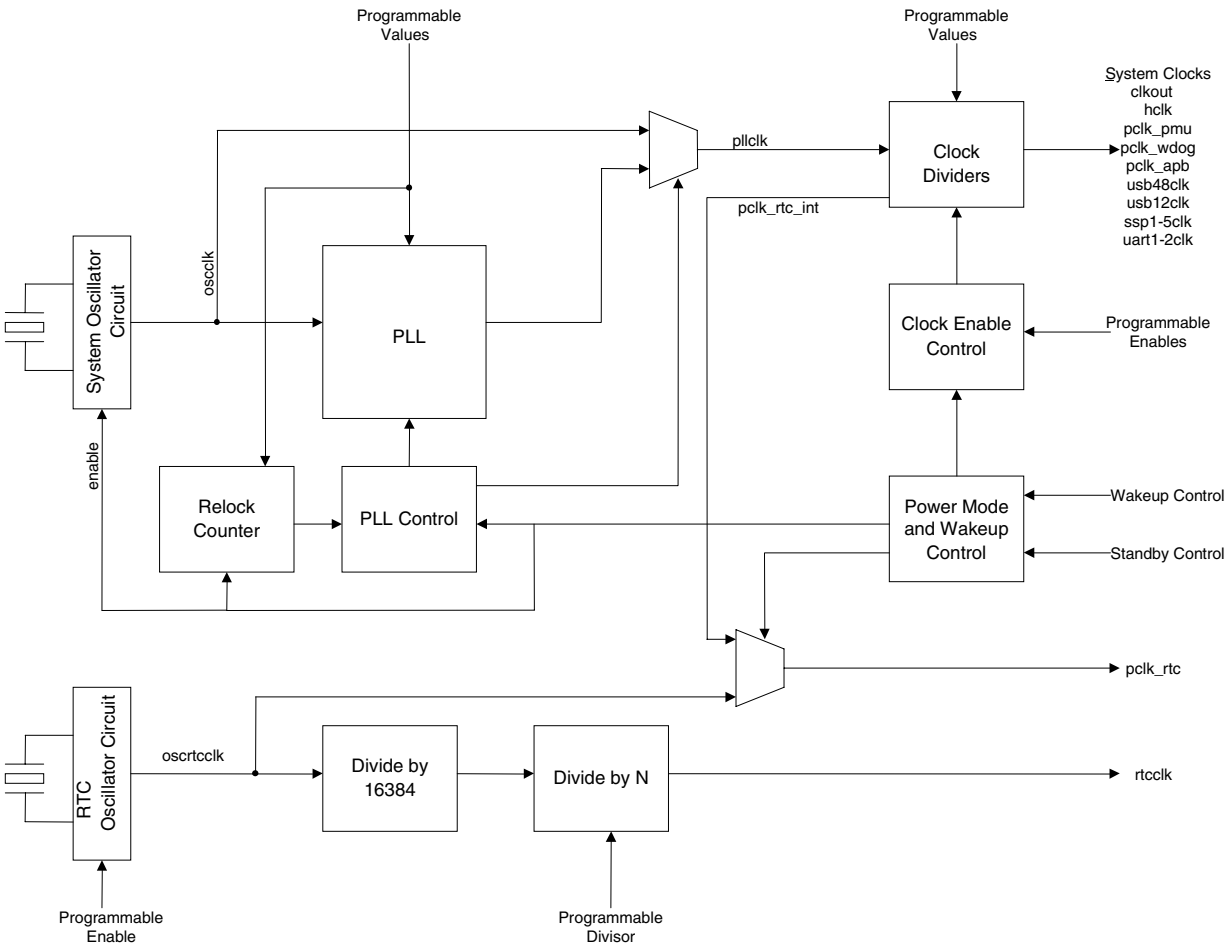


Fig. 5–1: Clock Generation Functional Block Diagram

The PUC 303xA provides two crystal oscillator circuits, one for the main system operation and one for the real time clock and standby operation. Both of these pads enable either direct clock input or a crystal to be connected.

The main system oscillator and subsequent circuitry is used to generate most of the required clocks for normal operation of the PUC 303xA.

The clock dividers can be sourced from either the oscillator clock input, oscclk, or from the output of an internal PLL, pll_fout. This selection is controlled by modifying the VCO_BYPASS bit of the PMU_CLK_EN register, described in Section 5.5.1.

The PLL output can be dynamically controlled by changing the programmable PLL dividers in the PMU_VCO_DIV register in the Power Management Unit (PMU, see Section 5.2.2. for details of how to update these dividers). The PLL control and relock counter blocks of the clock generator ensure safe handling of PLL divider changes, allowing the PLL output to settle before using it.

The clock enable and divider blocks are used to provide control over the individual clocks in the design. Each of the major clocks can be enabled or disabled, and the frequency controlled by means of separate programmable dividers. All of these options are controlled through the PMU registers, described in Section 5.5.1.

The real time clock oscillator pad is fed into a divider circuit, consisting of a fixed divide by 16384 and programmable divide by 2 to 511. The programmable element of this divider allows the oscillator input frequencies to be of the range 32.768 kHz to 8.372224 MHz, and still generate the required 1Hz clock, rtcclk, used by the Real Time Clock.

The power mode and wake-up control block is used to interpret information from the PMU registers, and then to place the clock generator in an appropriate power state. Please refer to the PMU section of this data sheet (Section 5.5.1.) for details of power modes.

Table 5–2 summarizes the clocks used or generated within the PUC 303xA.

Table 5–2: Clocks

Clock	Description	Sources	Divider Range	Frequency Range
oscclk	Main reference oscillator clock from pad	input clock pad	1	6 to 28 MHz
oscrtcclk	RTC oscillator clock from pad	input clock pad	1	32768 Hz to 8.372224 MHz
pll_fout	Output from PLL VCO	output from pll	See PLL section	80[1)] to 128[1)] MHz
pllclk	Multiplexed clock from oscclk and pll_fout (i.e bypass PLL)	pll_fout or oscclk	1	6 to 128 MHz
clkout	PUC 303xA output, used to clock external device (e.g. MASF).	pllclk or oscclk	1 to 15	334 kHz to 28 MHz[1)]
hclk	AHB Interface and ARM7TDMI system clock	pllclk	2 to 63	80 kHz to 64 MHz
pclk_pmu	APB system clock dedicated to Power Management Unit Registers	pllclk	shares common divider with hclk	
pclk_wdog	APB system clock dedicated to watchdog	pllclk	shares common divider with hclk	
pclk_rtc_int	Clock generator version of Real Time Clock APB system clock.	pllclk	shares common divider with hclk	
pclk_apb	APB system clock (used for all other APB peripherals)	pllclk	shares common divider with hclk	
pclk_rtc	APB system clock dedicated to the Real Time Clock. Generated by multiplexing pclk_rtc_int (for operating state) or oscrtcclk (for standby state)	pclk_rtc_int or oscrtcclk	1	32 kHz to 64 MHz
usb48clk	USB sampling clock	pllclk	2 and 4	48 MHz[2)]
usb12clk	USB bus clock	pllclk	8 and 16	12 MHz[2)]
ssp1clk	Sampling clock for SSP 1	pllclk	2 to 63	80 kHz to 64 MHz
ssp2clk	Sampling clock for SSP 2	pllclk	shares common divider with ssp1clk	
ssp3clk	Sampling clock for SSP 3	pllclk	shares common divider with ssp1clk	
ssp4clk	Sampling clock for SSP 4	pllclk	shares common divider with ssp1clk	
ssp5clk	Optional sampling clock for SSP 5. This is selectable between the divided pllclk or the ssp5clk_in input.	pllclk or ssp5clk_in	shares common divider with ssp1clk	
ssp5clk_in	Sampling clock supplied from external device (e.g. MASF)	gpio pad	1	24 MHz[1)]
uart1clk	Sampling clock for UART 1	pllclk	2 to 63	80 kHz to 15 MHz[1)]
uart2clk	Sampling clock for UART 2	pllclk	shares common divider with uart1clk	
rtcclk	Count enable for Real Time Clock (only used as an enable to pclk_rtc)	oscrtcclk	32768 to 4194304	1 Hz

[1)] theoretical limit overridden by physical constraints

[2)] nominal values for dividers 2/8 at pllclk=96MHz are also upper limit values

5.2.1. Clock Utilization Summary

Table 5–3, below, summarizes the utilization of clocks on a module-by-module basis.

Table 5–3: Clock Utilization

Module	Clocks Used Within Module
JTAG Port	tclk
ARM7TDMI processor	mclk (derived from hclk in AHB interface)
ARM7TDMI processor AHB Interface	hclk
Memory Protection Unit	hclk
48 kByte SRAM AHB Interface	hclk
256 kByte Flash AHB Interface	hclk
USB AHB Interface	hclk, usb12clk
USB core	usb48clk, usb12clk
AHB to APB Bridge	pclk_apb
UARTs	pclk_apb, uart*clk
SSPs	pclk_apb, ssp*clk
GPIO	pclk_apb
Watchdog	pclk_wdog
Real Time Clock	pclk_rtc, rtcclk[1)]
Timer / Counters	pclk_apb
Interrupt Controller	pclk_apb
Power Management Unit	pclk_pmu
Clock Generator	oscclk, oscrtcclk, pllclk

1) rtcclk is not used to clock registers but rather as a waveform that is sampled by pclk_rtc

5.2.2. Phase-Locked Loop Device

The PUC 303xA incorporates an embedded Phase Locked Loop (PLL) device, which can be selected to provide a source for the system clocks. The PLL can be configured under software control, providing an extraordinarily wide range of available frequencies and power-down capability.

The following features are available:

– A maximum output frequency of 128 MHz.

– A minimum output frequency of 80 MHz.

– Wide input clock frequency range of 6–28 MHz.

– Low jitter.

– Low power consumption in power down mode.

– Software selectable divider values for input and feedback dividers.

– Automatic re-lock waiting, where all system clocks are disabled until PLL output is stable.

Figure 5–2 shows a simplified diagram of the structure of the PLL.

The frequency relationship between the PLL output frequency and the source frequency is shown in the equation below.

$$pll\_fout = pll\_fin \times \frac{NF'}{NR'}$$

The division factors NR' and NF' are programmable through the Power Management Unit PMU_VCO_DIV Register. The actual value programmed needs to be two less than the required value.

For example, if a 96MHz output frequency clock is required, from a 13 MHz crystal, the dividers chosen could be NF' = 96 and NR' = 13. Therefore the divider values programmed into the PMU_VCO_DIV would be NF = 94 ($5E_{hex}$) and NR = 11 ($0B_{hex}$). So the value of the register should be set to $005E.000B_{hex}$.

Note that divider values of NF=192 and NR=26 could have been chosen to achieve the same frequency relationship, however a narrower loop bandwidth (larger NF divider value) increases PLL output jitter. This is due to lower comparison frequency at PFD. Therefore it is recommended to always choose the smallest values of equivalent divider sets.

Range for NF:

NF' from 2 ... 513 ⇒ **NF from 0 ... 511**

Range for NR:

NR' from 2 ... 33 ⇒ **NR from 0 ... 31**

Warning: Due to the very high flexibility in on-chip clock selection, special care must be taken not to overclock the PUC 303xA chip or regions of the chip. PLL division factors and clock dividers must be set to proper values, otherwise unpredictable operation or damage may result (refer to Table 5–2 for details on clock constraints).

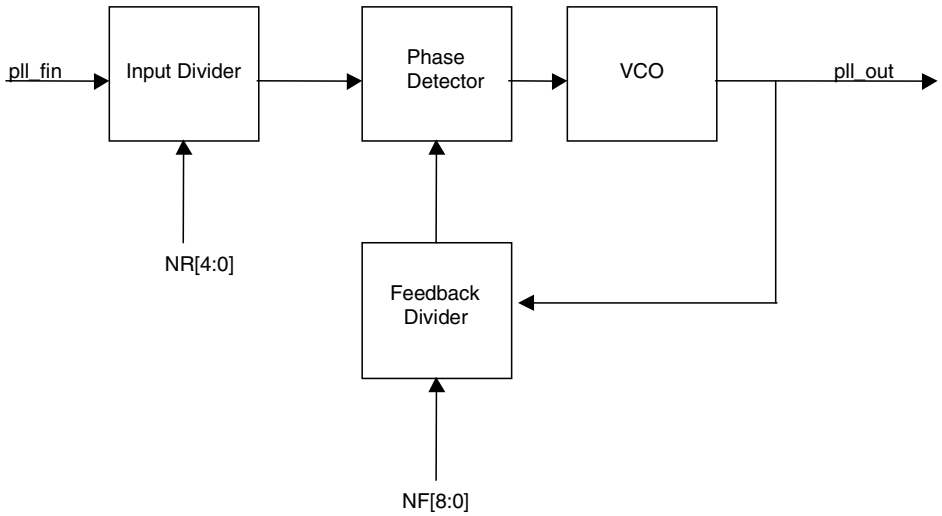


Fig. 5–2: Simplified Structure of the PLL

5.3. Power-on and Resets

The reset scheme is implemented to ensure seamless integration with various Micronas audio devices (e.g. MAS 35xxF family). There are four main reasons for the PUC 303xA to reset, each of which initiates a unique reset sequence. The four reset circumstances are:

– Reset pin pulled low for power-on and warm reset

– Wake-up from standby state

– Software induced reset

– Watchdog time-out induced reset

To ensure correct timings on the reset sequence, there are three counters within the reset controller.

Figure 5–3 below shows a full power-on reset sequence, illustrating the use of the three counters. The timing of this sequence relies on counting a fixed number of oscillator clock edges and hence will vary depending on oscillator frequency. The times shown are based on a 28 MHz system oscillator clock frequency.

The first counter, OSC_COUNT, is enabled whilst the main system oscillator clock is settling. At completion the CLKOUT clock is enabled. This counter is only used during a power-on or wake-up reset.

The second counter, EXT_COUNT, ensures sufficient time for external Micronas audio device's DC/DC converters to supply sufficient power to power up the rest of the PUC 303xA. At this point all system clocks are enabled within the PUC 303xA. Again this counter is only used during a power-on or wake-up reset.

The third counter, SYS_COUNT, is used to keep the ARM7TDMI processor and other peripherals in reset, whilst the security state machine in the Flash interface decides the security state of the device (see Section 5.4.2.).

For a waveform diagram of the reset sequence, see Figure 5–3 of this data sheet.

Table 5–3 summarizes how these resets are asserted depending on the source of the reset.

The internal PUC 303xA signals n_reset, wakeup_rst, soft_rst and n_wd_rst, are the trigger signals for the four reset sequences, and are activated as follows (refer to Section 5.5.1. for information about setting the PMU registers):

– n_reset: 'Reset' input pin of PUC 303xA pulled low at power-up or warm reset.

– wakeup_rst: conditioned "wake-up" input pin of PUC 303xA, causing a wake-up reset from standby state if the appropriate mask bit in the PMU_WAKE_MASK register in the PMU is set.

– soft_rst: Software induced reset caused by writing to bit [0] of the PMU_RESET register.

– n_wd_rst: Watchdog Timeout induced reset trigger from the watchdog module, as described in Section 5.5.8.

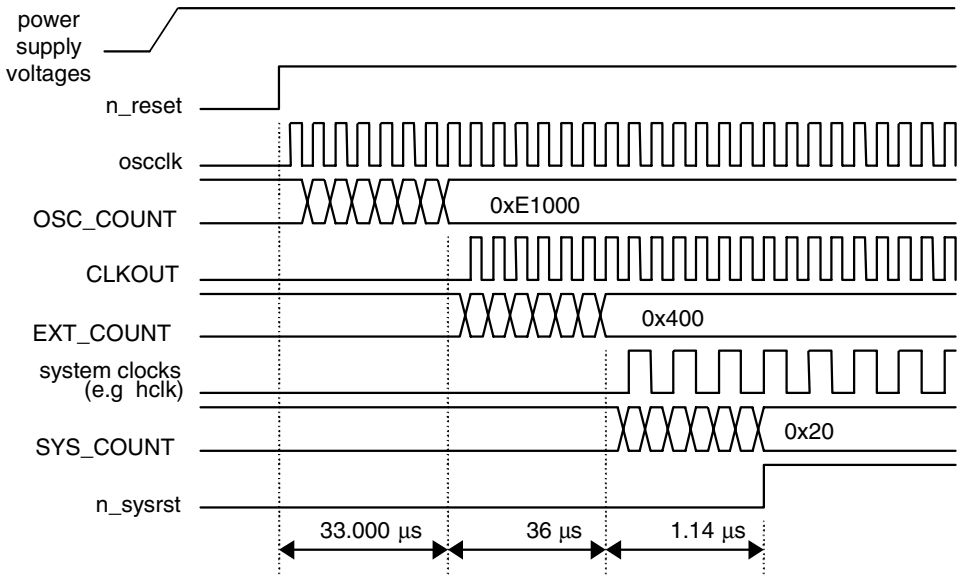


Fig. 5–3: Power-On reset sequence

Table 5–4: Resets

Source of Reset	PLL	Security and Flash Control	Watchdog	Real Time Clock	Rest of Chip
nRESET Pin	reset	reset	reset	reset	reset
Wake-up Reset	reset	reset	reset	X	reset
Software Reset	reset	X	X	X	reset
Watchdog Reset	X	X	X	X	reset
X=not reset by source					

5.4. AHB Peripherals

5.4.1. Flash Memory

5.4.1.1. Flash Memory Map

The PUC 303xA contains 256 kBytes of embedded Flash memory, which is split into two sections:

– 24 kByte BOOT area, for boot code,

– 232 kByte USER area, for application code,

There is also a 1 kByte INFORMATION area, for storing device specific production information, such as serial number.

These areas are illustrated below in Figure 5–4.

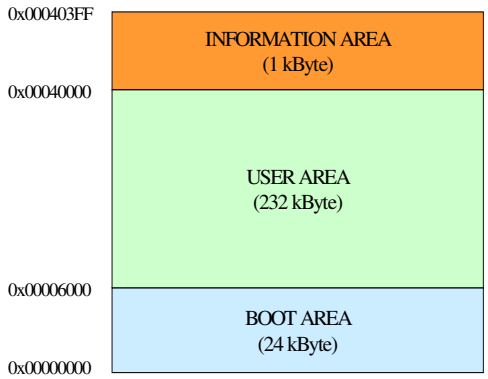


Fig. 5–4: Flash Memory Map (256 kBytes total)

The Flash Memory is organized in rows of 64 words of width 32 bits, so that each row contains 256 Bytes.

For the BOOT and USER areas, 8 adjacent rows form a page of 2048 Bytes, whereas the INFO area consists of a single page of 4 rows = 1024 Bytes.
The page organization is relevant for Flash Page Erase cycles.

The Flash may be accessed through three separate interfaces:

– from the ARM7TDMI processor,

– through the JTAG port with the ARM7TDMI processor in debug state,

– through a dedicated parallel flash programming test mode (Parallel TM) using GPIO pins (intended for fast mass production programming and testing)

The JTAG and parallel interfaces may be enabled or disabled, depending on the security status of the device (please refer to Section 5.4.2.).

The three interfaces have different access rights to the flash memory. Table 5–5 summarizes all of the access permissions.

The rest of this section describes accesses from the ARM7TDMI processor only, unless otherwise stated.

Table 5–5: Access Permissions

Interface	ARM7TDMI processor	JTAG	Parallel TM
BOOT Area Read	OK	OK	OK
BOOT Area Write	BLOCKED	OK	OK
USER Area Read	OK	OK	OK
USER Area Write	OK	OK	OK
INFO Area Read	OK	OK	OK
INFO Area Write	BLOCKED	BLOCKED	OK
BOOT Area Page Erase	BLOCKED	OK	OK
USER Area Page Erase	OK	OK	OK
INFO Area Page Erase	BLOCKED	BLOCKED	OK
OK=Access to region permitted BLOCKED=Access blocked by Security Controller			

5.4.1.2. Flash Configuration Registers

The Flash configuation registers are located after the top of the INFORMATION (INFO) area of the Flash at an offset of 0006.0000_{hex} from the Flash area base.

The memory map for the flash configuration registers is shown in Table 5–6. These are fully described in Table 5–7. All addresses given are offset addresses relative to the Flash configuration register base of 0006.0000_{hex}.

Table 5–6: Flash Configuration Register Map

Offset	R/W	Width	Name	Reset
00_{hex}	W	2	WAIT_STATE	N/A
04_{hex}	W	6	PERF_COUNT	N/A
08_{hex}	W	16	BOOT_PROT	N/A
$0C_{hex}$	W	18	ADDRESS_REG	N/A
10_{hex}	W	32	DATA_REG	N/A
14_{hex}	W	8	CONFIG_REG	N/A

Table 5–7: Flash Configuration Registers

Bits	Name	Function
WAIT_STATE		
[31:2]	Reserved	
[1:0]	WAIT_VAL	Number of added Wait States during Read cycle.(refer to Table 5–8)
PERF_COUNT		
[31:6]	Reserved	
[5]	PF_COUNT_EN	When this bit is set, the Flash performance counter is enabled. If the gap between two consecutive read accesses to the Flash is long enough to allow the count to expire, the embedded Flash device will enter standby state. With this bit disabled (’0’) the controller will never enter standby mode, delivering the most efficient times in terms of access speed.
[4:0]	PF_COUNT_VAL	These 5 bits correspond to the number of cycles after the last read access the Flash memory will enter standby mode. PF_COUNT_EN must be set to enable the value of PF_COUNT_VAL. Example: $01111_{bin}=15_{dec}$ hclk timeout cycles.
BOOT_PROT		
[31:16]	Reserved	
[15:0]	BOOT_WR_KEY	BOOT Flash area write enable key. A key value of $ABCD_{hex}$ must be written via the JTAG port, e.g. by using the ARM ICE debugger, to enable further programme/erase accesses to the BOOT area of the Flash. Once written, the ’ARM7TDMI-processor’-mode will have full access permissions, to allow dedicated embedded software routines to programme/erase the BOOT area.
ADDRESS_REG		
[17:8]	XADR_VAL (row)	Flash X address when SW_SELECT (s. CONFIG_REG) is enabled
[7:2]	YADR_VAL (column)	Flash Y address when SW_SELECT (s. CONFIG_REG) is enabled
[1:0]	Byte address	These bits are ignored. It is recommended to write them as ’0’.
DATA_REG		
[31:0]	DIN_VAL	Data on Flash data bus when SW_SELECT is enabled

Table 5–7: Flash Configuration Registers

Bits	Name	Function
CONFIG_REG **Note:** Set up the DATA_REG and ADDRESS_REG with the correct information before using this register.		
[31:8]	Reserved	
[7]	SW_SELECT	When set, this bit enables programme and erase mode.
[6]	WRITE_PROTECT	This bit is write only and when set will prevent SW_SELECT from being set. This prevents ANY writes or erases to the flash, until a new power-on or wake up reset.
[5]	PROG_VAL	The value on this register is output on the PROG line to the Flash when SW_SELECT is set.
[4]	ERASE_VAL	The value on this register is output on the ERASE line to the Flash when SW_SELECT is set.
[3]	RESERVED	
[2]	NVSTR_VAL	The value on this register is output on the NVSTR line to the Flash when SW_SELECT is set.
[1]	YE_VAL	The value on this register is output on the YE line to the Flash when SW_SELECT is set.
[0]	XE_VAL	The value on this register is output on the XE line to the Flash when SW_SELECT is set.

5.4.1.3. Flash Features

The following list describes the features of the embedded Flash in the PUC 303xA device:

– Separate Software Control for Flash Programming (see Section 5.4.1.5.)

– Hardware security mechanism for boot code (see Section 5.4.2.)

– Byte, Halfword and Word addressable for reads

– Programmable wait states, to permit up to three wait states to be inserted during any read access.

– BOOT area write protection disable register, that is accessible only through the JTAG ports. This allows the ARM7TDMI processor to program the BOOT area of Flash.

5.4.1.4. Programmable Wait States

The required number of wait states for read accesses to the Flash are shown in Table 5–8

.

Table 5–8: Wait States to be programmed in WAIT_STATE register

System Clock Freq (MHz)		Programmed Wait State Value
Min	**Max**	
> 50	64	3
> 25	50	2
> 16.6	25	1
> 0	16.6	0

5.4.1.5. Flash Programming and Erasing

All Writes and Erases are software controlled through the following three Flash configuration registers:

– ADDRESS_REG: Stores the address of the Flash location to be programmed.

– DATA_REG: Stores the data to be programmed into the Flash location.

– CONFIG_REG: Enables direct control of the control strobes to the Flash device. If a bit is set, then the corresponding signal is forced high to the Flash, if it is reset then the signal is low.

To perform a programming cycle to the flash, the following sequence should be performed:

– The address in the flash required to be written to, should be programmed into the ADDRESS_REG.

– The data to be written should be programmed into the DATA_REG.

– The CONFIG_REG should then be programmed a number of times in order to select the required sequence on the control strobes to the Flash.

An example of a typical programming cycle is shown in Figure 6.4.5.. The timing symbols and parameters are defined in Table 6–1. In addition, an example of a typical erase cycle is shown below in Figure 6–4.

Since the required programming times for a Flash programming cycle are in the order of milliseconds, the PUC 303xA's internal timers should be used to correctly time the programming sequence.

5.4.2. Security of Application Code

The security of the downloaded application code can be ensured by disabling external interfaces from directly reading the stored code in the Flash.

The Flash contains a readable memory location called the Protection Control Register, PCR, located at offset address 0001.8000_{hex}. Contents of this location will indicate whether the Parallel Flash interface, the JTAG interface, or the Test Peripheral Controller interface of the device are operating in secure or non-secure mode:

Secure mode is used to inhibit any accesses to the Flash from these interfaces, in order to ensure that application code cannot be read.

The default state of the device is to be non-secure. Once sensitive application code is loaded, the PCR can be programmed by either the JTAG or Parallel Flash interface, to enable security.

During a power on or wake-up reset, secure mode is selected for all interfaces. During the reset the Security Controller will read the protection control register and determine whether non-secure mode can then be selected.

Since the ARM7TDMI can boot from embedded Flash, this decision is made before the ARM7TDMI comes out of reset. This decision will remain fixed until the main system reset input is cycled again, or a wake-up from standby state occurs.

The bits of the protection control register are defined in :Table 5–9.

Table 5–9: Protection Control Register Bits

Bit	Function
2	JTAG Enable
1	Test Peripheral Controller Enable
0	Parallel Flash Interface Enable

The security of the protection control register shall be under the supervision of the boot software, by means of a unique access key.

5.4.3. SRAM

5.4.3.1. Features

– 48 kByte embedded SRAM accessible with zero wait states on the AHB.

– 8 kBytes embedded SRAM shared between the ARM7TDMI processor and the USB core.

– Byte-, Halfword-, and Word-addressable

– 48 kByte SRAM has a buffer to store last 32-bit data read. This can reduce power consumption, especially in THUMB mode

– Interfaces through AHB (the ARM7TDMI processor or JTAG)

– 48 kByte SRAM can be remapped to address 0_{hex} (see Section 5.4.4.)

5.4.3.2. General Overview

The PUC 303xA supports 48 kByte of memory directly on the AHB bus. Additional 8 kByte of SRAM are accessible through the USB interface. Accesses to this memory will incur additional wait states. Please refer to Section 5.4.5. for details of the USB interface. The rest of this section only applies to the main 48 kBytes of SRAM.

A buffer in the SRAM interface stores the last data read from the SRAM device. Power consumption can be reduced by releasing data to the ARM7TDMI processor from this internal buffer rather than actually reading the SRAM whenever possible.

This is particularly beneficial when the ARM7TDMI processor is running in THUMB mode as the instructions are only 16 bits wide. Therefore, one location in SRAM actually holds two instructions and can be serviced with only one read to the SRAM.

5.4.4. Memory Protection Unit

The Memory Protection Unit (MPU) provides the following functionality:

– manages memory protection within the ARM7TDMI core's memory map,

– controls the remapping of SRAM to the ARM7TDMI processor base address.

The MPU is an AHB peripheral. Access to all AHB peripherals including the MPU is controlled by an AHB address decoder, and is dependent upon the enable outputs from the MPU.

The MPU consists of 16 user-definable segments, each consisting of a start address MPUSTRTn, an end address MPUENDn, a read enable MPURDEN[n] and a write enable MPUWREN[n]. The start and end addresses are compared with bits 10 through 20, the most significant 11 bits used in the PUC 303xA architecture.

The current address is continually compared against the 16 segment address pairs. If an access address is detected to be (inclusively) within a segment's start and end values then permission for the ARM7TDMI processor to perform the current access will be granted using the state of the segment's read and write enable bits. The logic function of this mechanism for segment 0 is shown in Figure 5–5.

The MPU is only active when the ARM7TDMI processor is in USER mode, it has no protection effect in other processor operating modes.

The default values of the MPU registers dictate that the MPU has to be programmed before USER mode is entered, otherwise all memory accesses will result in a Data Abort interrupt. Thereafter, the MPU can also be programmed in USER mode as long, as its register locations are not protected.

5.4.4.1. SRAM Remapping

The REMAP bit of the MPUREMAP register defines the location of the SRAM within the ARM7TDMI processor's memory map. When 0 (default) the Flash based at 0008.0000_{hex} is mirrored at 0000.0000_{hex}, and the 48KB SRAM is located at 0010.0000_{hex}.
When this bit is set to 1, the SRAM is relocated from 0010.0000_{hex} to 0000.0000_{hex}. Address space above the remapped SRAM that was previously occupied by mirrored Flash becomes unused and causes an abort if accessed.

5.4.4.2. MPU Address Space

Table 5–10 below shows the registers within the MPU. Addresses are address offsets, relative to the MPU base address $001A.0000_{hex}$

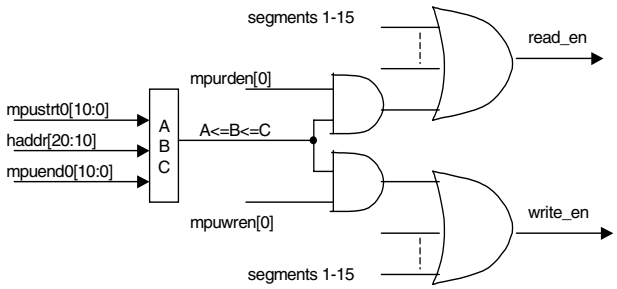


Fig. 5–5: Memory Protection Logic Function

Table 5–10: MPU Registers

Offset	R/W	Width	Name	Comment	Default
000_{hex}	R/W	2	MPUREMAP	bit 0: Reserved bit 1: REMAP (R/W) =0 mirrors Flash to the ARM7TDMI processor base address =1 remaps SRAM to the ARM7TDMI processor base address	0_{hex}
004_{hex}	R/W	16	MPURDEN	Bits 0 through 15 are active high read enables for MPU segments 0 through 15 respectively.	0000_{hex}
008_{hex}	R/W	16	MPUWREN	Bits 0 through 15 are active high write enables for MPU segments 0 through 15 respectively	0000_{hex}
$00C_{hex}$	R/W	27	MPUSEG0	Start and End address for MPU segment 0	000.0000_{hex}
010_{hex}	R/W	27	MPUSEG1	Start and End address for MPU segment 1	000.0000_{hex}
014_{hex}	R/W	27	MPUSEG2	Start and End address for MPU segment 2	000.0000_{hex}
018_{hex}	R/W	27	MPUSEG3	Start and End address for MPU segment 3	000.0000_{hex}
$01C_{hex}$	R/W	27	MPUSEG4	Start and End address for MPU segment 4	000.0000_{hex}
020_{hex}	R/W	27	MPUSEG5	Start and End address for MPU segment 5	000.0000_{hex}
024_{hex}	R/W	27	MPUSEG6	Start and End address for MPU segment 6	000.0000_{hex}
028_{hex}	R/W	27	MPUSEG7	Start and End address for MPU segment 7	000.0000_{hex}
$02C_{hex}$	R/W	27	MPUSEG8	Start and End address for MPU segment 8	000.0000_{hex}
030_{hex}	R/W	27	MPUSEG9	Start and End address for MPU segment 9	000.0000_{hex}
034_{hex}	R/W	27	MPUSEG10	Start and End address for MPU segment 10	000.0000_{hex}
038_{hex}	R/W	27	MPUSEG11	Start and End address for MPU segment 11	000.0000_{hex}
$03C_{hex}$	R/W	27	MPUSEG12	Start and End address for MPU segment 12	000.0000_{hex}
040_{hex}	R/W	27	MPUSEG13	Start and End address for MPU segment 13	000.0000_{hex}
044_{hex}	R/W	27	MPUSEG14	Start and End address for MPU segment 14	000.0000_{hex}
048_{hex}	R/W	27	MPUSEG15	Start and End address for MPU segment 15	000.0000_{hex}

Table 5–11: MPUSEGx Register

Bit	31:27	26:16	15:11	10:0
Field	Reserved	MPUEND[10:0]	Reserved	MPUSTRT[10:0]
This register stores the most significant 11 bits of a Start address MPUSTRTx and the most significant 11 bits of an End address MPUENDx of user-definable memory segment x. Both, MPUEND[10:0] and MPUSTRT[10:0] are compared continually with the most significant bits [20:10] of the address bus, thus implying a resolution of 1 kByte as the smallest segment quantity that can be permitted access in USER mode.				

5.4.5. USB Interface

The PUC 3030A* device incorporates a USB specification 1.1 compliant slave interface with the following features:

– USB specification 1.1/1.0 compliant, with built in transceiver cell.

– Full speed slave interface with 12 Mbit/s bit rate.

– 8 kByte dedicated USB SRAM.

– The hardware provides the following end points:

  • EP0: dedicated.

  • EP1 to EP7 software-programmable.

– Can be programmed with bulk endpoints, interrupt, or isochronous endpoints.

– Suspend mode, which can be used to control PUC 3030A power management.

– Embedded controller, dedicated to USB housekeeping.

– Interface with AHB allows access to whole address space of USB core from the ARM7TDMI processor processor, including SRAM.

– Three optimized access types from the ARM7TDMI processor, including pipelined read to free up the processor for other tasks.

– Application code download of embedded controller software from embedded Flash to USB SRAM.

The architecture of the USB core is illustrated in Figure 5–6.

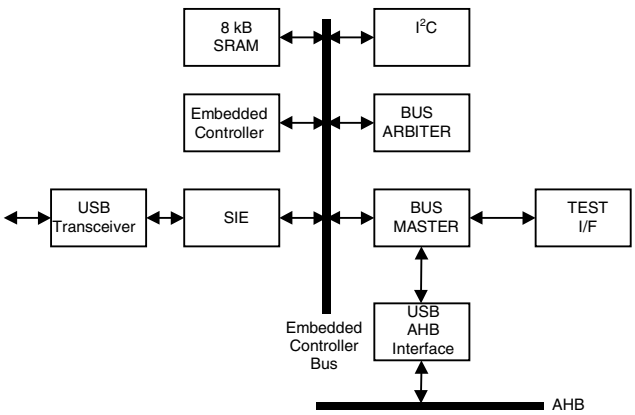


Fig. 5–6: USB Function Core Functional Diagram

The USB core comprises the following blocks:

– Embedded controller: Used to control and update the USB endpoint registers and for general housekeeping of the SRAM.

– USB serial interface (SIE): For USB transaction handling and transfer of data to SRAM.

– 8 kBytes SRAM: Used primarily for storing the USB endpoint registers, data buffers and application code for the embedded controller. The USB SRAM may also be used as scratchpad RAM by the ARM7TDMI processor if the USB interface is not being used (scratchpad RAM functionality is also available on PUC 3033A).

– Bus arbiter: for arbitration between the 3 bus masters: SIE, embedded controller and Test/USB AHB interface.

– Bus Master: A port to allow an external source (either the Test or the USB AHB interface) direct control of any addressable location.

– I^2C bus master and slave devices, used under control of the embedded controller (functionality is also available on PUC 3033A).

The ARM7TDMI processor can access any region of memory within the USB core space, including the SRAM. However because of clock domain differences between the two buses, accesses from the ARM7TDMI processor may take several cycles, and hold up the processor with wait states. The exact quantity of cycles taken depends on the relationship between the hclk and usb12clk, and the traffic load on the USB embedded controller bus.

To allow the ARM7TDMI processor to complete other system tasks during this time, the USB core memory map is repeated four times in the ARM7TDMI processor memory map. Each copy performs a different type of access to the USB core as shown in Figure 5–12 (yyyy = address offset within USB core)

* On PUC 3033A devices the USB interface is not available but SRAM and I^2C blocks are functional.

Table 5–12: USB Core Access

Address	Read or Write	Description
0012.yyyy$_{hex}$	read or write	Initiate a new byte, half or word read or write, and wait for completion. In this mode the ARM7TDMI processor will be held with wait states, until the access has completed.
0014.yyyy$_{hex}$	read	First pipeline read. Initiates a new byte, half or word read, but does not wait for completion. In this mode the ARM7TDMI processor is immediately released, unless the USB interface is busy with a previous access. Data read by this access is invalid. This enables the data to be immediately available the next time the ARM7TDMI processor performs a read.
	write	Initiate a new byte, half or word write. In this mode the ARM7TDMI processor is immediately released, unless the USB interface is busy with a previous access.
0016.yyyy$_{hex}$	read	Pipelined read. Initiates a new byte, half or word read, and waits for previously requested data. In this mode the read data supplied will be the previously addressed data. In this way successive reads can be requested, so that the ARM7TDMI processor is free to handle the data in between access to the USB.
	write	Initiate a new byte, half or word write. In this mode the ARM7TDMI processor is immediately released, unless the USB interface is busy with a previous access.
0018.yyyy$_{hex}$	–	Reserved for test purposes

5.5. APB peripherals

The PUC 303xA device incorporates numerous commonly used peripherals which make the device very flexible and ideally match the demands of Micronas' MASF devices. These peripherals are accessible to the ARM7TDMI processor core through the AMBA Advanced Peripheral Bus (APB) and are described in the following sections.

5.5.1. Power Management Unit

The Power Management Unit is used to control the various power states of the device.

5.5.1.1. Power States

The device supports the following power-related states:

– The OPERATING State
– The IDLE State
– The STANDBY State (if Real Time Clock RTC is available)
– The OFF State (if RTC is unavailable)

Figure 5–7 illustrates how these states are reached.

5.5.1.1.1. OPERATING State

In the OPERATING state, all peripherals can be active, depending on the clocks enabled by the Power Management Unit.

Within the OPERATING state, the PLL may be enabled or turned off/bypassed.

5.5.1.1.2. IDLE State

In the IDLE state, the clock to the ARM7TDMI processor bus and the watchdog timer clock are disabled, while the peripherals on the APB bus may remain active. The device will return to the OPERATING state if an interrupt is generated from any of the APB peripherals.

The PLL will operate as in OPERATING state.

5.5.1.1.3. STANDBY State

This state is only valid in applications and with device packages where the Real Time Clock (RTC) is available.

In the STANDBY state, all the clocks to both the AHB and the APB are disabled, with the exception of the RTC clocks.

For further power efficiency, the PLL and main oscillator are also powered down when the device is in the STANDBY state.

The device re-enters OPERATING state when either an RTC alarm, a USB plug, end of USB suspend, or a valid wake-up condition is detected.

5.5.1.1.4. OFF State

In the OFF state, all clocks are disabled and can only be restarted following a power-on or warm reset via n RESET pin.

OFF state is only valid if the RTC is not available with application or device package.

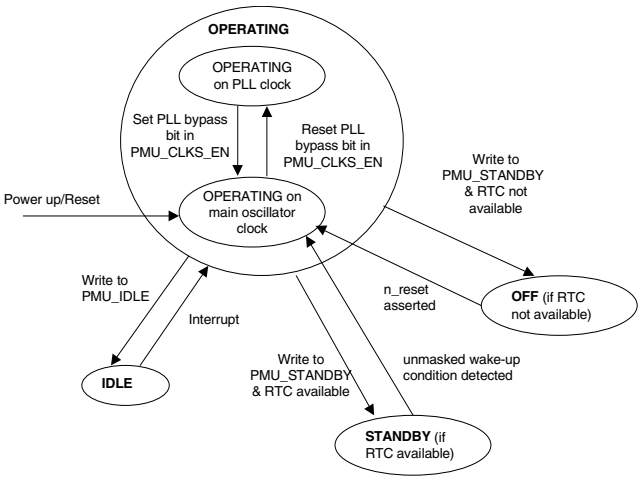


Fig. 5–7: Power Management Unit: Power States

5.5.1.2. Power Management Register Descriptions

The power management unit is a peripheral on the APB, and thus will enable software configuration of the states as shown in the register map in Table 5–13.

Addresses shown are offset addresses from the PMU base address of $001F.0000_{hex}$. For more detailed descriptions of these registers, see Table 5–14 to Table 5–26.

Table 5–13: Power Management Unit Register Map

Offset	Name	Access	Bits	Description	Default
00_{hex}	PMU_VCO_DIV	R/W	24:0	Dividers used by the PLL (use with PMU_VCO_DIV_UP)	$05E.000B_{hex}$
04_{hex}	PMU_VCO_DIV_UP	W/O	31:16	Update register for PLL dividers	N/A
08_{hex}	PMU_HCLK_DIV	R/W	5:0	Divider used for hclk/pclk generation	04_{hex}
$0C_{hex}$	PMU_CLKOUT_DIV	R/W	4:0	Divider used for CLKOUT generation	11_{hex}
10_{hex}	PMU_UARTCLK_DIV	R/W	5:0	Divider used for uartclk generation (UART 1–2)	$3F_{hex}$
14_{hex}	PMU_SSPCLK_DIV	R/W	5:0	Divider used for sspclk generation (SSP 1–5)	$3F_{hex}$
18_{hex}	PMU_RTCCLK_DIV	R/W	8:0	Divider used for rtcclk generation	002_{hex}
$1C_{hex}$	PMU_USBCLKS_DIV	R/W	1 bit	Divider used for usb48clk and usb12clk generation	1_{hex}
20_{hex}	PMU_CLKS_EN	R/W	29:0	Enables for each controllable clock domain	$013F.3711_{hex}$
24_{hex}	PMU_IDLE	W/O	31:0	Forces the device into IDLE state	N/A
28_{hex}	PMU_STANDBY	W/O	31:0	Forces the device into STANDBY or OFF state	N/A
$2C_{hex}$	PMU_RESET	R/W	31:0	Forces a software reset	unknown
30_{hex}	PMU_WAKE_SEL	R/W	3:0	Selects which events may cause a wake-up	0_{hex}

Table 5–14: PMU_VCO_DIV Register, Offset 00_{hex} , Reset $005E.000B_{hex}$

Bit	31: 25	24: 16	15: 5	4: 0
Field	Reserved	NF[8:0]	Reserved	NR[4:0]
This register stores the values used as the NF[8:0] Feedback divider and NR[4:0] Input divider for the PLL (note that programmed values need to be 2 less than their corresponding integer values in the quotient that is to be realised by the PLL). When a write is performed to the PMU_VCO_DIV_UP register, the clock generator stops all of the clocks in the system at a low phase, before presenting the divider values from PMU_VCO_DIV to the PLL. The required settling time for the PLL is then waited, before clocks are restarted. For this register to have an affect on the system, the PMU_VCO_DIV_UP register must be written to with the correct key.				

Table 5–15: PMU_VCO_DIV_UP Register, Offset 04_{hex} , Reset N/A

Bit	31: 16	15: 0
Field	Update Key This half word must be set to 4321_{hex} to force the system to use the new PLL dividers.	Reserved
This register must be written to with the correct key, in order for a new value in the PMU_VCO_DIV register to have an affect on the system operation. The register has a key that is required in bits [31:16] of the write data.		

Table 5–16: PMU_HCLK_DIV Register, Offset 08_{hex} , Reset 0000.0004_{hex}

Bit	31:6	5:0
Field	Reserved	HCLK_DIV[5:0] AHB system clock divider, range 2 to 63
This register stores the divider used within the clock generator, to divide the PLL bypass clock to produce the AHB system clock, hclk. The divider is alo used for pclk_pmu, pclk_wdog, pclk_apb, and pclk_rtcint. A value of 0 or 1 in this register will stop hclk.		

Table 5–17: PMU_CLKOUT_DIV Register, Offset $0C_{hex}$, Reset 0000.0011_{hex}

Bit	31:5	4	3:0
Field	Reserved	MF_SEL Selects source for CLKOUT divider 1: Always use main oscillator clock (oscclk) 0: Use PLL bypassed clock (pllclk)	MASF_DIV[3:0] CLKOUT divider, range 1–15
This register stores the divider used within the clock generator, to divide the selected clock to produce the MASF CLKOUT output clock. The source clock for this division is also selectable between the reference oscillator clock and the PLL bypassed clock. A value of 0 in MASF_DIV[3:0] will stop CLKOUT.			

Table 5–18: PMU_UARTCLK_DIV Register, Offset 10_{hex} , Reset $0000.003F_{hex}$

Bit	31:6	8:0
Field	Reserved	UART_DIV[5:0] UART clock divider, range 2–63
This register stores the divider used within the clock generator, to divide the PLL bypass clock to produce the dedicated UART clocks, uart1clk and uart2clk. A value of 0 or 1 will stop the UART clocks.		

Table 5–19: PMU_SSPCLK_DIV Register, Offset 14_{hex} , Reset 0000.003F$_{hex}$

Bit	31:6	5:0
Field	Reserved	SSP_DIV[5:0] SSP clock divider, range 2–63
This register stores the divider used within the clock generator, to divide the PLL bypass clock to produce the dedicated SSP clocks, ssp1clk, ssp2clk, ssp3clk, ssp4clk, ssp5clk. Note that if ssp5clk is selected to be derived from an external source, this divider does not affect the clock used by SSP 5. A value of 0 or 1 will stop the SSP clocks.		

Table 5–20: PMU_RTCCLK_DIV Register, Offset 18_{hex} , Reset 0000.0002$_{hex}$

Bit	31:9	8:0
Field	Reserved	RTC_DIV[8:0] RTC clock divider, range 2–511
This register stores the divider used within the clock generator, to divide the RTC oscillator clock input to produce the required 1 Hz Real Time Clock counter clock, rtcclk. The divider circuit consists of a fixed 16384 divider and a programmable divider in the range of 2 to 511. This allows the required 1 Hz to be generated from a range of clock inputs from 32768 Hz up to 8.372224 MHz, in increments of 16384 Hz. A value of 0 or 1 will stop the RTC clock.		

Table 5–21: PMU_USBCLK_DIV Register, Offset $1C_{hex}$, Reset 0000.0001$_{hex}$

Bit	31:1	0
Field	Reserved	USBCLK_DIV[0] USB clock divider
When this register is set to 0, the USB48CLK frequency is obtained by dividing pllclk by 2, and the USB12CLK frequency is obtained by dividing pllclk by 8. Use this configuration for nominal operation of USB (when pllclk = 96MHz) or to access USB core's SRAM at pllclk frequencies less than or equal to 96MHz. When this register is set to 1, the USB48CLK frequency is obtained by dividing pllclk by 4, and the USB12CLK frequency is obtained by dividing pllclk by 16. Use this configuration to access the USB core's SRAM for pllclk frequencies above 96MHz.		
This register is useful when USB operation is not needed but USB SRAM should stay accessible for system clocks above 48 MHz (pllclk above 96MHz) without violating timing constraints for the USB SRAM.		

Table 5–22: PMU_CLKS_EN Register, Offset 20_{hex} , Reset $013F.3711_{hex}$

Bit	29	28	27:25	24	23:22
Field	OSCRTCCLK_EN If low the RTC oscillator pad is disabled.	RTCCLK_EN If low the generated 1Hz clock enable to the RTC is disabled.	Reserved	CLKOUT_EN If low the generated CLKOUT output clock is disabled.	Reserved
Bit	21	20	19	18	17
Field	USBCLK_EN If low the dedicated clocks (usb12clk & usb48clk) to USB are disabled.	SSP5CLK_EN If low the dedicated clock to SSP 5 is disabled (unless fed by external source).	SSP4CLK_EN If low the dedicated clock to SSP 4 is disabled.	SSP3CLK_EN If low the dedicated clock to SSP 3 is disabled.	SSP2CLK_EN If low the dedicated clock to SSP 2 is disabled.
Bit	16	15:14	13	12	11
Field	SSP1CLK_EN If low the dedicated clock to SSP 1 is disabled.	Reserved	UART2CLK_EN If low the dedicated clock to UART 2 is disabled.	UART1CLK_EN If low the dedicated clock to UART 1 is disabled.	Reserved
Bit	10	9	8	7:5	4
Field	PCLKRTC_EN If low the APB system clock to the RTC is disabled.	PCLKWDOG_EN If low the APB system clock to the watchdog is disabled.	PCLKAPB_EN If low the APB system clock is disabled to all APB peripherals which do not have dedicated PCLK clocks.	Reserved	HCLK_EN If low the AHB system clock is disabled.
Bit	3:1				0
Field	Reserved				VCO_BYPASS If set high, the PLL is powered down and bypassed so that oscclk feeds the PLL bypass clock, pllclk. If low, pllclk will be fed by the PLL generated clock, pll_fout.
This register controls the clock and enables PLL bypass mode selection.					

Table 5–23: PMU_IDLE Register, Offset 24_{hex} , Reset N/A

Bit	31:16	15:0
Field	IDLE_KEY This half word must be set to $FEDC_{hex}$ to force the system into IDLE state.	Reserved

A write to this register will force the state of PUC 303xA to change from OPERATING mode into IDLE mode. This will immediately disable both the hclk and pclk_wdog clocks. These will be enabled again when an edge is seen on the NIRQ or NFIQ interrupt lines of the ARM7TDMI core.

There is a delay of around 4 cycles in writing to the IDLE register and the clock being stopped. For this reason it is recommended that at least 4 NOP instructions are placed directly after the IDLE register write.

The ARM7TDMI processor execution will initially carry on from where it left off when an interrupt is received, and then jump to the interrupt handler.

The register has a key that is required in bits [31:16] of the write data.

Table 5–24: PMU_STANDBY Register, Offset 28_{hex} , Reset N/A

Bit	31:16	15:0
Field	STANDBY_KEY This half word must be set to $CDEF_{hex}$ to force the system into STANDBY state.	Reserved

A write to this register, with the correct key, will force the state of PUC 303xA to change from OPERATING state into STANDBY or OFF state, depending on whether the RTC oscillator is active.

If the RTC oscillator is inactive, either unavailable, as in the 81-pin BGA package, or not enabled, the OFF state will be entered. Otherwise STANDBY state shall be entered.

In either of the new states all clocks in the system, that are not associated with the Real Time Clock, will be stopped in their low phases.

Clocks associated with the Real Time Clock (oscrtcclk, rtcclk and pclk_rtc) will remain active or inactive. The APB clock for the RTC, pclk_rtc, will now be fed by oscrtcclk, instead of being derived from the PLL bypass clock.

To leave the OFF state a power-on or warm reset (low on reset pin) is required. This will return the system to OPERATING state driven by the main oscillator clock (PLL bypassed).

To leave the STANDBY state, either a power-on or warm reset can be used, or an unmasked wake-up condition (see Table 5–26) is required. Any of these valid events will result in a wake-up reset sequence, and the device will return to OPERATING state using the main oscillator clock (PLL bypassed).

ARM7TDMI processor execution will always begin at the reset vector 0_{hex} after exiting OFF and STANDBY states.

Table 5–25: PMU_RESET Register, Offset $2C_{hex}$, Reset value: unknown, depends on source of reset

Bit	31:16	15:3	2	1	0
Field	**SOFTRST_KEY**	Reserved	**WD_RST**	**POWERON_RST**	**SOFT_RST**
This register can be read after the end of a reset sequence, to find out the reason for the reset. On a read all of the status bits are cleared. This register can be used to force a software reset from the system reset controller.					

Bit Definitions:

Label	Bits	Access	Description
SOFTRST_KEY	[31:16]	WO	SOFTRESET_KEY-Field: This field must be set to **ABCD**$_{hex}$ to allow a write to the register.
WD_RST	[2]	RO	This read only bit is set if the latest reset is detected as a watchdog reset. On a read access to the register this bit is cleared. **1 R**: indicates a watchdog reset has occurred, clears bit **0 R**: no watchdog reset since last read
POWERON_RST	[1]	RO	This read only bit is set if the latest reset is detected as a power-on or warm reset. On a read access to the register this bit is cleared. **1 R**: indicates a power-on or warm reset has occurred, clears bit **0 R**: no power-on or warm reset since last read
SOFT_RST	[0]	RW	Writing 1 to this bit forces a software reset only if it was previously 0. After a software reset, this bit can be read to indicate that a software reset was performed. It must be cleared by reading the register, before another software reset can be performed. **1 R**: indicates a software reset has occurred, clears bit **0 R**: no software reset since last read **1 W**: forces software reset **0 W**: no effect

Table 5–26: PMU_WAKE_SEL Register, Offset 30_{hex} , Reset 0000.0000_{hex}

Bit	31:4	3	2	1	0
Field	Reserved	USB_SENSE If set high, then detect the rising edge of the USB sense input as a wake-up.	USB_SUP If set high, then detect the falling edge of the USB suspend mode as a wake-up.	RTC_INT If set high then detect the rising edge of the RTC interrupt as a wake-up.	WAKE_PIN If set high then detect the rising edge of the wake-up pin input as a wake-up.
This register is used to select which sources will generate a valid wake-up condition, to force PUC 303xA from STANDBY state back into OPERATING state. These sources are only valid if the RTC oscillator clock input oscrtcclk is active. The sources are logically ORed to form the wake-up condition.					

5.5.2. Interrupt Controller

Since the ARM7TDMI processor core provides only two interrupt input lines, an interrupt controller is used to manage the various on-chip interrupt sources.

The interrupt controller has been designed to optimize the performance of interrupt handling, by reducing latency and providing priority nesting of interrupts.

5.5.2.1. Features

– Up to 32 Interrupt Sources (31 IRQ, 1 FIQ)
– Programmable Selection for FIQ Source
– Hardware Priority Encoding for IRQ
– Selectable Source Type (Hardware / Software)
– All Interrupts Maskable
– Programmable Interrupt Source Type
  – Edge, Triggered
  – Level-Sensitive.

5.5.2.2. Interrupt List

Although there are a possible 32 interrupts, only 25 are directly used by hardware. The remaining 7 can only be used for triggering an interrupt through software. The interrupt sources are listed in Table 5–27.

The interrupt index column is used to decode the IRQSTATUS indexed interrupt status register, and defines the value to write back to the IRQCOMPLETE register.

Table 5–27: Interrupt List

Interrupt number	Interrupt index	Function
0	1	Realtime clock interrupt
1	2	WATCHDOG interrupt
2	3	Timer 1 interrupt
3	4	Timer 2 interrupt
4	5	Timer 3 interrupt
5	6	USB interrupt
6	7	USB SIE interrupt
7	8	I2S (SSP 5) interrupt
8	9	SSP 1 interrupt
9	10	SSP 2 interrupt
10	11	SSP 3 interrupt
11	12	SSP 4 interrupt
12	13	UART 1 interrupt
13	14	UART 2 interrupt
14	15	ARM7TDMI comms rx
15	16	ARM7TDMI comms tx
16	17	GPIO a interrupt
17	18	GPIO b interrupt
18	19	GPIO c interrupt
19	20	GPIO d interrupt
20	21	GPIO e interrupt
21	22	GPIO f interrupt
22	23	GPIO g interrupt
23	24	GPIO h interrupt
24	25	GPIO i interrupt
25 - 31	26-32	Can be used as software interrupts only.

5.5.2.3. General Overview

The Interrupt Controller can process and control up to 32 interrupt sources. Of these, 31 are IRQ style interrupts and the final, programmable source is the Fast Interrupt (FIQ). The FIQ service routine offers a considerable time advantage over the IRQ, due to a combination of reduced software and hardware requirements. The individual sources do not require to be held until the relevant service routine is complete, although this is optional. During the service of one interrupt, it is also possible to store one pending interrupt request from any source, allowing up to 31 interrupt to be pending and one being serviced.

The PUC 303xA allows to apply one of eight levels of priority to each of the 32 sources. Note, however, that these only apply to the IRQ sources and will have no effect on the FIQ. The FIQ is, in any case, granted a higher hardware priority than IRQ anyway. The rules of the priority encoder are as follows:

– Every source can have a priority level between 0 and 7, programmed by software.
– 7 is the highest priority, 0 the lowest. The reset value of the priority is 0 for all sources.
– If, during a particular service routine, a new interrupt source of lower priority is enabled, the source will be stored as pending until the current source is complete.
– If, during a particular service routine, a new interrupt source of higher priority is enabled, the IRQ line will be enabled to permit the ARM7TDMI processor to service the new interrupt.*
– If the Interrupt Controller is free and 2 interrupt sources are enabled simultaneously, the interrupt source with the higher priority is granted. If the sources are of equal priority, the lower interrupt source will be granted. Source 1 is, for example, higher priority than source 31
– If the programmed FIQ source is enabled during an IRQ service routine, the FIQ line will be enabled to permit the ARM7TDMI processor to service the new interrupt.

Any of the 32 Interrupt sources can be programmed to be high or low level sensitive; or rising or falling edge triggered. This, along with the priority is software controlled through the 32 R/W registers INTxTYPE.

There are multiplexers at the source input to enable each of the 32 sources by software or hardware. At reset, these multiplexers connect all of the sources directly to the hardware interrupts.

The registers IRQSOURCESEL and FIQSOURCESEL implement the select signals for these multiplexers. The R/W registers IRQSWSOURCE and FIQSWSOURCE are used to trigger selected software interrupts.

After source selection, the resulting sources are stored in a read-only register, IRQRAWSTATUS. Depending on the status of the mask register, these are used to generate the IRQPENDING read-only register. This is processed to detect the highest priority IRQ request in the list, and the result passed through to the status registers, IRQSTATUS and IRQSTATUSALL. This, in turn is used to generate the relevant IRQ and FIQ outputs.

The IRQSTATUS register provides an encoded index (see Table 5–27) for only the highest priority interrupt which is generating the IRQ output. This provides an easy method for jumping to the appropriate interrupt handler.

5.5.2.4. Programming Guide

The ARM7TDMI processor has two levels of external interrupt, FIQ and IRQ. For an interrupt to be taken, the disable bit in the CPSR must be clear. The CPSR is written to at the beginning of the program code, using specific instructions to enable the interrupts and set the ‘running’ mode of the system. The stack pointer for that particular mode must also be set.

The ARM7TDMI processor documentation contains further, more detailed, information on modes and associated operating conditions.

FIQs have higher priority than IRQs in 2 ways:

1. FIQs are serviced first when multiple interrupts occur. Servicing an FIQ causes IRQs to be disabled, preventing them from being serviced until after the FIQ handler has re-enabled them. This is achieved by restoring the original CPSR from SPSR_FIQ at the end of the handler.

2. For higher execution speed, the FIQ vector is the last entry in the vector table, which allows the FIQ handler to be placed directly at the vector location and run sequentially from that address. This removes the need for a branch and associated delays. There are also banked working registers within the ARM7TDMI in FIQ mode avoiding the need for saving a large number of registers on the stack.

The interrupt controller can accommodate up to 32 interrupt sources, 31 of which will be allocated to the IRQ line. This establishes the need for a specific handler capable of interrupt prioritization. The problem is that after a single exception to IRQ, the CPSR will automatically be altered to disable any further IRQs.

* If nested in this way, the software must use a stack to save the original interrupt position.

The core achieves this by copying the contents of the CPSR into a specific SPSR, in this case SPSR_IRQ. When the routine is finished, the core will then automatically copy the SPSR back into the CPSR, thereby re-enabling the IRQ. A true priority system should permit any new sources of higher priority to interrupt the current exception during the routine.

There is a specific sequence of instructions that must take place in order to ensure that priority encoding is handled correctly. Briefly, this is:

1. IRQ exception takes place. Program branches to IRQ handler from main code.

2. Save the link address (R14) to the stack.

3. Get the Current SPSR (the CPSR will have been automatically copied to the SPSR when the exception took place) and store to stack.

4. Note that IRQ interrupts are currently DISABLED.

5. Determine the source of the interrupt: read status in Interrupt Controller. The IRQ line is automatically cleared by reading the status registers. This allows higher priority interrupts to generate a new IRQ.

6. Now read-modify-write the CPSR to ENABLE interrupts.

7. Perform specific Interrupt Routine.

8. Read-modify-write the CPSR to DISABLE interrupts.

9. Write the index of the interrupt handled to the IRQ_COMPLETE register in Interrupt Controller, to inform it that interrupt has been serviced.

10. Restore SPSR from stack.

11. Jump back to main program.

During step 7, any pending new interrupts of higher priority will break the current service and attend to the new interrupt (see Example 5–1).

Example 5–1: SW Instructions, Assembly Code

```
=========================================
Normal Interrupt response routine (r7 holds Interrupt
Controller Base Address)
=========================================

IrqHandler ROUT

SUB     lr,lr,#4
STMFD   SP!, {lr}               ; Store return address

MRS     r14, SPSR
STMFD   SP!, {r0-r2,r14}        ; Push registers on
                                ; stack

LDR     r1,[r7,#IRQSTATUS]      ; Read status in
                                ;integer format
                                ;this also clears down
                                ;the IRQ line

  ; read-modify-write CPSR to enable interrupts
MRS     r0, CPSR                ; Get CPSR
BIC     r0, r0, #0x80           ; Clear the I bit
MSR     CPSR_c, r0              ; Write back to enable
                                ; Interrupt

=========================================
Perform particular routine (Source is stored in r1)
Switch to SYS state if lr register is to be used with C-rou-
tines.
=========================================

;  Read-Modify-Write CPSR to disable interrupts
 MRS     r0, CPSR               ; Get CPSR
 ORR     r0, r0, #0x80          ; Set the I bit
 MSR     CPSR_c, r0             ; Write back to disable
                                ; Interrupt

;  Interrupt END
STR r1,[r7,#IRQ_COMP]           ; Write back index of
                                ; interrupt just
                                ; handled

LDMFD  SP!, {r0-r2,r14}         ; Pop registers from
                                ; stack
MSR    SPSR, r14
LDMFD  SP!, {pc}^               ; Return address
                                ; stored in pc

;*********************************************
```

5.5.2.5. Interrupt Controller Register Definitions

The Interrupt Controller is programmed through the ARM7TDMI processor APB interface. The Register Map is given in Table 5–28.

Addresses are offsets to the interrupt controller base address, $001C.0000_{hex}$.

The registers are fully described in Table 5–29 to Table 5–47.

Table 5–28: Interrupt Controller Register Map

Offset	R/W	Width	Name	Default
00_{hex}	R	32	IRQRAWSTATUS	0000.0000_{hex}
04_{hex}	R	32	IRQPENDING	0000.0000_{hex}
08_{hex}	R	6	IRQSTATUS	0000.0000_{hex}
$0C_{hex}$	R	32	IRQSTATUSALL	0000.0000_{hex}
10_{hex}	W	32	IRQMASKSET	N/A
14_{hex}	W	32	IRQMASKCLEAR	N/A
18_{hex}	R	32	IRQMASK	$FFFF.FFFF_{hex}$
$1C_{hex}$	R	1	FIQRAWSTATUS	0000.0000_{hex}
20_{hex}	R	1	FIQSTATUS	0000.0000_{hex}
24_{hex}	W	1	FIQMASKSET	N/A
28_{hex}	W	1	FIQMASKCLEAR	N/A
$2C_{hex}$	R	1	FIQMASK	1_{hex}
30_{hex}	W	6	FIQSOURCESEL	00_{hex}
34_{hex}	W	32	IRQSOURCESEL	0000.0000_{hex}
38_{hex}	R/W	1	FIQSWSOURCE	0_{hex}
$3C_{hex}$	R/W	32	IRQSWSOURCE	0000.0000_{hex}
40_{hex}	R/W	5	INT0TYPE	02_{hex}
44_{hex}	R/W	5	INT1TYPE	02_{hex}
48_{hex}	R/W	5	INT2TYPE	02_{hex}
$4C_{hex}$	R/W	5	INT3TYPE	02_{hex}
50_{hex}	R/W	5	INT4TYPE	02_{hex}
54_{hex}	R/W	5	INT5TYPE	02_{hex}
58_{hex}	R/W	5	INT6TYPE	02_{hex}
$5C_{hex}$	R/W	5	INT7TYPE	02_{hex}
60_{hex}	R/W	5	INT8TYPE	02_{hex}
64_{hex}	R/W	5	INT9TYPE	02_{hex}
68_{hex}	R/W	5	INT10TYPE	02_{hex}
$6C_{hex}$	R/W	5	INT11TYPE	02_{hex}
70_{hex}	R/W	5	INT12TYPE	02_{hex}
74_{hex}	R/W	5	INT13TYPE	02_{hex}
78_{hex}	R/W	5	INT14TYPE	02_{hex}
$7C_{hex}$	R/W	5	INT15TYPE	02_{hex}

Table 5–28: Interrupt Controller Register Map

Offset	R/W	Width	Name	Default
80_{hex}	R/W	5	INT16TYPE	02_{hex}
84_{hex}	R/W	5	INT17TYPE	02_{hex}
88_{hex}	R/W	5	INT18TYPE	02_{hex}
$8C_{hex}$	R/W	5	INT19TYPE	02_{hex}
90_{hex}	R/W	5	INT20TYPE	02_{hex}
94_{hex}	R/W	5	INT21TYPE	02_{hex}
98_{hex}	R/W	5	INT22TYPE	02_{hex}
$9C_{hex}$	R/W	5	INT23TYPE	02_{hex}
$A0_{hex}$	R/W	5	INT24TYPE	02_{hex}
$A4_{hex}$	R/W	5	INT25TYPE	02_{hex}
$A8_{hex}$	R/W	5	INT26TYPE	02_{hex}
AC_{hex}	R/W	5	INT27TYPE	02_{hex}
$B0_{hex}$	R/W	5	INT28TYPE	02_{hex}
$B4_{hex}$	R/W	5	INT29TYPE	02_{hex}
$B8_{hex}$	R/W	5	INT30TYPE	02_{hex}
BC_{hex}	R/W	5	INT31TYPE	02_{hex}
$C0_{hex}$	W	5	IRQCOMPLETE	N/A
$C4_{hex}$	W	1	FIQCOMPLETE	N/A
Important: Ensure that the Interrupt Controllers sources are MASKED before the type registers are altered. **Disregarding this may lead to undefined results.**				

Table 5–29: IRQRAWSTATUS Register, Offset 00_{hex} , Reset 0000.0000_{hex}

Bit	31:0
Field	Valid Source
Each source is validated according to its current type.	

Table 5–30: IRQPENDING Register, Offset 04_{hex} , Reset 0000.0000_{hex}

Bit	31:0
Field	Unmasked Valid Source
logical 'AND' IRQ Raw Status and IRQ Mask.	

Table 5–31: IRQSTATUS Register, Offset 08_{hex} , Reset 0000.0000_{hex}

Bit	31:6	5:0
Field	Reserved	Current IRQ Service Status
This register states which source is currently being serviced as a pointer. Reading this status register will clear down the IRQ line, to allow other higher priority interrupt to generate an new IRQ.		

Table 5–32: IRQSTATUSALL Register, Offset $0C_{hex}$, Reset 0000.0000_{hex}

Bit	31:0
Field	All Current Serviced Interrupts
This register shows all interrupts that are either being serviced or have been halted following the arrival of a new exception of higher priority. Reading this status register will clear down the IRQ line, to allow other higher priority interrupt to generate an new IRQ.	

Table 5–33: IRQMASKSET Register, Offset 10_{hex} , Reset N/A

Bit	31:0
Field	Set Mask Bits (disable interrupts)
Set bits in the IRQ interrupt mask. 1: Set mask bit 0: No effect.	

Table 5–34: IRQMASKCLEAR Register, Offset 14_{hex} , Reset N/A

Bit	31:0
Field	Clear Mask Bits (enable interrupts)
Clear bits in the IRQ interrupt mask. 1: Clear mask bit. 0: No effect.	

Table 5–35: IRQMASK Register, Offset 18_{hex} , Reset $FFFF.FFFF_{hex}$

Bit	31:0
Field	IRQ Mask Value
Read only register of the current value of the interrupt mask. The Setting and Clearing of the Mask register is achieved using IRQMASKSET and IRQMASKCLEAR registers, respectively.	

Table 5–36: FIQRAWSTATUS Register, Offset $1C_{hex}$, Reset 0000.0000_{hex}

Bit	31:1	0
Field	Reserved	Valid Source
FIQ source is validated according to its current type.		

Table 5–37: FIQSTATUS Register, Offset 20_{hex} , Reset 0000.0000_{hex}

Bit	31:1	0
Field	Reserved	Valid FIQ Source
This register states whether an FIQ interrupt is currently active		

Table 5–38: FIQMASKSET Register, Offset 24_{hex} , Reset N/A

Bit	31:1	0
Field	Reserved	Set FIQMask Bits
Any write sets bits in the FIQ interrupt mask.		

Table 5–39: FIQMASKCLEAR Register, Offset 28_{hex} , Reset N/A

Bit	31:1	0
Field	Reserved	Clear FIQMask Bits
Any write clears bits in the FIQ interrupt mask.		

Table 5–40: FIQMASK Register, Offset $2C_{hex}$, Reset 1_{hex}

Bit	31:1	0
Field	Reserved	FIQ Mask Value
Read only register of the current value of the interrupt mask. The Setting and Clearing of the Mask register is achieved using FIQMASKSET and FIQMASKCLEAR registers, respectively.		

Table 5–41: FIQSOURCESEL Register, Offset 30_{hex} , Reset 00_{hex}

Bit	31:6	5:1	0
Field	Reserved	FIQ Source Select 5 bit Pointer to determine FIQ source	SW/HW Select ‘1’: Source is SW driven. ‘0’: Source is HW driven.

Table 5–42: IRQSOURCESEL Register, Offset 34_{hex} , Reset 0000.0000_{hex}

Bit	31:0
Field	SW/HW Select ‘1’: Source is SW driven. ‘0’: Source is HW driven.

Table 5–43: FIQSWSOURCE Register, Offset 38_{hex} , Reset 0_{hex}

Bit	31:1	0
Field	Reserved	FIQ software interrupt source
If selected by FIQSOURCESEL, this register acts as the FIQ interrupt source.		

Table 5–44: IRQSWSOURCE Register, Offset $3C_{hex}$, Reset 0000.0000_{hex}

Bit	31:0
Field	IRQ software interrupt source
If corresponding bit is set in IRQSOURCESEL, this register acts as the interrupt source. For example: if bit 3 of IRQSOURCESEL is set then setting bit 3 of IRQSWSOURCE will generate an interrupt at source 3.	

Table 5–45: INT*TYPE Register, Offsets 40_{hex} - BC_{hex} , Reset 02_{hex}

Bit	31:7	6:4	3:2	1:0
Field	Reserved	PRIORITY[2:0] One of 8 levels: 0 to 7 Priority level of interrupt, between 0 and 7.	Reserved	Type of interrupt: 00: High Level Detection 01: Low Level Detection 10: Rising Edge Detection 11: Falling Edge Detection

Table 5–46: IRQCOMPLETE Register, Offset $C0_{hex}$, Reset N/A

Bit	31:6	5:0
Field	Reserved	IRQ Complete on Interrupt
Write-back interrupt index, to inform the interrupt controller that the IRQ service is complete. As each interrupt is successfully handled, the corresponding index of that interrupt, as specified in Table 5–27, should be written back to this register to enable pending interrupts of lower priority to be seen.		

Table 5–47: FIQCOMPLETE Register, Offset $C4_{hex}$, Reset N/A

Bit	31:1	0
Field	Reserved	Service Complete
Write to update the interrupt controller when FIQ service is complete.		

5.5.3. UARTs

Two independently configurable UART interfaces are provided.

5.5.3.1. Features

– Offers similar functionality to the industry standard 16C550 UART.

– Supports bit rates up to 115.2 Kbits/second.

– Separate transmit and receive FIFO, each 16 bytes, with maskable level interrupts. The FIFOs can be disabled.

– Programmable baud rate generator, allowing the division of the UART reference clocks, by 32 to 1048576.

– Programmable data size, odd or even parity, stop bit size.

– Automatic handling of start, stop and parity bits.

– False start bit detection.

– Line break generation & detection

– Full support of modem handshaking signals

– Includes IrDA SIR interface, supporting data rates up to 115.2 Kbits/second half duplex, with low power mode.

5.5.3.2. UART functions

The main functional blocks of each of the PUC 303xA's two UARTs is shown in Figure 5–9.

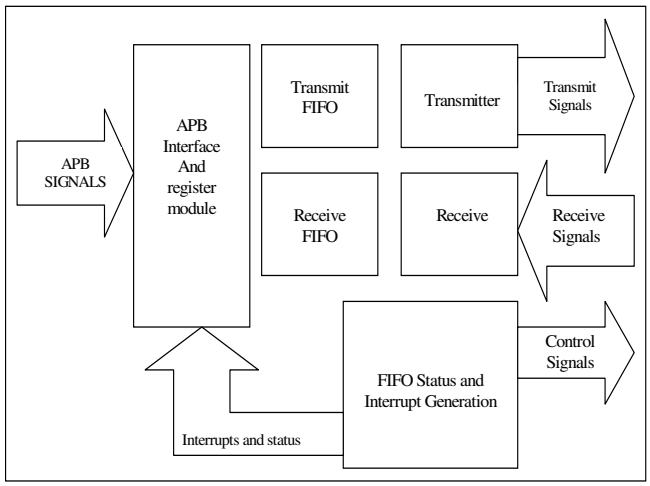


Fig. 5–8: UART Block Diagram

5.5.3.2.1. AMBA APB interface

The AMBA APB interface generates read and write decodes for access to status and control registers, and transmit and receive FIFO memories.

A Register Block stores data written, or to be read across the AMBA APB interface.

5.5.3.2.2. Baud Rate Generator

The baud rate generator incorporates free-running counters that generate the internal x16 clocks, Baud16, and the IrLPBaud16 signal.

Baud 16 furnishes timing information for the UART transmit and receive control. Baud16 comprises a stream of pulses with a width of one UARTCLK clock period, and a frequency of 16 times the baud rate.

IrLPBaud16 is the source of the timing information that is used to generate the pulse width of the IrDA encoded transmit bit stream, when in the low-power mode.

5.5.3.2.3. Transmit FIFO

The transmit and receive paths are buffered with the internal FIFO's, enabling up to 16 bytes to be stored in both receive and transmit data paths. CPU data written across the APB interface is stored in the FIFO until read out by the transmit logic. The transmit FIFO can be disabled to behave as a one-byte holding register.

5.5.3.2.4. Receive FIFO

The Receive FIFO is a 12-bit wide (4-bit flags + 8-bit data), 16-word depth FIFO memory buffer. The receive logic stores received data, and corresponding error bits, in the receive FIFO until the CPU reads it out across the APB interface. The receive FIFO can be disabled to behave as like a one-byte holding register.

5.5.3.2.5. Transmit logic

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. Control logic outputs the serial bit stream beginning with a start bit, data bits (least significant bit (LSB) first) followed by the parity bit, and then stop bits. The precise content of the serial bit stream depends upon the programme configuration in the control registers.

5.5.3.2.6. Receive Logic

The receive logic performs serial-to-parallel conversion on the received bit stream after a valid start pulse has been detected. Overrun, parity, frame error checking, and line break detection are also performed: the data with the associated overrun, parity, framing, and break error bits is written to the receive FIFO.

5.5.3.2.7. Interrupt Generation Logic

The UART generates an interrupt if any of the maskable interrupt conditions are active.

5.5.3.2.8. IrDA Interface

The IrDA interface forms an integral part of both UART modules. The IrDA I/O is multiplexed with the standard UART receive and transmit lines, as shown in Figure 5–9.

The selection of the multiplexing is under the control of an additional register mapped to the UART address space as shown in Table 5–48.

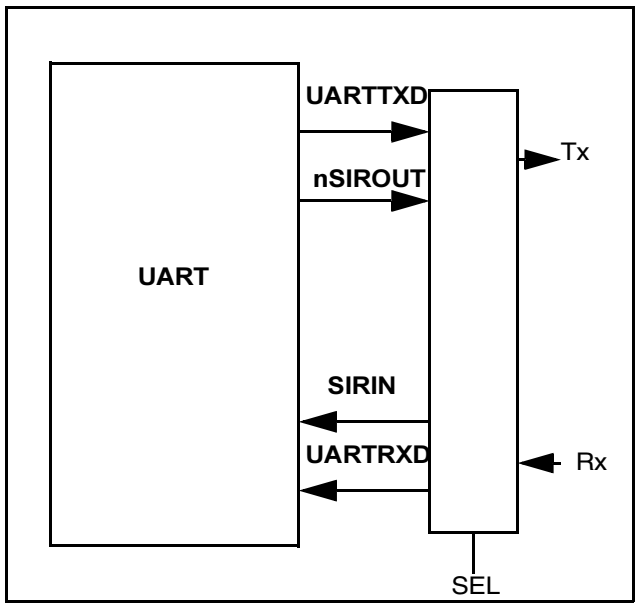


Fig. 5–9: IrDA UART Selection

5.5.3.3. UART Register Map

The locations and brief descriptions of the UART registers are given in Table 5–48.

Addresses shown are relative to the two UART base addresses. UART 1 is located at a base address of $001C.4000_{hex}$ and UART 2 is located at $001C.8000_{hex}$

Table 5–48: UART Register Map[1)]

Address Offset	Type	Width	Reset value	Name	Description
0000_{hex}	R/W	12/8	$---_{hex}$	UARTDR	Data read or written from the interface. It is 12 bits wide on a read, and 8 on a write.
0004_{hex}	R/W	4/0	0_{hex}	UARTRSR/ UARTECR	Receive status register (read)/ Error clear register (write).
0008_{hex} - 0014_{hex}	-	-	-	-	Reserved.
0018_{hex}	R	9	$-10010---_{bin}$	UARTFR	Flag register (read only).
$001C_{hex}$	-	-	-	-	Reserved.
0020_{hex}	R/W	8	00_{hex}	UARTILPR	IrDA low-power counter register.
0024_{hex}	R/W	16	0000_{hex}	UARTIBRD	Integer baud rate divider register.
0028_{hex}	R/W	6	00_{hex}	UARTFBRD	Fractional baud rate divider register.
$002C_{hex}$	R/W	8	00_{hex}	UARTLCR_H	Line control register, HIGH byte.
0030_{hex}	R/W	16	0300_{hex}	UARTCR	Control register.
0034_{hex}	R/W	6	12_{hex}	UARTIFLS	Interrupt FIFO level select register.
0038_{hex}	R/W	11	000_{hex}	UARTIMSC	Interrupt mask set/clear.
$003C_{hex}$	R/W	11	$00-_{hex}$	UARTRIS	Raw interrupt status.
0040_{hex}	R/W	11	$00-_{hex}$	UARTMIS	Masked interrupt status.
0044_{hex}	W	11	N/A	UARTICR	Interrupt clear register.
0048_{hex} - $0FFC_{hex}$	-	-	-	-	Reserved.
1000_{hex}	R/W	1	01_{hex}	UARTMuxSel	IrDA/UART selection register

1) '---' stand for undefined digits after reset

Table 5–49: UARTDR Register, Offset 0000_{hex} , Reset $---_{hex}$

Bit	15:12	11	10	9

Table 5–49: UARTDR Register, Offset 0000_{hex} , Reset $---_{hex}$

Field	Reserved	Overrun Error (OE) This bit is set to 1 if data is received and the receive FIFO is already full. This is cleared to 0 once there is an empty space in the FIFO and a new character can be written to it.	Break Error (BE) This bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits). In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state), and the next valid start bit is received.	Parity Error (PE) When this bit is set to 1, it indicates that the parity of the received data character does not match the parity selected as defined by bits 2 and 7 of the UARTLCR_H register. In FIFO mode, this error is associated with the character at the top of the FIFO.
Bit	8	7:0		
Field	Framing Error (FE) When this bit is set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1). In FIFO mode, this error is associated with the character at the top of the FIFO.	DATA Receive (read) data character. Transmit (write) data character.		

Table 5–50: Combined UARTRSR/UARTECR Register, Offset 0004_{hex}

Write					
Bit	7:0				
Field	Write				
A write to this register clears the framing-, parity-, break-, and overrun errors. The data value is not important.					
Read, Reset 0_{hex}					
Bit	7:4	3	2	1	0
Field	Reserved	Overrun Error (OE)	Break Error (BE)	Parity Error (PE)	Framing Error (FE)
	Reserved, unpredictable when read.	This bit is set to 1 if data is received and the FIFO is already full.	This bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity, and stop bits).	When this bit is set to 1, it indicates that the parity of the received data character does not match the parity selected as defined by bits 2 and 7 of the UARTLCR_H register.	When this bit is set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1). This bit is cleared to 0 by a write to UARTECR.

Table 5–51: UARTFR Register, Offset 0018_{hex} , Reset $-10010---_{bin}$

Bit	15:9	8	7	6	5
Field	Reserved	Ring Indicator (RI)	Transmit FIFO Empty (TXFE)	Receive FIFO Full (RXFF)	Transmit FIFO Full (TXFf)
		This bit is the complement of the UART ring indicator (nUARTRI) modem status input. The bit is 1 when the modem status input is 0.	The meaning of this bit depends on the state of the FEN bit in the UARTLCR_H register. If the FIFO is disabled, this bit is set when the transmit holding register is empty. If the FIFO is enabled, this bit is set when the transmit FIFO is empty.	The meaning of this bit depends on the state of the FEN bit in the UARTLCR_H register. If the FIFO is disabled, this bit is set when the receive holding register is full. If the FIFO is enabled, this bit is set when the receive FIFO is full.	The meaning of this bit depends on the state of the FEN bit in the UARTLCR_H register. If the FIFO is disabled, this bit is set when the transmit holding register is full. If the FIFO is enabled, this bit is set when the transmit FIFO is full.
Bit	4	3	2	1	0
Field	Receive FIFO Empty (RXFE)	UART Busy (BUSY)	Data Carrier Detect (DCD)	Data Set Ready (DSR)	Clear to Send (CTS)
	The meaning of this bit depends on the state of the FEN bit in the UARTLCR_H register. If the FIFO is disabled, this bit is set when the receive holding register is empty. If the FIFO is enabled, this bit is set when the receive FIFO is empty.	If this bit is set to 1, the UART is busy transmitting data. This bit remains set until the complete byte, including all stop bits, has been sent from the shift register. This bit is set as soon as the transmit FIFO becomes non-empty, whether the UART is enabled or not.	This bit is the complement of the PrimeCell UART data carrier detect (nUARTDCD) modem status input. The bit is 1 when the modem status input is 0.	eCell UART data set ready (nUARTDsr) modem status input. The bit is 1 when the modem status input is 0.	eCell UART clear to send (nUARTcts) modem status input. The bit is 1 when the modem status input is 0.

Table 5–52: UARTILPR Register, Offset 0020_{hex} , Reset 00_{hex}

Bit	7:0
Field	Read/Write
	8-bit low-power divider value. These bits are cleared to 0 at reset.

The **IrLPBaud16** signal is generated by dividing down the **UARTCLK** signal according to the low-power divider value written to UARTILPR.

The low-power divider value is calculated as follows:

low-power divider (ILPDVSR) = ($\mathbf{F_{UARTCLK}}$ / $\mathbf{F_{IrLPBaud16}}$)

where $\mathbf{F_{IrLPBaud16}}$ is nominally 1.8432MHz.

Choose the divider so that 1.42MHz < $\mathbf{F_{IrLPBaud16}}$ < 2.12MHz, that results in a low-power pulse duration of 1.41–2.11µs (three times the period of **IrLPBaud16**).

The minimum frequency of **IrLPBaud16** ensures that pulses less than one period of **IrLPBaud16** are rejected, but that pulses greater than 1.4µs are accepted as valid pulses.

Note that programming a zero value results in no **IrLPBaud16** pulses being generated.

Table 5–53: UARTIBRD Integer Baud Rate Register, Offset 0024_{hex} , Reset 0000_{hex}

Bit	15:0
Field	Read/Write
	Integer baud rate divider value. These bits are cleared to 0 at reset.

Table 5–54: UARTIBRD Fractional Baud Rate Register, Offset 0028_{hex} , Reset 00_{hex}

Bit	5:0
Field	Read/Write
	Fractional baud rate divider value. These bits are cleared to 0 at reset.

The baud rate divider is calculated as follows:

Baud rate divider BAUDDIV = ($\mathbf{F_{UARTCLK}}$/ {16 * Baud rate})

where $\mathbf{F_{UARTCLK}}$ is the UART reference clock frequency.

The BAUDDIV is comprised of the integer value (BAUD DIVINT) and the fractional value (BAUD DIVFRAC).

The contents of the UARTIBRD and UARTFBRD registers are not updated until transmission or reception of the current character is complete.

The minimum divide ratio possible is 1 and the maximum is 65535 (2^{16} - 1). That is, UARTIBRD = 0 is invalid and UARTFBRD is ignored when this is the case.

Similarly, when UARTIBRD = 65535 (that is $FFFF_{hex}$), then UARTFBRD must not be greater than zero. If this is exceeded it results in an aborted transmission or reception.

Example 5–1 shows how to calculate the divider value.

Example 5–1: Calculating the divider value

If the required baud rate is 115200 bps and **UARTCLK** = 2 MHz then:

1. Baud Rate Divider = ($2 * 10^6$)/(16 * 115200) = 1.085

2. Therefore, BRD_I = 1 and BRD_F = 0.085,

3. Therefore, the fractional part m = integer((0.085 * 64) + 0.5) = 5

4. Generated baud rate divider = 1 + 5/64 = 1.078

5. Generated baud rate = ($2 * 10^6$)/(16 * 1.078) = 115955

6. Error = (115955 - 115200)/115200 * 100 = 0.656%

7. The maximum error using a 6-bit UARTFBRD register = 1/64 * 100 = 1.56%. This occurs when m = 1, and the error is cumulative over 64 clock ticks.

Table 5–55 shows some typical bit rates and their corresponding dividers, given the UART clock frequency of 7.3728 MHz. These values do not use the fractional divider so the value in the UARTFBRD register is zero.

Table 5–55: Typical Baud Rates and Dividers

Programmed Integer Divider	**Bit Rate (bps)**
4_{hex}	115200
6_{hex}	76800
8_{hex}	57600
C_{hex}	38400
18_{hex}	19200
20_{hex}	14400
30_{hex}	9600
$C0_{hex}$	2400
180_{hex}	1200
$105D_{hex}$	110

Table 5–56 shows some required bit rates and their corresponding integer and fractional divider values and generated bit rates given a clock frequency of 4.0 MHz

.

Table 5–56: Typical Baud Rates and Integer and Fractional Dividers for UARTCLK = 4 MHz

Programmed divider (integer)	Programmed divider (fraction)	Required bit rate (bps)	Generated bit rate (bps)	Error (%)
2_{hex}	B_{hex}	115200	115101	0.086
3_{hex}	10_{hex}	76800	76923	0.160
6_{hex}	21_{hex}	38400	38369	0.081
11_{hex}	17_{hex}	14400	14401	0.007
68_{hex}	B_{hex}	2400	2400	~0
$8E0_{hex}$	$2F_{hex}$	110	110	~0

Table 5–57: UARTLCR_H Register, Offset $002C_{hex}$, Reset 00_{hex}

Bit	15:8	7	6:5	4
Field	Reserved	Stick Parity Select (SPS) When bits 1,2 and 7 of the UARTLCR_H register are set, the parity bit is transmitted and checked as a 0. When bits 1 and 7 are set, and bit 2 is 0, the parity bit is transmitted and checked as a 1. When this bit is cleared stick parity is disabled.	Word length [1:0] (WLEN) The select bits indicate the number of data bits transmitted or received in a frame as follows: 11 = 8 bits 10 = 7 bits 01 = 6 bits 00 = 5 bits.	Enable FIFOs (FEN) If this bit is set to 1, transmit and receive FIFO buffers are enabled (FIFO mode). When cleared to 0 the FIFOs are disabled (character mode) that is, the FIFOs become 1-byte-deep holding registers.
Bit	3	2	1	0
Field	Two Stop Bits Select (STP2) If this bit is set to 1, two stop bits are transmitted at the end of the frame. The receive logic does not check for two stop bits being received.	Even Parity Select (EPS) If this bit is set to 1, even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits. When cleared to 0 then odd parity is performed which checks for an odd number of 1s. This bit has no effect when parity is disabled by **Parity Enable** (bit 1) being cleared to 0.	Parity Enable (PEN) If this bit is set to 1, parity checking and generation is enabled, else parity is disabled and no parity bit added to the data frame.	Send Break (BRK) If this bit is set to 1, a low-level is continually output on the UARTTXD output, after completing transmission of the current character. This bit must be asserted for at least one complete frame transmission time in order to generate a break condition. The transmit FIFO contents remain unaffected during a break condition. For normal use, this bit must be cleared to 0.

UARTLCR_H, UARTIBRD and UARTFBRD form a single 30-bit wide register (UARTLCR) which is updated on a single write strobe generated by a UARTLCR_H write. So, in order to internally update the contents of UARTIBRD or UARTFBRD, a UARTLCR_H write must always be performed at the end.

To update the three registers there are two possible sequences:

1. UARTIBRD write, UARTFBRD write and UARTLCR_H write

2. UARTFBRD write, UARTIBRD write and UARTLCR_H write.

To update UARTIBRD or UARTFBRD only: UARTIBRD write (or UARTFBRD write) and UARTLCR_H write.

Table 5–58 is a truth table for the SPS, EPS, and PEN bits of the UARTLCR_H register.

Table 5–58: Truth Table

Parity Enable (PEN)	Even Parity Select (EPS)	Stick Parity Select (SPS)	Parity bit (transmitted or checked)
0	x	x	Not transmitted or checked
1	1	0	Even parity
1	0	0	Odd parity
1	0	1	1
1	1	1	0

The contents of the UARTLCR_H register are not updated until transmission or reception of the current character is complete.

The UARTCR register is the control register. All bits are cleared to 0 on reset, except for bits 9 and 8 which are set to 1. The table below shows the bit assignment of the UARTCR register.

Table 5–59: UARTCR Register, Offset 0030_{hex} , Reset 0300_{hex}

Bit	15	14	13	12
Field	CTS Hardware Flow Control Enable (CTSEn) If this bit is set to 1, CTS hardware flow control is enabled. Data is only transmitted when the nUARTCTS signal is asserted.	RTS Hardware Flow Control Enable(RTSEn) If this bit is set to 1, RTS hardware flow control is enabled. Data is only requested when there is space in the receive FIFO for it to be received.	Reserved	Reserved
Bit	11	10	9	8
Field	Request to Send (RTS) This bit is the complement of the PrimeCell UART request to send (nUARTRTS) modem status output. That is, when the bit is programmed to a 1, the output is 0.	Data Transmit Ready (DTR) This bit is the complement of the PrimeCell UART data transmit ready (nUARTDTR) modem status output. That is, when the bit is programmed to a 1, the output is 0.	Receive Enable (RXE) If this bit is set to 1, the receive section of the PrimeCell UART is enabled. Data reception occurs for either PrimeCell UART signals or SIR signals according to the setting of SIR Enable (bit 1). When the PrimeCell UART is disabled in the middle of reception, it completes the current character before stopping.	Transmit Enable (TXE) If this bit is set to 1, the transmit section of the PrimeCell UART is enabled. Data transmission occurs for either PrimeCell UART signals, or SIR signals according to the setting of SIR Enable (bit 1). When the PrimeCell UART is disabled in the middle of transmission, it completes the current character before stopping.

Table 5–59: UARTCR Register, Offset 0030_{hex} , Reset 0300_{hex}

Bit	7	6:3	2	1
Field	Loop Back Enable (LBE) If this bit is set to 1 and the SIR Enable bit is set to 1 and the test register UARTTCR bit 2 (SIRTEST) is set to 1, then the nSIROUT path is inverted, and fed through to the SIRIN path. The SIRTEST bit in the test register must be set to 1 to override the normal half-duplex SIR operation. This must be the requirement for accessing the test registers during normal operation, and SIRTEST must be cleared to 0 when loopback testing is finished.This feature reduces the amount of external coupling required during system test. If this bit is set to 1, and the SIRTEST bit is set to 0, the UARTTXD path is fed through to the UARTRXD path. In either SIR mode or normal mode, when this bit is set, the modem outputs are also fed through to the modem inputs. This bit is cleared to 0 on reset, which disables the loopback mode.	Reserved	IrDA SIR Low Power Mode (SIRLP) This bit selects the IrDA encoding mode. If this bit is cleared to 0, low-level bits are transmitted as an active high pulse with a width of $^3/_{16}$th of the bit period. If this bit is set to 1, low-level bits are transmitted with a pulse width which is 3 times the period of the **IrLPBaud16** input signal, regardless of the selected bit rate. Setting this bit uses less power, but might reduce transmission distances.	SIR Enable (SIREN) If this bit is set to 1, the IrDA SIR ENDEC is enabled. This bit has no effect if the UART is not enabled by bit 0 being set to 1. When the IrDA SIR ENDEC is enabled, data is transmitted and received on nSIROUT and SIRIN. UARTTXD remains in the marking state (set to 1). Signal transitions on UARTRXD or modem status inputs have no effect. When the IrDA SIR ENDEC is disabled, nSIROUT remains cleared to 0 (no light pulse generated), and signal transitions on SIRIN have no effect.
Bit	0			
Field	UART Enable (UARTEN) If this bit is set to 1, the ARM7TDMI PrimeCell UART is enabled. Data transmission and reception occurs for either PrimeCell UART signals or SIR signals according to the setting of SIR Enable (bit 1). When the PrimeCell UART is disabled in the middle of transmission or reception, it completes the current character before stopping.			

The UARTIFLS register is the interrupt FIFO level select register. The UARTIFLS register can be used to define the FIFO level at which the **UARTTXINTR** and **UARTRXINTR** are triggered.

The interrupts are generated based on a transition through a level rather than being based on the level. That is, the design is such that the interrupts are generated when the fill level progresses through the trigger level.

The bits are reset so that the trigger level is when the FIFOs are at the half-way mark. Table 5–60 shows the bit assignment of the UARTIFLS register.

Table 5–60: UARTIFLS Register, Offset 0034_{hex} , Reset 12_{hex}

Bit	15:6	5:3	2:0
Field	Reserved	Receive Interrupt FIFO Level Select (RXIFLSEL) The trigger points for the receive interrupt are as follows: 000 = Receive FIFO becomes >= 1/8 full 001 = Receive FIFO becomes >= 1/4 full 010 = Receive FIFO becomes >= 1/2 full 011 = Receive FIFO becomes >= 3/4 full 100 = Receive FIFO becomes >= 7/8 full 101:111 = reserved.	Transmit Interrupt FIFO Level Select (TXIFLSEL) The trigger points for the transmit interrupt are as follows: 000 = Transmit FIFO becomes <= 1/8 full 001 = Transmit FIFO becomes <= 1/4 full 010 = Transmit FIFO becomes <= 1/2 full 011 = Transmit FIFO becomes <= 3/4 full 100 = Transmit FIFO becomes <= 7/8 full 101:111 = reserved.

The UARTIMSC register is the interrupt mask set/clear register. It is a read/write register.

On a read this register gives the current value of the mask on the relevant interrupt. On a write of 1 to the particular bit, it sets the corresponding mask of that interrupt. A write of 0 clears the corresponding mask.

All of the bits are cleared to 0 when reset. Table 5–61 shows the bit assignment of the UARTIMSC register.

.
Table 5–61: UARTIMSC Register, Offset 0038_{hex} , Reset 000_{hex}

Bit	15:11	10	9	8
Field	Reserved	Overload Error Interrupt Mask (OEIM) On a read, the current mask for the OEIM interrupt is returned. On a write of 1, the mask of the OEIM interrupt is set. A write of 0 clears the mask.	Break Error Interrupt Mask (BEIM) On a read the current mask for the BEIM interrupt is returned. On a write of 1, the mask of the BEIM interrupt is set. A write of 0 clears the mask.	Parity Error Interrupt Mask (PEIM) On a read the current mask for the PEIM interrupt is returned. On a write of 1, the mask of the PEIM interrupt is set. A write of 0 clears the mask.
Bit	7	6	5	4
Field	Framing Error Interrupt Mask (FEIM) On a read the current mask for the FEIM interrupt is returned. On a write of 1, the mask of the FEIM interrupt is set. A write of 0 clears the mask.	Receive Timeout Interrupt Mask (RTIM) On a read the current mask for the RTIM interrupt is returned. On a write of 1, the mask of the RTIM interrupt is set. A write of 0 clears the mask.	Transmit Interrupt Mask (TXIM) IOn a read the current mask for the TXIM interrupt is returned. On a write of 1, the mask of the TXIM interrupt is set. A write of 0 clears the mask.	Receive Interrupt Mask (RXIM) On a read the current mask for the RXIM interrupt is returned. On a write of 1, the mask of the RXIM interrupt is set. A write of 0 clears the mask.
Bit	3	2	1	0
Field	nUARTDSR Modem Interrupt Mask (DSRMIM) On a read the current mask for the DSRMIM interrupt is returned. On a write of 1, the mask of the DSRMIM interrupt is set. A write of 0 clears the mask.	nUARTDCD Modem Interrupt Mask (DCDMIM) On a read the current mask for the DCDMIM interrupt is returned. On a write of 1, the mask of the DCDMIM interrupt is set. A write of 0 clears the mask.	nUARTCTS Modem Interrupt Mask (CTSMIM) On a read the current mask for the CTSMIM interrupt is returned. On a write of 1, the mask of the CTSMIM interrupt is set. A write of 0 clears the mask.	nUARTRI Modem Interrupt Mask (RIMIM) On a read the current mask for the RIMIM interrupt is returned. On a write of 1, the mask of the RIMIM interrupt is set. A write of 0 clears the mask.

The UARTRIS register is the raw interrupt status register. It is a read-only register. On a read this register gives the current raw status value of the corresponding interrupt. A write has no effect.

All the bits, are cleared to 0 when reset, except for the modem status interrupt bits, 3 to 0. The modem status interrupt bits are undefined after reset.

Table 5–62 shows the bit assignment of the UARTRIS register.

Table 5–62: UARTRIS Register, Offset $003C_{hex}$, Reset $00-_{hex}$

Bit	15:11	10	9	8
Field	Reserved	Overrun Error Interrupt Status (OERIS) Gives the raw interrupt state (prior to masking) of the UARTOEINTR interrupt	Break Error Interrupt Status (BERIS) Gives the raw interrupt state (prior to masking) of the UARTBEINTR interrupt	Parity Error Interrupt Status (PERIS) Gives the raw interrupt state (prior to masking) of the UARTPEINTR interrupt
Bit	7	6	5	4
Field	Framing Error Interrupt Status (FERIS Gives the raw interrupt state (prior to masking) of the UARTFEINTR interrupt	Receive Timeout Interrupt Status (RTRIS) Gives the raw interrupt state (prior to masking) of the UARTRTINTR interrupt[1)]	Transmit Interrupt Status (TXRIS) IGives the raw interrupt state (prior to masking) of the UARTTXINTR interrupt	Receive Interrupt Status (RXRIS) Gives the raw interrupt state (prior to masking) of the UARTRXINTR interrupt
Bit	3	2	1	0
Field	nUARTDSR Modem Interrupt Status (DSRRMIS) Gives the raw interrupt state (prior to masking) of the UARTDSRINTR interrupt	nUARTDCD Modem Interrupt Status (DCDRMIS) Gives the raw interrupt state (prior to masking) of the UARTDCDINTR interrupt	nUARTCTS Modem Interrupt Status (CTSRMIS) Gives the raw interrupt state (prior to masking) of the UARTCTSINTR interrupt	nUARTRI Modem Interrupt Status (RIRMIS) Gives the raw interrupt state (prior to masking) of the UARTRIINTR interrupt

1) In this case the raw interrupt cannot be set unless the mask is set, this is because the mask acts as an enable for power saving. That is, the same status can be read from UARTMIS and UARTRIS for the receive timeout interrupt.

The UARTMIS register is the masked interrupt status register. It is a read-only register. On a read this register gives the current masked status value of the corresponding interrupt. A write has no effect.

All of the bits are cleared to 0 when reset, except for the modem status interrupt bits 3 to 0. The modem status interrupt bits are undefined after reset. Table 5–63 shows the bit assignment of the UARTMIS register.

Table 5–63: UARTMIS Register, Offset 0040_{hex} , Reset $00-_{hex}$

Bit	15:11	10	9	8
Field	Reserved	Overrun Error Masked Interrupt Status (OEMIS) Gives the masked interrupt state (after masking) of the UARTOEINTR interrupt	Break Error Masked Interrupt Status (BEMIS) Gives the masked interrupt state (after masking) of the UARTBEINTR interrupt	Parity Error Masked Interrupt Status (PEMIS) Gives the masked interrupt state (after masking) of the UARTPEINTR interrupt
Bit	7	6	5	4
Field	Framing Error Masked Interrupt Status (FEMIS) Gives the masked interrupt state (after masking) of the UARTFEINTR interrupt	Receive Timeout Masked Interrupt Status (RTMIS) Gives the masked interrupt state (after masking) of the UARTRTINTR interrupt	Transmit Masked Interrupt Status (TXMIS) IGives the masked interrupt state (after masking) of the UARTTXINTR interrupt	Receive Masked Interrupt Status (RXMIS) Gives the masked interrupt state (after masking) of the UARTRXINTR interrupt
Bit	3	2	1	0
Field	nUARTDSR Modem Masked Interrupt Status (DSRMMIS) Gives the masked interrupt state (after masking) of the UARTDSRINTR interrupt	nUARTDCD Modem Masked Interrupt Status (DCDMMIS) Gives the masked interrupt state (after masking) of the UARTDCDINTR interrupt	nUARTCTS Modem Masked Interrupt Status (CTSMMIS) Gives the masked interrupt state (after masking) of the UARTCTSINTR interrupt	nUARTRI Modem Masked Interrupt Status (RIMMIS) Gives the masked interrupt state (after masking) of the UARTRIINTR interrupt

Table 5–64: UARTICR Register, Offset 0044_{hex} , Reset N/A

Bit	15:11	10	9	8
Field	Reserved	Overrun Error Interrupt Clear (OEIC) Clears the UARTOEINTR interrupt	Break Error Interrupt Clear (BEIC) Clears the UARTBEINTR interrupt	Parity Error Interrupt Clear (PEIC) Clears the UARTPEINTR interrupt
Bit	7	6	5	4
Field	Framing Error Interrupt Clear (FEIC) Clears the UARTFEINTR interrupt	Receive Timeout Interrupt Clear (RTIC) Clears the UARTRTINTR interrupt	Transmit Interrupt Clear (TXIC) IClears the UARTTXINTR interrupt	Receive Interrupt Clear (RXIC) Clears the UARTRXINTR interrupt
Bit	3	2	1	0
Field	nUARTDSR Modem Interrupt Clear (DSRMIC) Clears the UARTDSRINTR interrupt	nUARTDCD Modem Interrupt Clear (DCDMIC) Clears the UARTDCDINTR interrupt	nUARTCTS Modem Interrupt Clear (CTSMIC) Clears the UARTCTSINTR interrupt	nUARTRI Modem Interrupt Clear Status (RIMIC) Clears the UARTRIINTR interrupt

Table 5–65: UARTMuxSel Register, Offset 1000_{hex} , Reset 01_{hex}

Bit	15:1	0
Field	Reserved	Multiplexer Select A value of 1 selects UART, a value of 0 selects IrDA

5.5.4. Synchronous Serial Ports

The PUC 303xA device provides five independent Synchronous Serial Ports (SSP).

5.5.4.1. SSP Peripheral Operation

The general structure of each of the five SSP peripheral blocks available on PUC 303xA is shown in Figure 5–10.

5.5.4.1.1. Interface Signals

The transmit and receive signals consist of the following signals:

– sclk(x), serial clock

– sfrm(x), serial data framing signal

– stxd(x), serial transmit data

– srxd(x), serial receive data

Note, that the interface for SSP5 is actually the I^2S interface.

5.5.4.1.2. Transmit and Receive Logic

When the SSP is configured as a master, the serial clock to the connected slave is derived from a divided version of the ssp(x)clk. The master transmit logic reads a value from the transmit FIFO and carries out a parallel to serial conversion on it. The serial data and frame logic control signals are synchronized to sclk(x), and then output to the externally connected slave through the stxd(x) and sfrm(x) pins respectively.

Simultaneously, the master receive logic performs serial to parallel conversion on the incoming synchronous srxd(x) data stream, extracting and storing values into the receive FIFO, for subsequent reading through the APB interface.

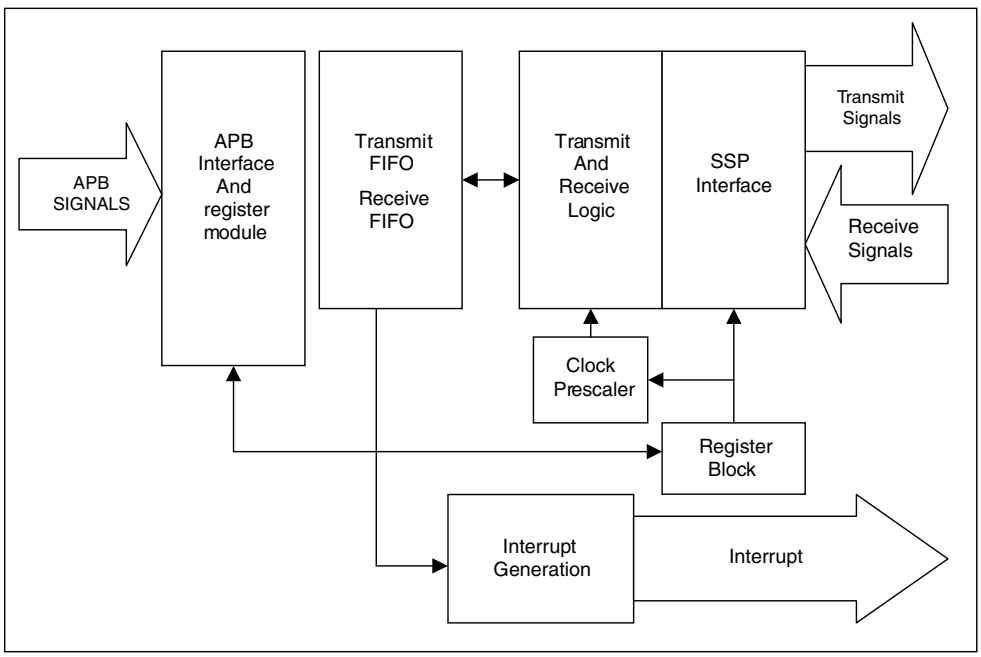


Fig. 5–10: General SSP Structure

When configured as a slave, both the sclk(x) and sfrm(x) signals are provided by an attached master and used to time its transmission and reception sequence.

The slave transmit logic, under control of the master clock, will successively read a value from the transmit FIFO, performs parallel to serial conversion, then output the serial data stream through the slave stxd(x) pin.

The master receive logic performs serial to parallel conversion on the incoming srxd(x) data stream, extracting and storing values into its receive FIFO, for subsequent reading through the APB interface.

Depending on the operating mode selected, the sfrm(x) output operates as an active HIGH frame synchronization output for Texas Instruments synchronous serial interface format or an active LOW slave select for the Motorola SPI and National Semiconductor Microwire modes.

5.5.4.2. SSP modes

The description below applies to all of the five SSPs in the PUC 303xA, where '(x)' in the signal names is between 1 and 5, signifying the SSP number.

All of the five SSP peripherals can be configured to perform both master and slave operation in any of five available protocols. Each SSP peripheral comprises a core SSP peripheral and surrounding sequential logic, termed 'SSP Interface', for protocol conversion purposes.

The core SSP peripheral supports the following three primary protocols:

– Motorola SPI Mode

– National Semiconductor Microwire Mode

– Texas Instruments Synchronous Serial Interface Mode

In addition two secondary protocols are supported by means of the SSP interface that 'wraps' the core SSP peripheral:

– PUC Word mode (emulates I^2S)

– PUC Continuous mode

A more detailed description of these secondary modes can be found in Section 5.5.4.4.

In both, master and slave configurations, the SSP peripherals perform the following:

– Parallel-to-serial conversion on data written to an internal 16-bit wide, 8 location deep transmit FIFO.

– Serial-to-parallel conversion on the received data, buffering it in a similar 16-bit wide, 8-location deep receive FIFO.

The SSPs can operate with serial clocks up to 24 MHz, depending on whether the SSP is configured as a master or slave and whether the data is receive only.

The logical speed constraints are summarised below in terms of the derived SSP clocks from the clock generator, ssp(x)clk, (see Section 5.2. on page 8):

– Master duplex with serial clock up to ssp(x)clk/2

– Slave receive-only with serial clock up to ssp(x)clk/3

– Slave duplex with serial clock up to ssp(x)clk/12

5.5.4.3. SSP Primary Modes

When used in Texas Instruments SSI, Motorola SPI and National Semiconductor Microwire modes, the wrapping SSP interface that emulates the secondary protocols must be configured into 'Transparent' mode.

In this mode, the SSP interface logic feeds the relevant signals from the core SSP peripheral straight through, allowing direct communication with an external device based on the primary protocols.

Transparent mode is entered by writing to the relevant bits of the MODE_SELECT register. The DATA_SIZE register does not need to be set (see Section Table 5–66: on page 56).

For primary protocols, the SSP peripheral should be set up in accordance with the instructions contained in the register programming tables described below.

5.5.4.4. SSP Secondary Modes

In addition to the primary protocols, it is possible to configure any of the SSP interfaces to support either of the two secondary protocols:

– PUC Word mode (emulates I^2S)

– PUC Continuous mode

These modes have been included to ensure seamless interfacing to the Micronas audio devices (e.g. MAS 35xxF family).

To facilitate these secondary modes, the SSP peripherals provide programmable mode selection and data size, to be specified in the registers MODE_SELECT and DATA_SIZE (refer to Table 5–66).

The MODE_SELECT register allows the PUC Word or PUC Continous modes to be selected for 'full duplex' or 'receive only' SSPs. The benefit of the 'receive only' mode is the higher available serial clock rate of ssp(x)clk/3 as opposed to ssp(x)clk/12, when the PUC 303xA is acting as an SSP slave interface.

For the special case of SSP5 the ssp5clk source may be fed with an external clock, such that a multiple of an accurate audio clock (MAS 35x9F/MAS 3587F CLKO) can be used to transmit samples as I^2S audio streams. This is controlled by bit[3] in the MODE_SELECT register.

5.5.4.4.1. PUC Continuous mode

The PUC Continuous mode converts a Motorola SPI format to MAS 35x9F/MAS 3587F continuous format, in both master and slave modes.

The core SSP peripheral should be configured for Motorola SPI mode.

This mode is set by writing the appropriate value to the MODE_SELECT register. Master mode will be selected if the corresponding bit in the SSPCR1 register has been set to master mode.

For PUC Continuous mode the PUC 303xA can support data sizes of 16, 20, 24 and 32 bit words, by programming the DATA_SIZE register.

It is possible to activate this mode with either unidirectional or duplex communication.

5.5.4.4.2. PUC Word mode

The PUC Word mode converts a Texas Instruments Synchronous Serial Interface format to MAS 35x9F/ MAS 3587F word mode format. It is possible to activate this mode in either master or slave modes with unidirectional or duplex communication.
The core SSP peripheral must be configured for TI SSI mode.

For a transmitting SSP peripheral, the SSP interface circuitry generates a word framed protocol that emulates I^2S frames with an optional 1-bit delay on the data line. When operating in PUC Word mode as a receiver, the SSP interface will translate a non-delayed I^2S protocol into the required TI SSI mode for the receiving core SSP peripheral.

In this mode information must be written to the DATA_SIZE register to let the SSP's interface logic discriminate between data transfers of 16 bits and transfers of either 20, 24 or 32 bits, so the correct framing signal can be generated. Additionally the core SSP peripheral must be set up for the proper data size in the SSPCR0 register as follows:

– For word width of 16 bits, the data size for the core SSP peripheral is also configured as 16 bits.

– For word width of 32 bits, the data size for the core SSP peripheral is configured as 16 bits.

– For word width of 20 bits, the data size for the core SSP peripheral is configured as 10 bits.

– For word width of 24 bits, the data size for the core SSP peripheral is configured as 12 bits.

An external master may send a low impulse on sfrm(x) of length one bit-cycle during bit[1] of the last transferred data word as an end condition. This is ignored by the SSP interface, i.e. this low impulse has no resynchronisation effect.

5.5.4.5. Example SSP communication waveforms

Fig. 5–11: Transmission of a single frame in PUC Word Mode:

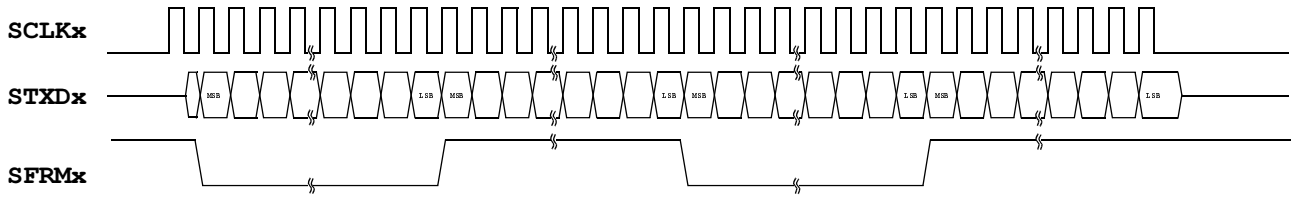


Fig. 5–12: Full duplex communication in PUC Word Mode:

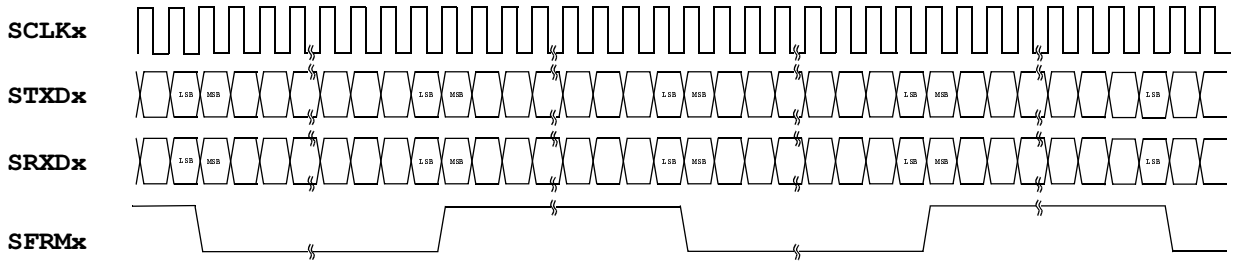


Fig. 5–13: Transmission of I^2S stream in PUC Word Mode with STXDx delayed:

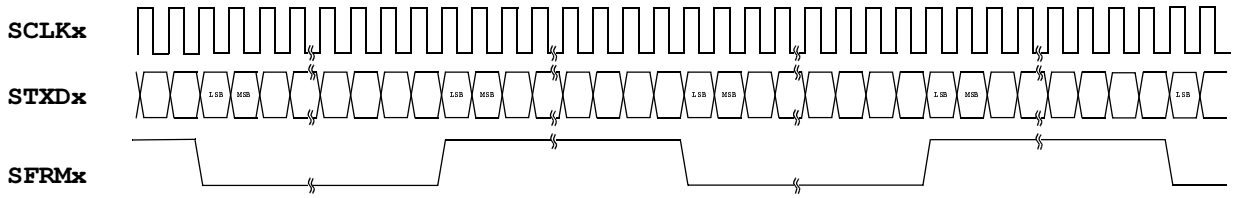


Fig. 5–14: Transmission of single word (16 bits) in Motorola SPI Mode with SPH=0; SPO=0/1:

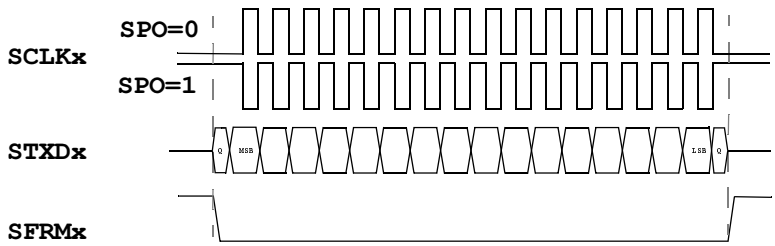


Fig. 5–15: Transmission of single word (16 bits) in Motorola SPI Mode with SPH=1; SPO=0/1:

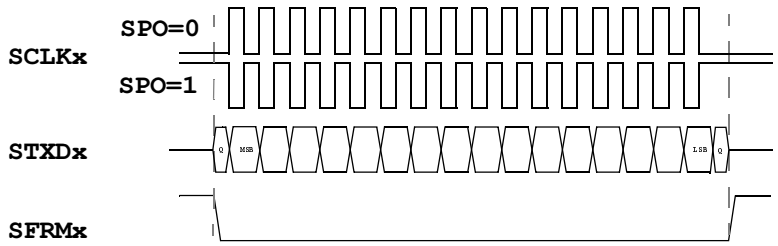


Fig. 5–16: Transmission of continuous stream in Motorola SPI Mode (SPH=1):

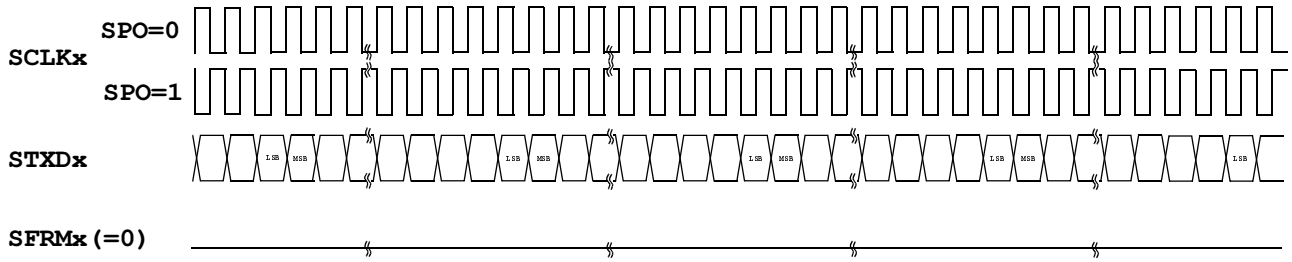


Fig. 5–17: Transmission of single word (16 bits) in Texas Instruments Mode:

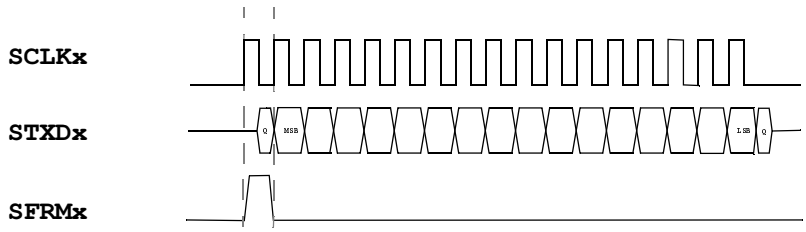
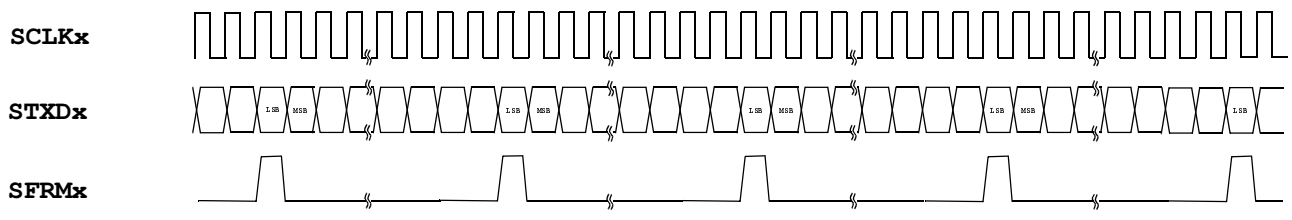


Fig. 5–18: Transmission of continuous stream in Texas Instruments Mode:



5.5.4.6. SSP Register Descriptions

Table 5–66 describes the SSP register map. Each of the five SSP peripherals within the PUC 303xA contains the same set of registers.

The addresses shown are an offset relative to each SSP base address. These are as follows:

- SSP 1: $001D.0000_{hex}$
- SSP 2: $001D.4000_{hex}$
- SSP 3: $001D.8000_{hex}$
- SSP 4: $001D.C000_{hex}$
- SSP 5: $001E.0000_{hex}$

Table 5–66: SSP Register Map

Offset	Access	Width	Name	Description	Default
00_{hex}	R/W	16	SSPCR0	Control Register 0	00_{hex}
04_{hex}	R/W	6	SSPCR1	Control Register 1	00_{hex}
08_{hex}	R/W	16	SSPDR	RX FIFO (read) / TX FIFO (write)	$----_{hex}$
$0C_{hex}$	R	5	SSPSR	Status Register	00_{hex}
10_{hex}	R/W	8	SSPCPSR	Clock Prescale Register	00_{hex}
14_{hex}	R/W	3/16	SSPIIR/SSPICR	Interrupt Status / Clear Register	00_{hex}
18_{hex}	R/W	4	MODE_SELECT	Mode Select Register	02_{hex}
$1C_{hex}$	R/W	2	DATA_SIZE	Data Size Register	00_{hex}

Table 5–67: SSPCR0 Register, Offset 00_{hex}, Reset 00_{hex}

Bit	31:16	15:8	7	6	5:4	3:0
Field	Reserved	**SCR**	**SPH**	**SPO**	**FRF**	**DSS**
This register controls various functions within the SSP peripheral						

Bit definitions for SSPCR0 register:

Bits	Name	Type	Function
15:8	**SCR**	Read/write	Serial clock rate division factor. The value SCR is used to generate the transmit and receive bit rate of the PrimeCell SSPMS. The bit rate is: $\frac{F_{SSPCLK}}{CPSDVSR \times (1 + SCR)}$ where CPSDVSR is an even value from 2 to 254, programmed via SSPCPSR register and SCR is a value from 0 to 255.
7	**SPH**	Read/write	**SCLKOUT** phase (applicable to Motorola SPI frame format only).
6	**SPO**	Read/write	**SCLKOUT** polarity (applicable to Motorola SPI frame format only).
5:4	**FRF**	Read/write	Frame format: 00 Motorola SPI frame format 01 TI synchronous serial frame format 10 National Microwire frame format 11 Reserved, undefined operation
3:0	**DSS**	Read/write	Data Size Select: 0000 Reserved, undefined operation 0001 Reserved, undefined operation 0010 Reserved, undefined operation 0011 4-bit data 0100 5-bit data 0101 6-bit data 0110 7-bit data 0111 8-bit data 1000 9-bit data 1001 10-bit data 1010 11-bit data 1011 12-bit data 1100 13-bit data 1101 14-bit data 1110 15-bit data 1111 16-bit data.

Table 5–68: SSPCR1 register, Offset 04_{hex}, Reset 00_{hex}

Bit	31:6	5	4	3	2	1	0
Field	Reserved	**MS**	**SSE**	**LBM**	**RORIE**	**TIE**	**RIE**
This register controls various functions within the SSP peripheral							

Bit definitions for SSPCR1 register:

Bits	Name	Type	Function
31:7	-	-	Reserved, read unpredictable, should be written as 0.
6	-	-	Reserved, must be written as 0.
5	**MS**	Read/write	Master/slave mode select. This bit can be modified only when the SSP is disabled (SSE=0). 0 = Device configured as master (default). 1 = Device configured as slave.
4	**SSE**	Read/write	Synchronous serial port enable: 0 = SSP operation disabled. 1 = SSP operation enabled.
3	**LBM**	Read/write	Loop back mode: 0 = Normal serial port operation enabled. 1 = Output of transmit serial shifter is connected to input of receive serial shifter internally.
2	**RORIE**	Read/write	Receive FIFO overrun interrupt enable: 0 = Overrun detection is disabled. Overrun condition does not generate the **SSPRORINTR** interrupt. Clearing this bit to zero also clears the **SSPRORINTR** interrupt if it is already asserted. 1 = Overrun detection is enabled. Overrun condition generates the **SSPRORINTR** interrupt.
1	**TIE**	Read/write	Transmit FIFO interrupt enable: 0 = Transmit FIFO half-full or less condition does not generate the **SSPTXINTR** interrupt. 1 = Transmit FIFO half-full or less condition generates the **SSPTXINTR** interrupt.
0	**RIE**	Read/write	Receive FIFO interrupt enable: 0 = Receive FIFO half-full or more condition does not generate the **SSPRXINTR** interrupt. 1 = Receive FIFO half-full or more condition generates the **SSPRXINTR** interrupt.

Table 5–69: SSPDR register, Offset 08_{hex}, Reset $----_{hex}$

Bit	31:16	15:0
Field	Reserved	**DATA**

When SSPDR is read, the entry in the receive FIFO (pointed to by the current FIFO read pointer) is accessed. As data values are removed by the SSP receive logic from the incoming data frame, they are placed into the entry in the receive FIFO (pointed to by the current FIFO write pointer).

When SSPDR is written to, the entry in the transmit FIFO (pointed to by the write pointer), is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. It is loaded into the transmit serial shifter, then serially shifted out onto the **SSPTXD** pin at the programmed bit rate.

When a data size of less than 16 bits is selected, the user must right-justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 bits is automatically right-justified in the receive buffer. When the PrimeCell SSPMS is programmed for National Microwire frame format, the default size for transmit data is eight bits (the most significant byte is ignored). The receive data size is controlled by the programmer. The transmit FIFO and the receive FIFO are not cleared even when SSE is set to zero. This allows the software to fill the transmit FIFO before enabling the PrimeCell SSPMS.

Bit definitions for SSPDR register:

Bits	Name	Type	Function
15:0	**DATA**	Read/write	Transmit/Receive FIFO: Read = Receive FIFO Write = Transmit FIFO. The user should right-justify data when the SSP is programmed for a data size that is less than 16 bits. Unused bits at the top are ignored by transmit logic. The receive logic automatically right-justifies.

Table 5–70: SSPSR register, Offset $0C_{hex}$, Reset 00_{hex}

Bit	31:5	4	3	2	1	0
Field	Reserved	**BSY**	**RFF**	**RNE**	**TNF**	**TFE**
This register is a read-only status register which contains bits that indicate the FIFO fill status and the SSP busy status.						

Bit definitions for SSPSR register:

Bits	Name	Type	Function
31:5	-	-	Reserved, read unpredictable, should be written as 0.
4	**BSY**	Read	SSP busy flag (read-only): 0 = SSP is idle. 1 = SSP is currently transmitting and/or receiving a frame or the transmit FIFO is not empty.
3	**RFF**	Read	Receive FIFO full (read-only): 0 = Receive FIFO is not full. 1 = Receive FIFO is full.
2	**RNE**	Read	Receive FIFO not empty (read-only): 0 = Receive FIFO is empty. 1 = Receive FIFO is not empty.
1	**TNF**	Read	Transmit FIFO not full (read-only): 0 = Transmit FIFO is full. 1 = Transmit FIFO is not full.
0	**TFE**	Read	Transmit FIFO empty (read-only): 0 = Transmit FIFO is not empty. 1 = Transmit FIFO is empty.

.
Table 5–71: SSPCPSR register, Offset 10_{hex}, Reset 00_{hex}

Bit	31:8	7:0
Field	Reserved	**CPSDVSR**
SSPCPSR is the clock prescale register and specifies the division factor by which the input **SSPCLK** should be internally divided before further use. The value programmed into this register should be an even number between 2–254. The least significant bit of the programmed number is hard-coded to zero. If an odd number is written to this register, data read back from this register will have the least significant bit as zero.		

Bit definitions for SSPCPSR register:

Bits	Name	Type	Function
31:8	-	-	Reserved, read unpredictable, should be written as 0.
7:0	**CPSDVSR**	Read/write	Clock prescale divider. Should be an even number from 2–254, depending on the frequency of **SSPCLK**. The least significant bit always returns zero on reads.

Table 5–72: Combined SSPIIR/SSPICR register, Offset 14_{hex}, Reset 00_{hex}

Bit	31:3	2	1	0
Field	Reserved	**RORIS**	**TIS**	**RIS**
The interrupt status is read from the SSP interrupt identification register (SSPIIR). A write of any value to the SSP interrupt clear register (SSPICR, bits 15:0) clears the SSP receive FIFO overrun interrupt. This interrupt clearing mechanism is in addition to the other mechanism mentioned for the SSPCR1 register. Therefore, clearing the **RORIE** bit in the SSPCR1 register will also clear the overrun condition if already asserted. All the bits are cleared to zero when reset.				

Bit definitions for combined SSPIIR/SSPICR register:

Bits	Name	Type	Function
15:0	-	Write	A write to this part of the register clears the receive overrun interrupt, regardless of the data value written.
31:3	-	Read	Reserved, read unpredictable.
2	**RORIS**	Read	Read: SSP Receive FIFO overrun interrupt status 0 = SSPRORINTR is not asserted. 1= SSPRORINTR is asserted
1	**TIS**	Read	Read: SSP transmit FIFO service request interrupt status 0 = SSPTXINTR is not asserted. 1= SSPTXINTR is asserted.
0	**RIS**	Read	Read: SSP receive FIFO service request interrupt status 0 = SSPRXINTR is not asserted. 1= SSPRXINTR is asserted.

Table 5–73: MODE_SELECT register, Offset 18_{hex}, Reset 02_{hex}

Bit	31:4	3	2:0
Field	Reserved	**ACLK**	**WMS**
This register selects the operational mode for the SSP interface circuitry. The ACLK field selects the SSP-peripheral's main operating clock (SSP5 only). The WMS field encodes different modes depending on whether the SSP operates as a master or a slave.			

Bit definitions for MODE_SELECT register:

Bits	Name	Type	Function	
3	**ACLK**	R/W	This bit selects the main clock SSPCLK for the SSP peripheral. It is only functional for SSP5. It should be written as '0' for SSPs 1 - 4. 0: The SSP5 peripheral uses the internally derived SSPCLK 1: The SSP5 peripheral uses the external clock input on I2S_AUDIOCLK pin	
2:0	**WMS**	R/W	The WMS field encodes the following SSP interface operational modes depending on whether the SSP operates as a master or a slave:	
			Master-SSP mode	Comment
			b'**000**: PUC Continuous	This SSP interface mode was designed for continuous communication with MAS 35xxF devices. It requires the SSP primecell to operate in Motorola SPI mode. In SSPCR0 register set FRF=b'00 and SPH=1 to put SSP primecell into required Motorola SPI mode.
			b'**001**: PUC Word	This SSP interface mode performs transmission of I^2S data streams in Sony format and complies with MAS 35xxF I^2S serial communication. It requires the SSP primecell to operate in TI mode. In SSPCR0 register set FRF=b'01 to put SSP primecell into required Texas Instruments mode.
			b'**010**: Transparent	Use this selection for operation of the SSP in it's primary modes: Motorola SPI, Texas Instruments and National Microwire.
			b'**011**: PUC Word TXdel	This SSP interface mode performs transmission of I^2S data streams in Philips format. It requires the SSP primecell to operate in TI mode. This mode only affects the transmitter; the receiver side operates as with PUC Word mode (b'001). In SSPCR0 register set FRF=b'01 to put SSP primecell into required Texas Instruments mode.
			b'100 - b'111: reserved	
			Slave-SSP mode	Comment
			b'**000**: PUC Continuous	Same as for Master-SSP
			b'**001**: PUC Word	Same as for Master-SSP
			b'**010**: Transparent	Same as for Master-SSP
			b'011: reserved	
			b'**100**: SlvRcv Continuous	High-speed equivalent for PUC Continuous mode for pure Slave-Receiver[1)]
			b'**101**: SlvRcv Word	High-speed equivalent for PUC Word mode for pure Slave-Receiver[1)]
			b'110 - b'111: reserved	

1) A pure Slave-Receiver (slave SSP where no bidirectional communication is required) can operate with serial input clocks SCLKx up to frequencies of f_SSPCLK / 3 in this mode. For slaves requiring bidirectional communication the frequency of SCLKx must be equal or less than f_SSPCLK / 12.

Table 5–74: DATA_SIZE register, Offset $1C_{hex}$, Reset 00_{hex}

Bit	31:2	1:0
Field	Reserved	**WS**
This register selects the word size of transferred data items for the SSP interface circuitry. The WS field must be written with the appropriate value for SSPs operating in PUC Word, PUC Word TXdel, or SlvRcv Word mode. The value of WS is irrelevant for all other modes.		

Bit definitions for DATA_SIZE register:

Bits	Name	Type	Function	
1:0	**WS**	R/W	This field encodes the word size of data items for all Word modes (PUC Word, PUC Word TXdel, SlvRcv Word). For any of these Word modes set FRF=b'01 in SSPCR0 register to put SSP primecell into required Texas Instruments mode. For additional settings see comments below.	
			Encoding	Comment
			b'**00**: 20-bit word size	Set DSS=b'1001 in SSPCR0 register to put SSP primecell Data Size to 10-bit data.
			b'**01**: 24-bit word size	Set DSS=b'1011 in SSPCR0 register to put SSP primecell Data Size to 12-bit data.
			b'**10**: 32-bit word size	Set DSS=b'1111 in SSPCR0 register to put SSP primecell Data Size to 16-bit data.
			b'**11**: 16-bit word size	

5.5.4.7. SSP Interrupt Logic

There are three interrupts generated by the SSP peripheral:

• SSPRXINTR
- SSP receive FIFO service interrupt request

• SSPTXINTR
- SSP transmit FIFO service interrupt request

• SSPRORINTR
- SSP receive overrun interrupt request.

All three interrupts are maskable, active HIGH interrupts. They are logically ORed to form a combined interrupt line to the interrupt controller, which should be programmed for level-sensitivity of the SSP interrupt (see Table 5–27 on page 31).

Each of the three maskable interrupts may be enabled or disabled by changing the enable bits in the SSPCR1 control register. Setting the appropriate enable bit to '1' enables the interrupt.

The status of the individual interrupt sources can be read from the interrupt status register SSPIIR. Note that reading the SSPIIR register does not clear the status bits.

5.5.4.7.1. SSPRXINTR

The receive interrupt status bit RIS in SSPIIR is set and the interrupt is signaled to the interrupt controller when there are four or more valid entries in the receive FIFO. They are cleared / deasserted when the number of entries decreases to two or less again, e.g. by servicing the interrupt.

5.5.4.7.2. SSPTXINTR

The transmit interrupt status bit TIS in SSPIIR is set and the interrupt is signaled to the interrupt controller when the transmit FIFO is less than or equal to half full (when there is space for four or more entries). They are cleared / deasserted when the number of entries increases to six or more again, e.g. by servicing the interrupt.

The transmitter interrupt **SSPTXINTR** is not qualified with the SSP enable signal 'SSE', which allows operation in one of two ways: Data can be written to the transmit FIFO prior to enabling the SSP and the interrupts. Alternatively, the SSP and interrupts can be enabled so that data can be written to the transmit FIFO by an interrupt service routine.

5.5.4.7.3. SSPRORINTR

The receive overrun status bit RORIS in SSPIIR is set and the interrupt is signaled to the interrupt controller when the receive FIFO is already full and an additional data frame is received, causing an overrun of the receive FIFO. Data is over-written in the shift register, but not the FIFO. To clear the RORIS bit and deassert the interrupt the lower 16 bits of the SSPIIR/SSPICR register should be written with any value.

5.5.5. I^2S Interface

The PUC 303xA provides an I^2S master interface through the following pins:

– I2S_CLK, serial data clock input/output.

– I2S_FRM, data word framing input/output.

– I2S_TXD, serial transmit data output.

– I2S_RXD, serial receive data input.

– I2S_AUDIOCLK, alternative ssp5clk clock source input.

This interface is implemented by using SSP5, as described in Section 5.5.4.

Note, that in principle any of the five SSP peripherals in the PUC 303xA device has the capability to perform the I^2S (PUC Word mode) protocol, though only SSP 5 has the ability to provide an alternative sampling clock. This is especially useful when PUC 303xA has to generate an accurate internal USB clock USB48CLK of 48 MHz while an accurate audio clock based I^2S data stream is to be output on the SSP5/I^2S interface (Micronas MAS 35xxF devices are ideally suited to provide accurate I2S_AUDIOCLK while acting as a I^2S slave).

5.5.6. Timers

The Triple Timer Counters (TTC) provide three independently programmable timing and counting functions for use within the PUC 303xA.

5.5.6.1. Features

– Three independently programmable 16-bit timer/ counters.
– Internal (pclk_apb) or external clock source (timers 1 and 2 only).
– Three independent prescalers, for clock speed selection.
– Interrupt generated either on overflow or at regular intervals.
– Interrupt generated when count matches a programmable value.
– Two output waveforms (timers 1 and 2), generated using overflow or interval and match interrupts.
– An Event Timer measures length of each external clock pulse, either high or low duration (timers 1 and 2 only).
– Counters can be incrementing or decrementing.
– Current value of each count register can be read at any time.

5.5.6.2. Description

The TTC module provides three independent timer and counter modules as shown in the diagram of Figure 5–19. This illustrates the top level structure of the module.

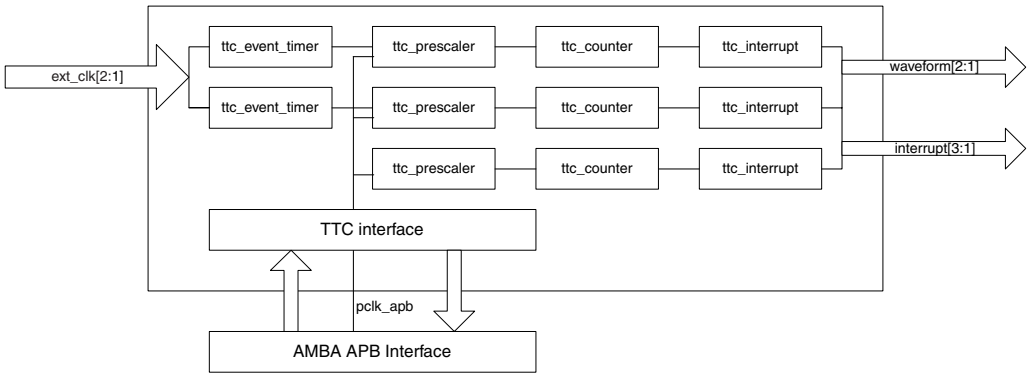


Fig. 5–19: TTC Top-Level Structure

All of the timers can be clocked using the APB system clock (pclk_apb). In addition for timers 1 and 2, an externally derived clock can be used instead. Each counter can independently prescale its selected clock input within a range from ÷2 to ÷65536 in powers of 2.

Counters can be set to decrement or increment.

Timers 1 and 2 have a pin on the device associated with them (TIMER_1 and TIMER_2 respectively). The pin can be used to clock the timer, count the number of events, measure event times and also generate PWM type waveforms.

Each external input is synchronized with pclk_apb before being applied to its timer or counter.

5.5.6.2.1. AMBA APB Interface

The AMBA APB interface contains the configuration and status registers that are addressed through the APB.

5.5.6.2.2. Event Timer

Event Timer modules are available for Timers 1 and 2 to measure durations of High or Low phases of externally applied signals in pclk cycles.

5.5.6.2.3. Prescaler

The design includes a Prescaler module to provide a selectable clock frequency for driving the timer-counter. The prescaler can be programmed to operate on the system clock, pclk_apb; or an external clock. The selected clock is then divided to provide the count clock. Division can be within the range ÷2 to ÷65536.

5.5.6.2.4. Counter Module

The Counter module can be incrementing or decrementing, and can be configured to count for a given interval. It will also compare three Match registers to the value of the Counter, and generate an interrupt if one matches.

5.5.6.2.5. Interrupt Module

There are three interrupt signals, one for each timer, available for use at system level. An interrupt occurs when a bit in the Interrupt Enable register and the corresponding bit in the Interrupt Detect register are both set. The resulting ANDed outputs are then ORed to generate the system interrupt signal.

The Interrupt register takes the interrupt signals from the corresponding timer-counter module and stores them until the register is read. When the Interrupt register is read by the processor, it is reset.

To enable an interrupt, a '1' must be written to the corresponding bit position in the Interrupt Enable register.

5.5.6.3. Initialization

On initialization, the counters are set to the following configuration:

– Overflow mode
– Internal clock selected.
– Counter free-running
– All interrupts disabled.
– Event Timer disabled.
– Output waveforms disabled.

5.5.6.4. Modes of Operation

Each of the timer counter modules can operate in one of four modes:

– Interval timing, incrementing count
– Interval timing, decrementing count
– Overflow detection, incrementing count
– Overflow detection, decrementing count

Register matching can also be programmed for each of these modes.

5.5.6.4.1. Interval Mode

If the interval bit is set in the counter control register, the counter counts up or down from a programmable interval value. An interrupt is generated when the count passes through zero. When Interval Mode operation is not enabled, the counter is free-running.

Incrementing

When the counter value register is equal to the interval register value, the counter is reset to zero, the interval interrupt is set and counting up restarted.

Decrementing

When the counter value register is equal to zero, the interval interrupt is set. The counter is then reset to the interval register value and counting down is restarted.

5.5.6.4.2. Overflow Mode

If the interval bit in the counter control register is not set, the counter can count up to or down from its full 16-bit value. An interrupt is generated when the count passes through zero.

Incrementing
When the counter value register reaches $FFFF_{hex}$ it overflows to zero, the overflow interrupt is set, and counting up is restarted.

Decrementing

When the counter value register reaches zero, the overflow interrupt is set. The counter then overflows to $FFFF_{hex}$ and counting down is restarted.

5.5.6.4.3. Register Matching

In the register matching operation, an interrupt is generated each time the counter value matches the value stored in the match register. Match events can be programmed for both Interval and Overflow Mode.

There are three match registers for each timer-counter. The module can, therefore, generate nine different match interrupts. When the match register select is set and the match register is equal to the counter value register then that match interrupt will be set.

5.5.6.5. TTC Register Descriptions

The TTC registers are summarized in Table 5–75 below. Addresses refer to the offsets from the TTC base address of $001E.4000_{hex}$.

A more detailed description of the register bits is shown in Table 5–75 to Table .

Table 5–75: TTC Register Map

Offset	R/W	Width	Name	Comment	Default
00_{hex}	r/w	7	Counter 1 Clock	Prescale and clock source control	00_{hex}
04_{hex}	r/w	7	Counter 2 Clock	Prescale and clock source control	00_{hex}
08_{hex}	r/w	7	Counter 3 Clock	Prescale control	00_{hex}
$0C_{hex}$	r/w	7	Counter 1 Control	Operational mode and reset	00_{hex}
10_{hex}	r/w	7	Counter 2 Control	Operational mode and reset	00_{hex}
14_{hex}	r/w	7	Counter 3 Control	Operational mode and reset	00_{hex}
18_{hex}	ro	16	Counter 1 Value	Current counter value	00_{hex}
$1C_{hex}$	ro	16	Counter 2 Value	Current counter value	00_{hex}
20_{hex}	ro	16	Counter 3 Value	Current counter value	00_{hex}
24_{hex}	r/w	16	Counter 1 Interval	Interval value	00_{hex}
28_{hex}	r/w	16	Counter 2 Interval	Interval value	00_{hex}
$2C_{hex}$	r/w	16	Counter 3 Interval	Interval value	00_{hex}
30_{hex}	r/w	16	Counter 1 match #1	Match value	00_{hex}
34_{hex}	r/w	16	Counter 2 match #1	Match value	00_{hex}
38_{hex}	r/w	16	Counter 3 match #1	Match value	00_{hex}
$3C_{hex}$	r/w	16	Counter 1 match #2	Match value	00_{hex}
40_{hex}	r/w	16	Counter 2 match #2	Match value	00_{hex}
44_{hex}	r/w	16	Counter 3 match #2	Match value	00_{hex}
48_{hex}	r/w	16	Counter 1 match #3	Match value	00_{hex}
$4C_{hex}$	r/w	16	Counter 2 match #3	Match value	00_{hex}
50_{hex}	r/w	16	Counter 3 match #3	Match value	00_{hex}
54_{hex}	ro	6	Interrupt Register 1	Counter 1 Interval, Match, Overflow and Event interrupts	00_{hex}
58_{hex}	ro	6	Interrupt Register 2	Counter 2 Interval, Match, Overflow and Event interrupts	00_{hex}
$5C_{hex}$	ro	6	Interrupt Register 3	Counter 3 Interval, Match, Overflow and Event interrupts	00_{hex}
60_{hex}	r/w	6	Interrupt Enable 1	ANDed with Interrupt Register 1	00_{hex}
64_{hex}	r/w	6	Interrupt Enable 2	ANDed with Interrupt Register 2	00_{hex}
68_{hex}	r/w	6	Interrupt Enable 3	ANDed with Interrupt Register 3	00_{hex}
$6C_{hex}$	r/w	3	Timer1 Event Control	Enable, pulse and overflow	0_{hex}
70_{hex}	r/w	3	Timer2 Event Control	Enable, pulse and overflow	0_{hex}

Table 5–75: TTC Register Map

Offset	R/W	Width	Name	Comment	Default
74_{hex}	r/w	3	Timer3 Event Control	Enable, pulse and overflow	0_{hex}
78_{hex}	ro	16	Timer1 Event Register	PCLK cycle count for event	00_{hex}
$7C_{hex}$	ro	16	Timer2 Event Register	PCLK cycle count for event	00_{hex}
80_{hex}	ro	16	Timer3 Event Register	PCLK cycle count for event	00_{hex}

Table 5–76: Counter Clock Register, Offsets: 00_{hex}, 04_{hex}, 08_{hex}; Reset to: 00_{hex}

Bit	31:7	6	5	4:1	0
Field	Reserved	Ex_E External Clock Edge: when this bit is set and the external clock is selected, the counter clocks on the negative-going edge of the external clock input.	C_Src Clock Source: when this bit is set the counter uses the external clock input, ext_clk; the default clock source is pclk (this bit must be set to 0 for timer 3).	PS_V Prescale value (N): if prescale is enabled, the count rate is divided by 2^{N+1} (÷2 to÷65536).	PS_En Prescale enable: when this bit is set the counter, clock source is prescaled; the default clock source is that defined by C_Src.

Table 5–77: Counter Control Register, Offsets: $0C_{hex}$, 10_{hex}, 14_{hex}; Reset to: 21_{hex}

Bit	31:7	6	5	4
Field	Reserved	Wave_pol Waveform polarity: When this bit is high, the waveform output goes from high to low on Match_1 interrupt and returns high on overflow or interval interrupt; when low, the waveform goes from low to high on Match_1 interrupt and returns low on overflow or interval interrupt.	Wave_en Output waveform enable, active low.	RST Setting this bit high resets the counter value and restarts counting; the RST bit is automatically cleared on restart.
Bit	3	2	1	0
Field	Match Register Match mode: when Match is set, an interrupt is generated when the count value matches one of the three match registers and the corresponding bit is set in the Interrupt Enable register.	DEC Decrement: when this bit is high the counter counts down.	INT When this bit is high, the timer is in Interval Mode, and the counter generates interrupts at regular intervals; when low, the timer is in overflow mode.	DIS Disable counter: when this bit is high, the counter is stopped, holding its last value until reset, restarted or enabled again.

Table 5–78: Counter Value Register, Offsets: 18_{hex}, $1C_{hex}$, 20_{hex}; Reset to: 00_{hex}

Bit	31:16	15:0
Field	Reserved	Value
At any time, a Timer Counter's count value can be read from its Counter Value Register.		

Table 5–79: Interval Register, Offsets: 24_{hex}, 28_{hex}, $2C_{hex}$; Reset to: 00_{hex}

Bit	31:16	15:0
Field	Reserved	Interval
If interval is enabled, this is the maximum value that the counter will count up to or down from.		

Table 5–80: Match Register, Offsets: 30_{hex}, 34_{hex}, 38_{hex}, $3C_{hex}$, 40_{hex}, 44_{hex}, 48_{hex}, $4C_{hex}$, 50_{hex}; Reset to: 00_{hex}

Bit	31:16	15:0
Field	Reserved	Match
When a counter has the same value as is stored in one of its match registers and match mode is enabled, a match interrupt is generated. Each counter has three match registers.		

Table 5–81: Interrupt Register, Offsets: 54_{hex}, 58_{hex}, $5C_{hex}$; Reset to: 00_{hex}

Bit	31:6	5	4	3
Field	Reserved	Ev Event timer overflow interrupt	Ov Counter overflow	M3 Match 3 interrupt
Bit	2	1	0	
Field	M2 Match 2 interrupt	M1 Match 1 interrupt	Iv Interval interrupt	

Table 5–82: Interrupt Enable Register, Offsets: 60_{hex}, 64_{hex}, 68_{hex}; Reset to: 00_{hex}

Bit	31:6	5:0
Field	Reserved	IEN
Enables for bits 5:0 in Interrupt Register. Corresponding bits must be set to enable the interrupt.		

Table 5–83: Timer Event Control Register, Offsets: $6C_{hex}$, 70_{hex}, 74_{hex}; Reset to: 00_{hex}

Bit	31:03	2	1	0
Field	Reserved	E_Ov Enable timer: when this bit is high, the event timer is enabled.	E_Lo When this bit is high, the timer counts pclk cycles during the low level duration of ext_clk; when low, the event timer counts the high level duration of ext_clk.	E_En When this bit is low, the event timer is disabled and set to zero when an Event Timer Register overflow occurs; when set high, the timer continues counting on overflow.

Table 5–84: Timer Event Register, Offsets: 78_{hex}, $7C_{hex}$, 80_{hex}; Reset to: 00_{hex}

Bit	31:16	15:0
Field	Reserved	Event
This register stores the result of the `pclk` count during the `ext_clk` high or low pulse.		

5.5.7. Real Time Clock

The Real Time Clock (RTC) Module provides programmable time of day, calendar and alarm functions for use within the PUC 303xA. Note, however, that the RTC is not available in some package options.

5.5.7.1. Features

– Complete time of day clock: 12/24 hour, hours, minutes, and seconds

– Calendar function: day of week, date of month, month, year, century, leap year compensation, and year 2000 compliant

– alarm function: month, date, hour, minute, and seconds resolution

– Event function: can set interrupt for every roll over of month, day, hour, minute, or second.

5.5.7.2. Description

The RTC module keeps track of time of day, to an accuracy of one second, and functions as a calendar keeping track of the day, month and year. The RTC comprises a binary coded decimal counter with the following data:

– Year from 1900 to 2999

– Month from 1 to 12

– Date from 1 to 28, 29, 30 or 31 (as a function of month and year)

– Day of week from 1 to 7 (mapping to date is not fixed)

– Hour from 0 to 23, or from 1 to 12 with the AM/PM flag set

– Minute from 0 to 59

– Second from 0 to 59.

An alarm register facilitates comparison of month, date, hour, minute and second. Each of these may be masked (that is forced to equality), so that, for example, an alarm can be generated at 12:37 on the 15th of every month. An interrupt can be generated on an alarm event.

Event detection is also available. This will generate an interrupt every time the time or calendar register rolls over into a new month, date, hour, minute, or second.

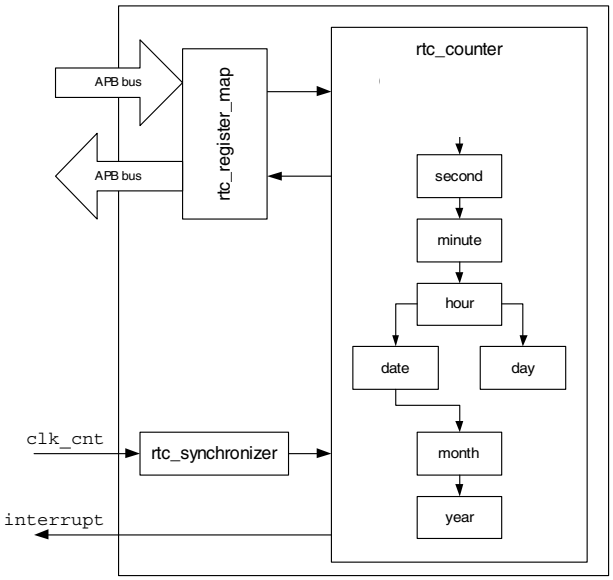


Fig. 5–20: Block Diagram

5.5.7.3. Standby Mode

The RTC module can be operated during STANDBY mode by ensuring that the oscrtcclk and rtcclk are enabled before entering STANDBY.

If unmasked within the PMU_WAKE_MASK register, an interrupt from the RTC will wake up the device.

5.5.7.4. Register Map and Formats

The register map lists the Offset, Name, Access, and Description associated with the RTC registers. In addition, the number of bits allocated to each register is listed (see Table 5–85)

The offset addresses listed are relative to the RTC base address, $001E.C000_{hex}$.

Table 5–85: RTC Register Map

Offset	Name	Access	Bits	Description	Default
00_{hex}	Control	RW	1:0	Start and stop for timing and calendar	00_{hex}
04_{hex}	12/24 Hour	RW	0:0	Set 12hr./24hr. mode	0_{hex}
08_{hex}	Time	RW, RO	30:8	Contains time values and the data valid flag	0000.0000_{hex}
$0C_{hex}$	Calendar	RW, RO	29:0	Contains date values and the data valid flag	$2000.010F_{hex}$
10_{hex}	Time alarm	RW	30:8	Contains time values for alarm	0000.0000_{hex}
14_{hex}	Calendar alarm	RW	13:3	Contains date values for alarm	0000_{hex}
18_{hex}	Alarm Enable	RW	5:1	Enables alarming on: month, date, hour, minute and second	00_{hex}
$1C_{hex}$	Event Flags	RO	6:1	Flags for alarm and roll-over events	00_{hex}
20_{hex}	Interrupt Enable	WO	6:1	Enables alarm and roll-over events	N/A
24_{hex}	Interrupt Disable	WO	6:1	Disables alarm and roll-over events	N/A
28_{hex}	Interrupt mask	RO	6:1	Masks off alarm and roll-over events	$7E_{hex}$
$2C_{hex}$	Status	RW	3:0	Valid data flags	F_{hex}
Each register address is defined by an offset from the RTC base address. The Interrupt-Enable and -Disable registers are not real registers. They cannot be read from.					

Table 5–86: Control Register, Offset 00_{hex}; Reset 00_{hex}

Bit	31: 02	01	00
Field	Reserved	Cal	Time
The Control Register is used to stop the RTC. Use this facility to update the time or calendar without affecting the other.			

Bit Definitions:

Time	When this field is set high the RTC stops incrementing the Calendar value.
Cal	When this field is set high the RTC stops incrementing the Time value.

Table 5–87: 12/24 Hour Register, Offset 04_{hex}; Reset 0_{hex}

Bit	31: 01	00
Field	Reserved	12/24

Bit Definitions

12/24	When this field is set high the RTC operates in 12 hour clock mode; otherwise, times are in 24 hour clock format.

Table 5–88: Time Register, Offset 08_{hex}, On Reset 0000.0000_{hex}

Bit	31	30	29: 28	27: 24	23	22: 20	19: 16
Field	Reserved	PM	HR_T	HR_U	Reserved	M_T	M_U
Bit	15	14: 12		11: 08	07: 00		
Field	Reserved	S_T		S_U	Reserved		
This register is loaded using the APB bus. Once the register has been used for writing data to the RTC it is then used for storing read data from the counters							

Bit Definitions

Label	Bits	Access	Description
PM	[30]	RW	AM/PM Select: in 12 hour clock mode, indicates PM when set.
HR_T	[29:28]	RW	Hours – tens BCD digit (0–2)
HR_U	[27:24]	RW	Hours – units BCD digit (0–9)
M_T	[22:20]	RW	Minutes – tens BCD digit (0–5)
M_U	[19:16]	RW	Minutes – units BCD digit (0–9)
S_T	[14:12]	RW	Seconds – tens BCD digit (0–9)
S_U	[11:8]	RW	Seconds – units BCD digit (0–9)

Table 5–89: Calendar Register, Offset $0C_{hex}$; Reset $2000.010F_{hex}$

Bit	31: 30	29: 28	27: 24		23: 20	19: 16
Field	Reserved	C_T	C_U		Y_T	Y_U
Bit	15: 14	13: 12	11: 08	07	06: 03	02: 00
Field	Reserved	D_T	D_U	M_T	M_U	DAY
This register is loaded using the APB bus. Once the register has been used for writing data to the RTC it is then used for storing read data from the counters.						

Bit Definitions

DAY	Day of the week (arbitrary) – units BCD digit (0–7)
M_U	Month – units BCD digit (0–9)
M_T	Month – tens BCD digit (0–1)
D_U	Date – units BCD digit (0–9)
D_T	Date – tens BCD digit (0–3)
Y_U	Year – units BCD digit (0–9)
Y_T	Year – tens BCD digit (0–2)
CUU	Century – units BCD digit (0–9)
CR_T	Century – tens BCD digit (1–2)

Table 5–90: Time Alarm Register, Offset 10_{hex}; Reset 0000.0000_{hex}

Bit	31	30	29: 28	27: 24	23	22: 20	19: 16
Field	Reserved	PM	HR_T	HR_U	Reserved	M_T	M_U
Bit	15	14: 12		11: 08	07: 00		
Field	Reserved	S_T		S_U	Reserved		
The Time alarm Register is used to program in a specific time when the alarm should cause an event, down to one hundredth of a second. The register map layout for the Time alarm Register is exactly the same as the Time Register.							

Table 5–91: Calendar Alarm Register, Offset 14_{hex}; Reset 0000_{hex}

Bit	31: 14	13: 12	11: 08	07	06: 03	02: 00
Field	Reserved	D_T	D_U	M_T	M_U	Reserved
The Calendar alarm Register can be used to program in a specific date and month when the alarm should cause an event. It is not possible to set an alarm more than 1 year into the future.						

Bit Definitions

M_U	Month – units BCD digit (0–9)
M_T	Month – tens BCD digit (0–1)
D_U	Date – units BCD digit (0–9)
D_T	Date – tens BCD digit (0–3)

Table 5–92: Alarm Enable Register, Offset 18_{hex}; Reset 00_{hex}

Bit	31: 6	05	04	03	02	01	00
Field	Reserved	MONTH	DATE	HOUR	MIN.	SEC.	Reserved
The alarm Enable Register is used to set the fields that can trigger an alarm. Setting a bit enables the corresponding time unit as a trigger event. The alarm triggering causes an event to be generated which is set in the Event Register. For example if all the fields are enabled then the alarm will trigger when a particular month, date, hour, minute, and second has been reached. If only the minutes field is enabled then the alarm will trigger when a particular minute is reached, and trigger every hour at that minute.							

Table 5–93: Event Flag Register, Offset $1C_{hex}$; Reset 00_{hex}

Bit	31: 7	06	05	04	03	02	01	00
Field	Reserved	ALARM	MONTH	DATE	HOUR	MIN.	SEC.	Reserved
The Event Flags Register is used to indicate that an event has occurred since the last reset. The register may be set even if the corresponding alarm Enable bit is not set. There are seven event registers that are written to when the corresponding event occurs. For example, if a minute rolls over, that is 59 seconds becomes 00 seconds, then the Minute event bit will be set. The Event Flags Register fields are cleared whenever their status is read. The Event Flags Register is read-only, and should not be written to. If an event occurs at the same time as the Event Flags Register is being read, the event will take priority.								

Table 5–94: Interrupt Enable Register, Offset 20_{hex}; Reset N/A

Bit	31: 7	06	05	04	03	02	01	00
Field	Reserved	ALARM	MONTH	DATE	HOUR	MIN.	SEC.	Reserved
The Interrupt Enable Register is used to set which events can generate an interrupt. An interrupt is enabled by writing a 1 to the appropriate Enable field. The interrupt generated will stay set until it is cleared, either by disabling the interrupt or by reading/clearing the event.								

Table 5–95: Interrupt Disable Register, Offset 24_{hex}; Reset N/A

Bit	31: 7	06	05	04	03	02	01	00
Field	Reserved	ALARM	MONTH	DATE	HOUR	MIN.	SEC.	Reserved
The Interrupt Disable Register is used to reset interrupts that are already enabled. An interrupt is disabled by writing a 1 to the appropriate Disable Register field. Any interrupts that are already set will remain set.								

Table 5–96: Interrupt Mask Register, Offset 28_{hex}; Reset $7E_{hex}$

Bit	31: 7	06	05	04	03	02	01	00
Field	Reserved	ALARM	MONTH	DATE	HOUR	MIN.	SEC.	Reserved
The Interrupt Mask Register indicates which interrupts are currently set. A field being set indicates that the corresponding interrupt is enabled. The Interrupt Mask Register is read-only, and should not be written to.								

Table 5–97: Status Register, Offset $2C_{hex}$; Reset F_{hex}

Bit	31: 04	03	02	01	00
Field	Reserved	Valid calendar alarm	Valid time alarm	Valid calendar	Valid time
The Status Register contains ‘data valid’ flags for data that has been input to the RTC. Invalid entry checking is performed after data is written in and when data is copied back from the counters to the register map. The appropriate field bit is set if the data is valid. The counter will not start if any of the valid flags is not set. The Status Register is read-only and should not be written to.					

5.5.7.5. Programming Instructions

5.5.7.5.1. Setting Time

The time counter must be stopped before setting a new time. The RTC must then be set into the desired 12-hr or 24-hr mode. The RTC can then be set to the desired time. As the RTC timer will not start if an attempt is made to load invalid data it is recommended that the status register be read to check this.

For example to set the RTC to 12:34:56 pm, the following sequence of register accesses should take place (see Example 5–7).

5.5.7.5.2. Setting Date

The date counter must be stopped before setting a new date. The RTC can then be set to a new date. The day of the week can also be set at this time. There is no defined mapping of the day of the week to the counter value. As the RTC timer will stop if an attempt is made to set an invalid date, it is recommended that the status register be read to check this.

For example to set the RTC to Sunday 4th March 2001 with the mapping, 0=Monday etc. the following sequence of register accesses should take place (see Example 5–2).

5.5.7.5.3. Setting Both Together

It is possible to perform both together. For example to set the RTC to set the RTC to 12:34:56 pm, on Sunday 4th March 2001 with the mapping, 0=Monday etc. the following sequence of register accesses should take place (see Example 5–3).

5.5.7.5.4. Setting Alarm to Specific Date and Time

Before an alarm can be changed, the interrupt must be disabled. Then the required date and time should be written into the various alarm registers. Note that if the RTC time is set to 12 hour mode, then the alarm must be set in the same mode.

For example to set the alarm to 23:55:00 on the 31st December the following sequence of register accesses should take place (see Example 5–4).

5.5.7.5.5. Setting Alarm to Regular Time

It is possible to set the alarm to repeat every month, day, hour, minute or second.

For example to set the alarm to repeat at 5 minutes past every hour the following sequence of register accesses should take place (see Example 5–5).

For example to set the alarm to repeat at 11:59:59 every day the following sequence of register accesses should take place (see Example 5–6).

5.5.7.5.6. Setting Events

It is possible to set an event to occur on each change of month, day etc. This is done by writing the appropriate bit in the interrupt enable register.

For example to set a daily event (at 12:00:00 am) execute the following sequence of register accesses (see Example).

5.5.7.5.7. Interrupts

Whenever an alarm or event trigger has occurred, the RTC raises its interrupt line to the system interrupt controller.

Once the software has decoded the interrupt as one from the RTC, it is possible to further decode the interrupt by reading the Event Flags register. A read of this register will return a bit for each alarm or event trigger that has occurred since the register was previously read, and will clear those bits in the register.

Example 5–1: Setting Time

```
write( Control,        0x00000001 )  // stop time counter
write( 12/24 hour,     0x00000001 )  // 12 hr mode
write( Time,           0x52345600 )  // 1.1.2.3.4.5.6. (see Time Register)
read ( Status,         result )
if  ( result !=        0x0000000f ) then
     error
else
     write ( Control, 0x00000000 ) // restart time counter
```

Example 5–2: Setting Date

```
write( Control,   0x00000002 )  // stop calendar counter
write( Calendar,  0x2001041f )  // 2.0.0.1.0.4.0.3.7 (see Calendar Register)
read ( Status,    result )
if ( result !=    0x0000000f ) then
     error
else
     write ( Control, 0x00000000 )  // restart calendar counter
```

Example 5–3: Setting Both Together

```
write( Control,   0x00000003 )  // stop both counters
write( 12/24 hour,0x00000001 )  // 12 hr mode
write( Time,      0x52345600 )  // 1.1.2.3.4.5.6 (see Time Register)
write( Calendar,  0x2001041f )  // 2.0.0.1.0.4.0.3.7 (see Calendar Register)
read ( Status,    result )
if ( result !=    0x0000000f ) then
     error
else
     write ( Control, 0x00000000 )  // restart calendar counter
```

Example 5–4: Setting Alarm to Specific Date and Time

```
write(Interrupt disable,   0x00000040)   // disable alarm
write(Time alarm,          0x23550000)   // set time
write(calendar alarm,      0x00003190)   // set calendar
write(alarm enable,        0x0000003f)   // enable all time & date fields
read ( Status,             result )
if ( result !=             0x0000000f ) then
     error
else
     write(interrupt enable,0x00000040) // enable alarm
```

Example 5–5: Setting Alarm to Regular Time

```
write(Interrupt disable,   0x00000040)   // disable alarm
write(Time alarm,          0x00050000)   // set time
write(alarm enable,        0x00000004)   // enable minute fields
read ( Status,             result )
if ( result !=             0x0000000f ) then
     error
else
     write(interrupt enable,0x00000040) // enable alarm
```

Example 5–6: Setting Alarm to Regular Time

```
write(Interrupt disable,   0x00000040)   // disable alarm
write(Time alarm,          0x115959)     // set time
write(alarm enable,        0x0000000f)   // enable all time fields
read ( Status,             result )
if ( result !=             0x0000000f ) then
     error
else
     write(interrupt enable,0x00000040) // enable alarm
```

Example 5–7: Setting Events

```
write(Interrupt enable,0x00000010)// enable date event
```

5.5.8. Watchdog Timer

The Watchdog Timer (WDT) IP Module provides programmable software activity monitoring functions for use within the PUC 303xA. Figure 5–21 shows a block diagram of the watchdog timer module.

5.5.8.1. Features

– On timeout, outputs one or a combination of:
  – System Reset
  – System Interrupt
– Variable timeout period. Programmable countdown rate and number of cycles to count.
– Timeout range 32,760 to 268,431,360 clock cycles (512 µs to 4.2s at 64 MHz)

5.5.8.2. Description

The Watchdog Timer can be used to prevent system lock-up, if e.g. software becomes trapped in a deadlock. In normal operation the user restarts the watchdog at regular intervals before the timer counts down to zero.

If the timer reaches zero and the watchdog is enabled, one or both of the following signals is generated:

– a system reset,
– an interrupt.

The watchdog time-out period is variable.

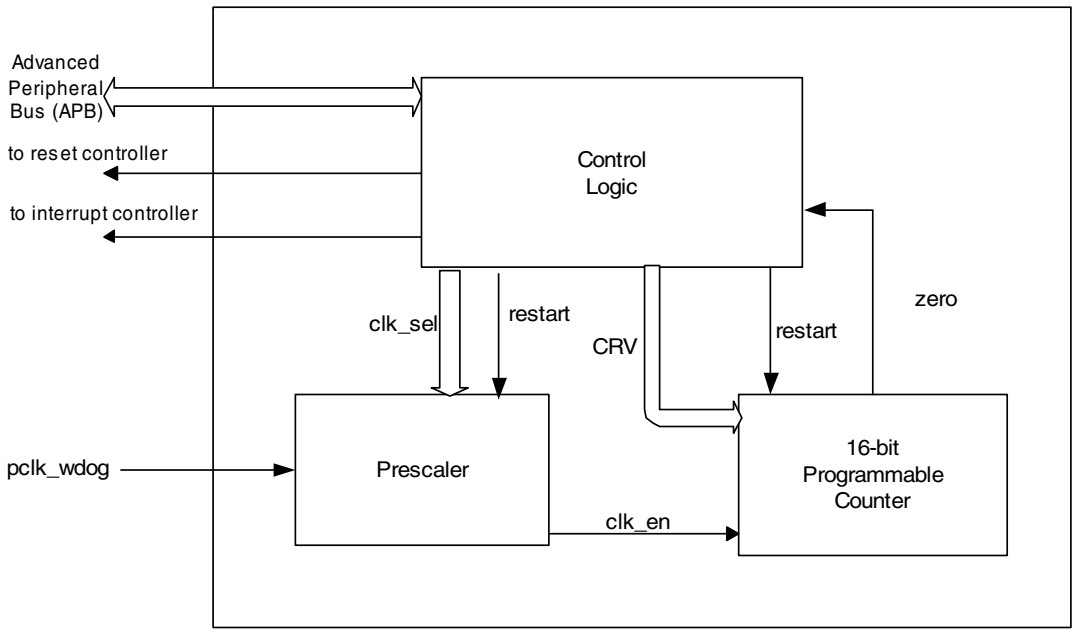


Fig. 5–21: Block diagram: watchdog timer module

5.5.8.2.1. Control Logic Block

The Control Logic Block sends and receives signals across the APB. If the data received from the APB matches the key of the register at the specified address within the block, the Control Logic block can be written to.

Depending upon which register is written to, this can result in the counter being restarted, the speed of the count being altered, or some change in the combination of signals output from the Watchdog Timer when the counter reaches zero.

5.5.8.2.2. Prescaler Block

This block inputs the CLK_SEL signal from the control register. The CLK_SEL signal is sampled on every rising clock edge. It determines how often the periodic CLK_EN output signal occurs according to Table 5–100 . If a restart signal is received the prescaler should reset.

5.5.8.2.3. 16-bit Programmable Counter

The 16-bit Programmable Counter counts down to zero at a rate defined by the COUNT_ENABLE input. The counter reads the restart signal on the positive edge of each clock cycle. If the restart signal is high the counter immediately restarts the countdown at the value read from the COUNTER_RESTART_VALUE signal.

If the counter reaches zero, it stays at zero until it is restarted. While the counter is at zero, the zero output signal goes high. The zero output signal remains high until the counter is restarted.

5.5.8.3. Watchdog Register Descriptions

Table 5–98 summarizes the watchdog registers. Addresses are specified as an offset to the Watchdog base address of $001E.8000_{hex}$.

Table 5–98: Watchdog Register Map

Offset	R/W	Width	Name	Comment	Default
00_{hex}	R/W / WO	24	MODE	WD Zero Mode register	$00.01C2_{hex}$
04_{hex}	R/W / WO	18	CONTROL	Counter Control register	$0.003C_{hex}$
08_{hex}	WO	16	RESTART	Restart Key Register	N/A
$0C_{hex}$	RO	1	STATUS	Watchdog Status Register	0_{hex}

5.5.8.3.1. Zero Mode Register (MODE)

Table 5–99: WD Zero Mode Register, Offset 0_{hex} , Reset $00.01C2_{hex}$

Bit	31:24	23:12	11:04	03	02	01	00
Field	Reserved	ZKEY	Reserved	Reserved	IRQEN	RSTEN	WDEN

Bit Definitions

0	**WDEN**	Watchdog Enable: If set, the watchdog is enabled and can generate any signals that are enabled.
1	**RSTEN**	Reset Enable: If set, the watchdog will issue an internal reset when the counter reaches zero, if WDEN = 1.
2	**IRQEN**	Interrupt Request Enable: If set, the watchdog will issue an interrupt request when the counter reaches zero, if WDEN = 1.
3	reserved	This reserved bit should be written with the Reset value (0_{bin})
11:4	reserved	This reserved field should be written with the Reset value ($1C_{hex}$)
23:12	**ZKEY**	Zero Access Key: Writes to the zero mode register are only valid if this field is set to ABC_{hex}; this field is Write Only.

The Zero Mode Register controls the behavior of the watchdog timer when the counter reaches zero. In order to write to this register, the correct ZKEY must be passed on the data bus along with the data to be written.

If the Control Logic block receives a zero signal it checks the fields RSTEN and IRQEN to discover which signals are enabled and can therefore be emitted.

If the Watchdog is not enabled, the value WDEN is low, and no signals are output.

Once emitted, the output signals can not be emitted again until the Programmable Counter is restarted, the zero signal dropped, and a new zero signal received. If a signal is being output, a signal is passed to the Zero Mode Register blocking it from being written to.

5.5.8.3.2. Counter Control Register (CONTROL)

Table 5–100: Counter Control Register

Bit	31:18	17:06	05:02	01:00
Field	Reserved	CKEY	CRV	CLKSEL

Bit Definitions

CLKSEL	Counter Clock Prescale: selects the prescaler division ratio: 00 = `pclk_wdog` ÷ 8 01 = `pclk_wdog` ÷ 64 10 = `pclk_wdog` ÷ 256 11 = `pclk_wdog` ÷ 4096 Note: If a restart signal is received the prescaler should be reset.
CRV	Counter Restart Value: The counter is restarted with the value $NFFF_{hex}$, where N is the value of this field.
CKEY	Counter Access Key: Writes to the control register are only valid if this field is set to 248_{hex}; this field is Write Only.

The Counter Control Register controls the time taken for a restarted counter to reach zero. If the APB writes to the Register with the appropriate CKEY, the restart value and clock rate values can be changed. These fields retain their values until they are changed again or a system reset occurs.

5.5.8.3.3. Restart Register (RESTART)

.

Table 5–101: Restart Register

Bit	31:16	15:00
Field	Reserved	RSTKEY Restart Key: the watchdog is restarted if this field is set to the value 1999_{hex}

This write-only register restarts the counter when it receives the correct key. The RESTART is a Write Only register.

If the data received does match the key, a restart signal is sent to the 16-bit Programmable Counter.

5.5.8.3.4. Watchdog Status Register (STATUS)

Table 5–102: Watchdog Zero Register

Bit	31 : 01	00
Field	Reserved	WDZ Watchdog Zero: this field is set when the watchdog reaches zero count, that is times out.

This register indicates whether a watchdog invoked reset has occurred. This register maintains it's value after the system has finished a watchdog reset sequence. Reading this register clears this status bit.

Alternatively the PMU register PMU_RESET can be read for indication of the reset cause.

5.5.8.4. Watchdog Timer Reset

In the event of a power-on reset, the Watchdog Timer resets to the following state:

– The watchdog timer is disabled.

– The timer is set to generate a reset on a timeout, but not an interrupt.

Note: The watchdog is reset by the n_watchrst signal from the reset generator module, to ensure that the watchdog is not reset by its own generated reset.

5.5.8.5. Watchdog Timer Enable Sequence

To enable the Watchdog Timer for the first time following a power-on reset, the following registers must be initialized.

1. Initialize the WD Counter Control Register. For example, writing $923C_{hex}$ to the CONTROL register sets prescale to pclk_wdog/8 and the Counter Restart Value to its maximum. The first twelve bits set the CKEY to enable a write to the register.

Enable the timer: for example write $AB.C1C5_{hex}$ to MODE register; the final four bits enable the watchdog and the interrupt signal.

5.5.9. General Purpose I/O module

The General Purpose I/O (GPIO) GPIO module enables the programming of pin level input and output functions for use within the PUC 303xA.

5.5.9.1. Features

– Up to 66 independently programmable I/O pins.
– Each pin provides input, output and output enable for bi-directional I/O pins.
– Each pin can be programmed as input or output.
– Each pin can be bypassed to or from a separate peripheral device.
– Each pin can separately trigger the GPIO interrupt on several event types.

5.5.9.2. Description

The GPIO module incorporates 66 programmable pins. Each pin can be independently programmed as an input pin, an output pin, or a ‘bypass’ pin. The general structure of each GPIO pin can be seen in Figure 5–22.

The pins are grouped into ports of widths between six and eight bits, and labelled as ports A to I.

5.5.9.2.1. GPIO Mode

In the default GPIO mode, the pins of the device are directly accessible through the GPIO registers. There are two possible types of GPIO mode, input mode and output mode.

Input Mode

In Input mode, data is routed through from the I/O pin, and the current value can be read in the INPUT_VALUE register.

To enable this mode for a pin, the appropriate bit in the DIRECTION_MODE register should be set to a 1.

Output Mode

In Output mode, data is routed through to the I/O pin from the OUTPUT_VALUE register.

To enable this mode for a pin, the appropriate bit in the DIRECTION_MODE should be set to a 0. The data HIGH or LOW, register should be written to the OUTPUT_VALUE register.

The output enable for the pin can then be controlled as desired, by setting the appropriate bit in the OUTPUT_ENABLE register.

5.5.9.3. Bypass Mode

Bypass mode and the two GPIO modes are mutually exclusive at an individual pin. In bypass mode, the complete control of the pin is handled by the functional block associated with the particular pin.

As the signals are not latched or registered, this allows the pin to behave as in the functional description for that block.

The direction control for the pad is taken from the bypassing module, so that the GPIO direction registers are “don’t care”.

5.5.9.4. Interrupts

An interrupt trigger is generated for a pin if a pre-defined event type is seen on the pin input. The event can be specified as high or low level, rising or falling edge, or any edge.

Use the registers INT_TYPE, INT_VALUE and INT_ON_ANY to define the event type that triggers an interrupt.

The event is recorded in a read-only register INT_STATUS. This register is cleared whenever the INPUT_STATUS register is read.

Each group of pins, called ports A to I, have one direct interrupt connection to the system interrupt controller. This interrupt line for each port, is the logical OR of all of the interrupt status for pins within the port.

An interrupt is output if an interrupt trigger is seen for any pin, whose INT_MASK register bit is clear.

5.5.9.5. Signal Interface

Each pin has a block structure as shown Figure 5–22.

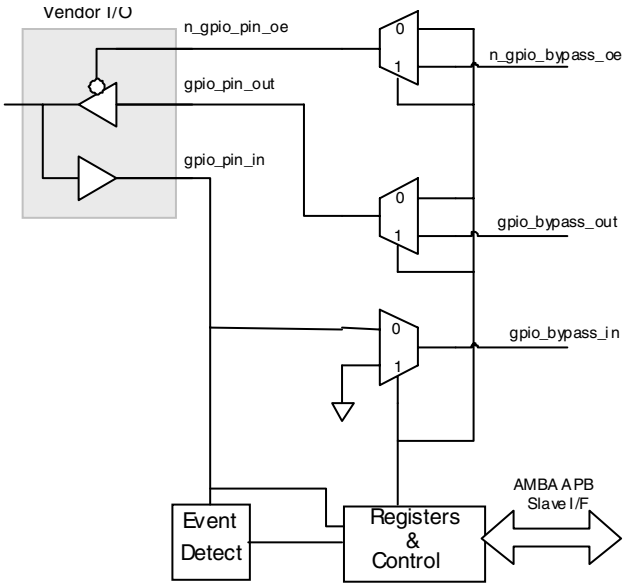


Fig. 5–22: Block structure of pins

The diagram shows the major structure of the block where the data to or from pins passes through multiplexers that are controlled by registers. Each GPIO pin has its own set of control registers. Additional register bits are used for setting the output and output enable values.

An event detector is used to determine input changes and this, in conjunction with a mask register, is used to generate a level triggered interrupt signal.

To put the pin in bypass mode, the BYPASS_MODE register must be set. The pin direction can be changed by setting the DIRECTION_MODE register. Bypass mode enables pins to be shared by the GPIO and another peripheral. The function of the pin can, therefore, be reallocated if the peripheral is not in use.

When in GPIO mode the output value on gpio_pin_out is driven by the output value register. When the direction register is in input mode n_gpio_pin_oe is always inactive (=1), implying that the output buffer is always disabled.

5.5.9.6. Initialization

On power-up, all GPIO pins on the PUC 303xA device are set into GPIO input mode, and all interrupts are masked.

The only exception to this is the USB data pullup pin, usb_pullup, which is initialized into bypass mode. The USB functional block therefore has direct control of the pin, which makes it a driven output.

Software can re-initialize by writing to the GPIO registers. When changing the mode of a pin, the output should be disabled to prevent contention on the GPIO bidirectional pad, and the interrupt disabled to prevent a spurious event occurring.

Note that during power down mode of the PUC 303xA device, the GPIO pins preserve their state. Only a full power-on reset sequence will reset the state of the GPIO pins.

5.5.9.7. GPIO Address Map

The 66 GPIO pins in the design are split into nine ports (ports H and J are only available on 100-pin packages). Table 5–103 summarizes the offsets for each port from the GPIO base address of $001C.C000_{hex}$.

Table 5–103: GPIO Port Offsets

Offset Address	Port	Port Width (N)
000_{hex}	A	8 bit
040_{hex}	B	8 bit
080_{hex}	C	8 bit
$0C0_{hex}$	D	7 bit
100_{hex}	E	7 bit
140_{hex}	F	7 bit
180_{hex}	G	5 bit
$1C0_{hex}$	H[1)]	8 bit
200_{hex}	J[1)]	8 bit

1) Port available on 100-pin packages only

5.5.9.8. Programming Interface

Each pin within a port has a field within each register, which is of the format shown in Table 5–105. The value of N is the port width defined in Table 5–103.

Each register address listed in Table 5–105 is defined by an offset that relates to the base address of the GPIO port as defined in Table 5–103.

Table 5–104: Pin Field Format

Bit	31: (N+1)	N:0
Field	Reserved (N<31)	Pin N:Pin 0

Table 5–105: GPIO Port Register Map

Offset	Name	Access	Bits	Description	Default
00_{hex}	BYPASS_MODE	RW	[N:0]	If bit is 1: set pin to bypass mode If bit is 0: set pin o GPIO mode	00_{hex}
04_{hex}	DIRECTION_MODE	RW	[N:0]	If bit is 1: set pin to input mode If bit is 0: set pin to output mode	FF_{hex}
08_{hex}	OUTPUT_ENABLE	RW	[N:0]	If bit is 1: set pin to output enabled If bit is 0: set pin to output disabled (Ignored if the pin is set to input or bypass modes)	00_{hex}
$0C_{hex}$	OUTPUT_VALUE	RW	[N:0]	This register contains the value to be driven out of the pins. The output will only appear at the port if the pin is set to GPIO output mode.	00_{hex}
10_{hex}	INPUT_VALUE	RO	[N:0]	The input value is read from this register, regardless of the pin mode.	00_{hex}
14_{hex}	INT_MASK	RO	[N:0]	This register is used to mask interrupt events being signalled by GPIO_INT. If Bit is 1: bit of INT_STATUS is ignored. If Bit is 0: an event on pin will set an interrupt. Bits are set and cleared using the registers INT_ENABLE and INT_DISABLE.	FF_{hex}
18_{hex}	INT_ENABLE	WO	[N:0]	If bit is 1, bit of INT_MASK is cleared.	00_{hex}
$1C_{hex}$	INT_DISABLE	WO	[N:0]	If bit is 1, bit of INT_MASK is set.	00_{hex}
20_{hex}	INT_STATUS	RO	[N:0]	If bit is 1 an interrupt-generating event has occurred on bit of INPUT_VALUE. INT_STATUS is set regardless of INT_MASK. This register is cleared by a read.	00_{hex}
24_{hex}	INT_TYPE	RW	[N:0]	If bit is 1, interrupt is level triggered If bit is 0, interrupt is edge triggered	00_{hex}
28_{hex}	INT_VALUE	RW	[N:0]	If bit is 1, interrupt is triggered on high level or rising edge, depending on INT_TYPE value If bit is 0, interrupt is triggered on low level or falling edge, depending on INT_TYPE value	00_{hex}
$2C_{hex}$	INT_ON_ANY	RW	[N:0]	If bit is 1 edge triggering occurs on any edge, otherwise edge specified in INT_VALUE triggers an interrupt. INT_ON_ANY is ignored if INT_TYPE =1.	00_{hex}

6. Specifications

6.1. Outline Dimensions

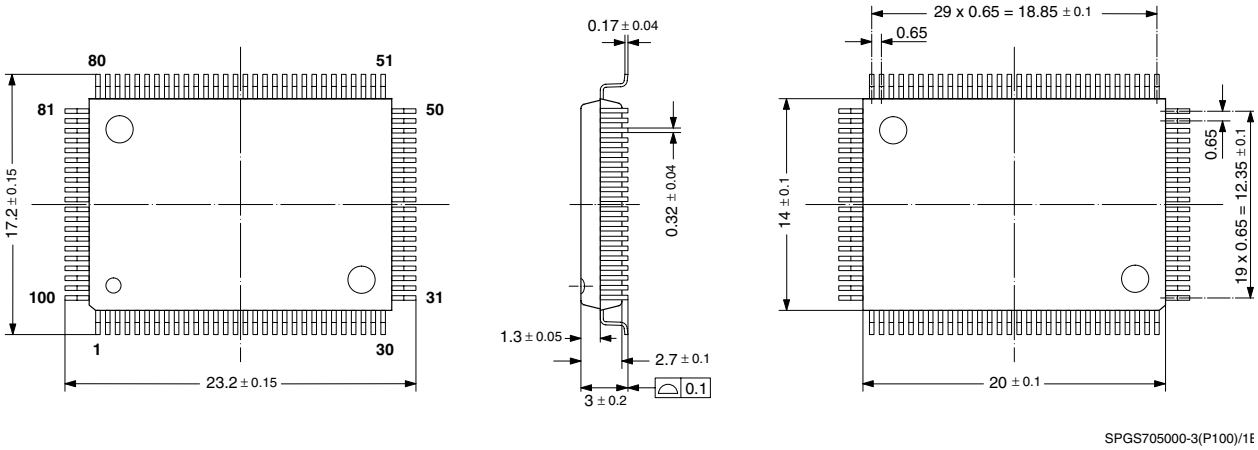


Fig. 6–1:
100-Pin Plastic Quad Flat Pack
(PQFP100)
Weight approximately 1.61 g
Dimensions in mm

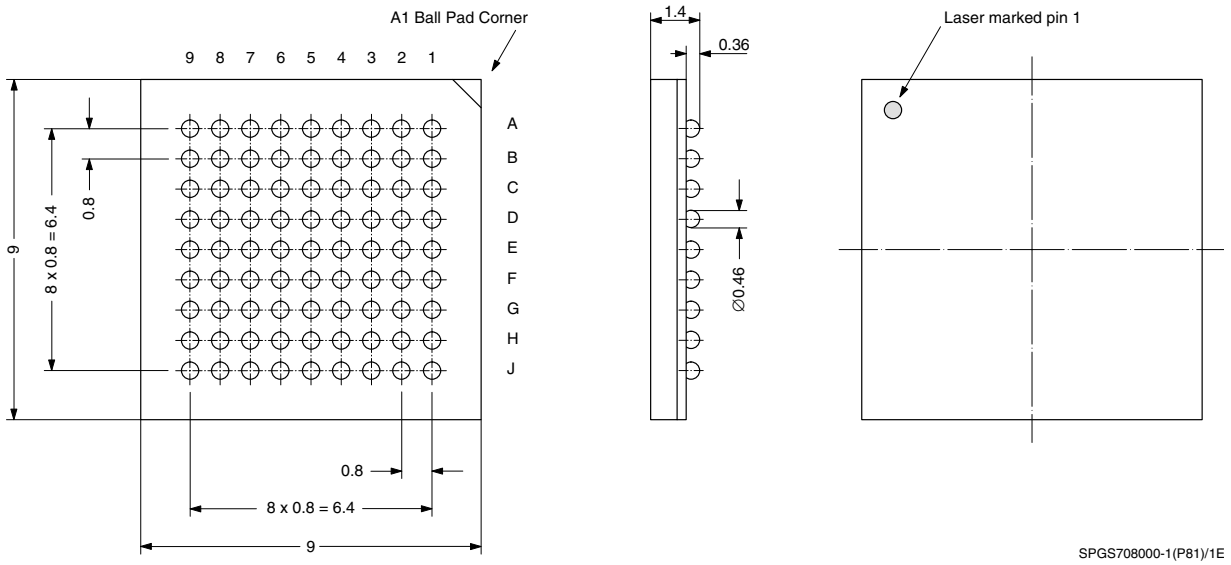


Fig. 6–2:
81-Pin Low-Profile Fine-Pitch Ball Grid Array Pack
(LFBGA81)
Weight approximately 0.19 g
Dimensions in mm

6.2. Pin Connections and Short Descriptions

NA = not available on package
NC = not connected
S = Schmitt-trigger input
U = internal pull-Up
D = internal pull-Down
4mA = output with 4mA drivestrength
12mA = output with 12mA drivestrength

Pin No.		Pin Name	GPIO port	I/O-Type	I / O properties	Short Description
PQFP 100-pin	LFBGA 81-pin					
1	B8	I2S_RXD	PE.0	IN/OUT	4mA	I2S Receive Data
2	C6	I2S_CLK	PE.1	IN/OUT	S / 4mA	I2S Serial Clock
3	A7	I2S_FRM	PE.2	IN/OUT	4mA	I2S Framing
4	B7	I2S_TXD	PE.3	IN/OUT	4mA	I2S Transmit Data
5	B6	SCLK4	PE.4	IN/OUT	S / 4mA	SSP4 Serial Clock
6	A6	SFRM4	PE.5	IN/OUT	4mA	SSP4 Framing
7	C5	SRXD4	PE.6	IN/OUT	4mA	SSP4 Receive Data
8	A9	VDD_CORE		SUPPLY		Core VDD supply
9	A8	VSS_CORE		SUPPLY		Core VSS supply
10	F7	TXD2	PA.0	IN/OUT	4mA	UART2 Transmit Data
11	F6	RXD2	PA.1	IN/OUT	4mA	UART2 Receive Data
12	E6	nRTS2	PA.2	IN/OUT	4mA	UART2 Request to Send, active low
13	D6	nCTS2	PA.3	IN/OUT	4mA	UART2 Clear To Send, active low
14	F5	TXD1	PA.4	IN/OUT	4mA	UART1 Transmit Data
15	E5	RXD1	PA.5	IN/OUT	4mA	UART1 Receive Data
16	D5	nRTS1	PA.6	IN/OUT	4mA	UART1 Request To Send, active low
17	F4	nCTS1	PA.7	IN/OUT	4mA	UART1 Clear To Send, active low
18	E4	VSS_IO		SUPPLY		I/O VSS supply
19	D4	VDD_IO		SUPPLY		I/O VDD supply
20	B5	nDSR2	PB.0	IN/OUT	4mA	UART2 Data Set Ready, active low
21	A5	nDTR2	PB.1	IN/OUT	4mA	UART2 Data Terminal Ready, active low
22	C4	nDCD2	PB.2	IN/OUT	4mA	UART2 Data Carrier Detect, active low

Pin No. PQFP 100-pin	Pin No. LFBGA 81-pin	Pin Name	GPIO port	I/O-Type	I / O properties	Short Description
23	A4	nRI2	PB.3	IN/OUT	4mA	UART2 Ring Indicator, active low
24	B4	nDSR1	PB.4	IN/OUT	4mA	UART1 Data Set Ready, active low
25	A3	nDTR1	PB.5	IN/OUT	4mA	UART1 Data Terminal Ready, active low
26	B3	nDCD1	PB.6	IN/OUT	4mA	UART1 Data Carrier Detect, active low
27	A2	nRI1	PB.7	IN/OUT	4mA	UART1 Ring Indicator, active low
28	A1	PORTG_0	PG.0	IN/OUT	4mA	Dedicated GPIO
29	B2	PORTG_1	PG.1	IN/OUT	4mA	Dedicated GPIO
30	B1	PORTG_2	PG.2	IN/OUT	4mA	Dedicated GPIO
31	C2	PORTG_3	PG.3	IN/OUT	4mA	Dedicated GPIO
32	C1	PORTG_4	PG.4	IN/OUT	4mA	Dedicated GPIO
33	D2	nRESET		IN	S	System Reset, active low
34	NA	WAKE_UP		IN	U, S	Wake-Up from STANDBY
35	D1	MODSEL_1		IN	D	Mode Select pin 1
36	C3	MODSEL_0		IN	D	Mode Select pin 0
37	D3	nTRST		IN	U, S	JTAG Reset, active low
38	E3	TCK		IN	S	JTAG Clock
39	F3	TDI		IN	U	JTAG Test Data Input
40	G3	TDO		OUT	4mA	JTAG Test Data Output
41	F2	TMS		IN	U	JTAG Test Mode Select
42	E1	PORTF_0	PF.0	IN/OUT	4mA	Dedicated GPIO
43	E2	PORTF_1	PF.1	IN/OUT	4mA	Dedicated GPIO
44	F1	I2S_AUDIOCLK	PF.2	IN/OUT	S / 4mA	I^2S (SSP5) Audio Clock Input
45	G2	UC_SDA	PF.3	IN/OUT	S / 4mA	I^2C Serial Data, handled by USB Core
46	H2	UC_SCL	PF.4	IN/OUT	S / 4mA	I^2C Serial Clock, handled by USB Core
47	G1	VSS_IO		SUPPLY		I/O VSS supply
48	H1	VDD_IO		SUPPLY		I/O VDD supply
49	NA	PORTH_0	PH.0	IN/OUT	D / 4mA	Dedicated GPIO

Pin No.		Pin Name	GPIO port	I/O-Type	I / O properties	Short Description
PQFP 100-pin	LFBGA 81-pin					
50	NA	PORTH_1	PH.1	IN/OUT	D / 4mA	Dedicated GPIO
51	NA	PORTH_2	PH.2	IN/OUT	D / 4mA	Dedicated GPIO
52	NA	PORTH_3	PH.3	IN/OUT	D / 4mA	Dedicated GPIO
53	NA	PORTH_4	PH.4	IN/OUT	D / 4mA	Dedicated GPIO
54	NA	PORTH_5	PH.5	IN/OUT	D / 4mA	Dedicated GPIO
55	NA	PORTH_6	PH.6	IN/OUT	D / 4mA	Dedicated GPIO
56	NA	PORTH_7	PH.7	IN/OUT	D / 4mA	Dedicated GPIO
57	J1	VSS_CORE		SUPPLY		Core VSS supply
58	J2	VDD_CORE		SUPPLY		Core VDD supply
59	H3	USB_SENSE	PF.5	IN/OUT	4mA	USB interface Sense
60	J3	USB_PULLUP	PF.6	IN/OUT	4mA	USB interface Pullup
61/NC	H4/NC	VSS_USB[1)]		SUPPLY		USB VSS supply
62/NC	G5/NC	USB_DPLUS[1)]		IN/OUT	differential	USB interface D^+
63/NC	G4/NC	USB_DMINUS[1)]		IN/OUT	differential	USB interface D^-
64/NC	J4/NC	VDD_USB[1)]		SUPPLY		USB VDD supply
65	H5	SCLK1	PC.0	IN/OUT	S / 4mA	SSP1 Serial Clock
66	G6	STXD1	PC.1	IN/OUT	4mA	SSP1 Transmit Data
67	J6	SFRM1	PC.2	IN/OUT	4mA	SSP1 Framing
68	H6	SRXD1	PC.3	IN/OUT	4mA	SSP1 Receive Data
69	J7	SCLK2	PC.4	IN/OUT	S / 4mA	SSP2 Serial Clock
70	H7	STXD2	PC.5	IN/OUT	4mA	SSP2 Transmit Data
71	J8	SFRM2	PC.6	IN/OUT	4mA	SSP2 Framing
72	J9	SRXD2	PC.7	IN/OUT	12mA	SSP2 Receive Data
73	J5	CLKOUT		OUT	4mA	Clock Output
74	NA	PORTI_0	PI.0	IN/OUT	D, 4mA	Dedicated GPIO
75	NA	PORTI_1	PI.1	IN/OUT	D, 4mA	Dedicated GPIO
76	NA	PORTI_2	PI.2	IN/OUT	D, 4mA	Dedicated GPIO
77	NA	PORTI_3	PI.3	IN/OUT	D, 4mA	Dedicated GPIO
78	NA	PORTI_4	PI.4	IN/OUT	D, 4mA	Dedicated GPIO
79	NA	PORTI_5	PI.5	IN/OUT	D, 4mA	Dedicated GPIO
80	NA	PORTI_6	PI.6	IN/OUT	D, 4mA	Dedicated GPIO

Pin No.		Pin Name	GPIO port	I/O-Type	I / O properties	Short Description
PQFP 100-pin	LFBGA 81-pin					
81	NA	PORTI_7		IN/OUT	D, 4mA	Dedicated GPIO
82	H9	VSS_IO		SUPPLY		I/O VSS supply
83	H8	VDD_IO		SUPPLY		I/O VDD supply
84	G9	VDD_CORE		SUPPLY		Core VDD supply
85	G8	VSS_CORE		SUPPLY		Core VSS supply
86	G7	SRXD3	PD.0	IN/OUT	12mA	SSP3 Receive Data
87	E7	SFRM3	PD.1	IN/OUT	4mA	SSP3 Framing
88	D7	SCLK3	PD.2	IN/OUT	S / 4mA	SSP3 Serial Clock
89	C7	STXD3	PD.3	IN/OUT	4mA	SSP3 Transmit Data
90	F9	TIMER_1	PD.4	IN/OUT	S / 4mA	Timer 1 I/O function pin
91	F8	TIMER_2	PD.5	IN/OUT	S / 4mA	Timer 2 I/O function pin
92	E8	STXD4	PD.6	IN/OUT	4mA	SSP4 Transmit Data
93	NA	OSC_RTC_OUT		OUT		RTC Oscillator Output
94	NA	OSC_RTC_IN		IN		RTC Oscillator Input
95	D9	OSC_SYS_OUT		OUT		System Oscillator Output
96	E9	OSC_SYS_IN		IN		System Oscillator Input
97	B9	AVDD_PLL		SUPPLY		PLL Analog VDD supply
98	C9	AVSS_PLL		SUPPLY		PLL Analog VSS supply
99	C8	DVDD_PLL		SUPPLY		PLL Digital VDD supply
100	D8	DVSS_PLL		SUPPLY		PLL Digital VSS supply

1) Pin not connected on PUC 3033A devices

6.3. Pin Descriptions

Note: All power supply pins must be connected. This is mandatory for the proper functioning of the PUC 303xA.

Pin Name	Type	Function	Comment
Power supply pins			
Core Supply			
VSS_CORE		2.5V Supply	The 3 VDD/VSS pairs are used to power the internal core logic
VDD_CORE		2.5V Supply	
PLL Supply			
DVSS_PLL		2.5V Supply	The DVSS/DVDD pair is used to power the digital section of the PLL.
DVDD_PLL		2.5V Supply	
AVSS_PLL		2.5V Supply	The AVSS/AVDD pair is used to power the analog section of the PLL.
AVDD_PLL		2.5V Supply	
I/O Supply			
VSS_IO		3.3V Supply	The 3 VSS/VDD pairs are used to power the I/O ring
VDD_IO		3.3V Supply	
USB Supply			
VSS_USB		3.3V Supply	This VSS/VDD pair is used to power the on-chip USB transceiver cell
VDD_USB		3.3V Supply	
Dedicated function pins			
System Pins			
nRESET	IN	Active low system reset pin	Assert ’0’ at power-on, warmstart or to return from OFF state
MODSEL_0	IN	PUC 303xA mode select bit 0	Tie to ’0’ for normal operation mode; ’1’ is reserved for test modes
MODSEL_1	IN	PUC 303xA mode select bit1	Tie to ’0’ for normal operation mode; ’1’ is reserved for test modes
OSC_SYS_IN	IN	System clock oscillator input	Terminal to connect crystal or external clock source
OSCSYS_OUT	OUT	System clock oscillator output	Terminal to connect crystal
OSC_RTC_IN	IN	Real Time Clock oscillator input	Terminal to connect crystal or external clock source. Available in 100 pin package only.
OSC_RTC_OUT	OUT	Real Time Clock oscillator output	Terminal to connect crystal
WAKE_UP	IN	To exit STANDBY mode	Available in 100 pin package only

Pin Name	Type	Function	Comment
CLKOUT	OUT	Clock output	Clock for external device, e.g. Micronas MAS 35xxF.
USB Transceiver			
USB_DPLUS	IN/OUT	Bidirectional USB transceiver D^+	
USB_DMINUS	IN/OUT	Bidirectional USB transceiver D^-	
JTAG Interface			
nTRST	IN	Active low JTAG reset	
TCK	IN	JTAG clock	
TDI	IN	JTAG data in	
TMS	IN	JTAG test mode select	
TDO	OUT	JTAG data out	
Shared GPIO / Special function pins			
UART Interfaces			
RXDy	IN/OUT	UART serial data in	substitute “y” by “1” or “2”
nCTSy	IN/OUT	Active low clear to send input	
nDSRy	IN/OUT	Active low data set ready input	
nDCDy	IN/OUT	Active low data carrier detect input	
nRTSy	IN/OUT	Active low request to send output	
nDTRy	IN/OUT	Active low data terminal ready output	
nRIy	IN/OUT	Active low ring indicator input	
TXDy	IN/OUT	UART serial data out	
Synchronous Serial Interfaces			
SRXDy[3)]	IN/OUT	Serial data input	[3)]substitute “y” by “1”, “2”, “3”, or “4
STXDy[3)]	IN/OUT	Serial data output	
SCLKy[3)]	IN/OUT	Serial data clock, input/output	
SFRMy[3)]	IN/OUT	Serial data framing, input/output	
I2S_AUDIOCLK	IN/OUT	Clock input for I^2S interface	Enables fast transmission synchronous to external audio device, e.g. MAS 35xxF
I2S_RXD	IN/OUT	Serial data input	
I2S_TXD	IN/OUT	Serial data output	
I2S_FRM	IN/OUT	Serial data framing, input/output	
I2S_CLK	IN/OUT	Serial data clock, input/output	

Pin Name	Type	Function	Comment
USB Interface			
USB_SENSE	IN/OUT	USB interface Sense input	
USB_PULLUP	IN/OUT	USB interface Pullup output	This GPIO pin is configured as output on reset
Timer Interfaces			
TIMER_1	IN/OUT	Timer I/O pins (timer 1)	Sample or generate waveforms
TIMER_2	IN/OUT	Timer I/O pins (timer 2)	Sample or generate waveforms
I^2C Interfaces			
I2C_SDA	IN/OUT	I^2C data	Typically used as I^2C master interface to control external device, e.g. MAS 35xxF
I2C_SCL	IN/OUT	I^2C clock	
UC_SDA	IN/OUT	I^2C data	I^2C interface managed by USB core's embedded controller
UC_SCL	IN/OUT	I^2C clock	
General Purpose I/O pins without special function			
PORTF_x[1)]	IN/OUT	Dedicated GPIO (no special function)	[1)]substitute "x" by "0", "1"
PORTG_y[2)]	IN/OUT	Dedicated GPIO (no special function)	[2)]subst. "y" by "0", "1", "2", "3", or "4"
PORTH_z[3)]	IN/OUT	Dedicated GPIO (no special function)	[3)]substitute "z" by "0", "1", "2", "3", "4", "5", "6", or "7"
PORTI_z[3)]	IN/OUT	Dedicated GPIO (no special function)	

6.4. Electrical Characteristics

6.4.1. Absolute Maximum Ratings

Symbol	Parameter	Value	Unit
V_{DD_CORE}	Core Supply Voltage	2.9	V
DV_{DD_PLL}	Digital PLL Supply Voltage	2.9	V
AV_{DD_PLL}	Analog PLL Supply Voltage	2.9	V
V_{DD_IO}	I/O Supply Voltage	4.6	V
V_{DD_USB}	USB Transceiver Supply Voltage	tbd.	V
T_S	Storage Temperature	-40 to 150	^oC
Note: Stresses beyond those listed in the “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only. Functional operation of the device at these or any other conditions beyond those indicated in the “Recommended Operating Conditions/Characteristics” of this specification is not implied. Exposure to absolute maximum ratings conditions for extended periods may affect device reliability.			

6.4.2. Recommended Operating Conditions

Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
V_{DD_CORE}	Core Supply Voltage	2.25	2.5	2.75	V	
DV_{DD_PLL}	Digital PLL Supply Voltage	2.25	2.5	2.75	V	
AV_{DD_PLL}	Analog PLL Supply Voltage	2.25	2.5	2.75	V	
V_{DD_IO}	I/O Supply Voltage	3.0	3.3	3.6	V	
V_{DD_USB}	USB Transceiver Supply Voltage	3.0	3.3	3,6	V	
CLK_F	System Oscillator Clock Frequency	10	14	20	MHz	
V_{IL}	Input Low Voltage	-0.5		1.0	V	
V_{IH}	Input High Voltage	1.8		5.5	V	
V_{OL}	Output Low Voltage			0.4	V	@I_{OL}=10mA
V_{OH}	Output High Voltage	2.4			V	@I_{OH}=18mA
I_{OL}	Low Level Output Current	6.6	10.8	13.1	mA	@V_{OL}=0.4V
I_{OH}	High Level Output Current	8.3	17.9	28.6	mA	@V_{OH}=2.4V

6.4.3. DC Characteristics

6.4.3.1. Hysteresis of Schmitt Trigger inputs

Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
V_T^H	Hysteresis	0.4	-	1.2	V	T = 25°C V_{DD_IO} = 3.3V V_{DD_CORE} = 2.5V
V_T^+	Low to High treshold voltage	1.75	1.95	2.25	V	
V_T^-	High to Low treshold voltage	1.05	1.25	1.35	V	

6.4.3.2. Internal Pull-up / Pull-down Resistor Values

Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
R_U	Internal pull-up resistor	74	103	173	KΩ	
R_D	Internal pull-down resistor	43	66	129	KΩ	

6.4.4. AC Characteristics

6.4.4.1. SSP slave (SCLKx and SFRMx are inputs)

Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
t_{setup}	SRXDx, SFRMx setup time relative to sampling edge of SCLKx	1.6	-	-	ns	
t_{hold}	SRXDx, SFRMx hold time relative to sampling edge of SCLKx	5.9	-	-	ns	
t_{del}	STXDx propagation delay relative to received edge of SCLKx	-	-	116.3	ns	f_ssp(x)clk = 48.0 MHz; f_SCLKx = f_ssp(x)clk/12 = 4.0 MHz

6.4.4.2. SSP master (SCLKx and SFRMx are outputs)

Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
t_{OE}	STXDx enable time relative to SFRMx	-	-	9.8	ns	
t_{OD}	STXDx disable time relative to SFRMx	-	-	4.9	ns	
t_{hold}	STXDx, SFRMx hold time relative to sampling edge of SCLKx	15.8	-	-	ns	
t_{del}	STXDx propagation delay relative to generating edge of SCLKx	-	-	12.1	ns	
t_{Hi}	Minimum SCLKx high time	18.1	-	-	ns	f_SCLKx = f_ssp(x)clk/2 = 24 MHz
t_{Lo}	Minimum SCLKx low time	22.5	-	-	ns	

6.4.4.3. AC Characteristics for CLKOUT pin

Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
t_{Hi}	Minimum CLKOUT high time	37.4	-	-	ns	f_CLKOUT = f_oscclk = 13 MHz
t_{Lo}	Minimum CLKOUT low time	39.2	-	-	ns	
t_{del}	Delay relative to system clock input (oscclk)	8.6	-	14.4	ns	CLKOUT fed from oscclk not pllclk

6.4.4.4. AC Characteristics for WAKEUP pin

Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
t_{Hi}	Minimum WAKEUP high time for rising edge detection	1	-	-	oscrtcclk cycles	

6.4.4.5. AC Characteristics for TIMER_x inputs

Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
t_{Imp}	Minimum impulse length for detection	1	-	-	pclk_apb cycles	

6.4.4.6. AC Characteristics for GPIO inputs

Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
t_{Hi}	Minimum impulse high time for level or edge detection	1	-	-	pclk_apb cycles	
t_{Lo}	Minimum impulse low time for level or edge detection	1	-	-	pclk_apb cycles	

6.4.4.7. Rise/Fall times of 4mA and 12mA outputs

Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
t_{rise4}	Rise time of 4mA output	-	3.9	-	ns	C_L = 30 pF T = 25°C VDD_{IO} = 3.3V VDD_{core} = 2.5V
t_{fall4}	Fall time of 4mA output	-	3.0	-	ns	
t_{rise12}	Rise time of 12mA output	-	3.6	-	ns	
t_{fall12}	Fall time of 12mA output	-	3.5	-	ns	

6.4.5. Flash Programming Characteristics

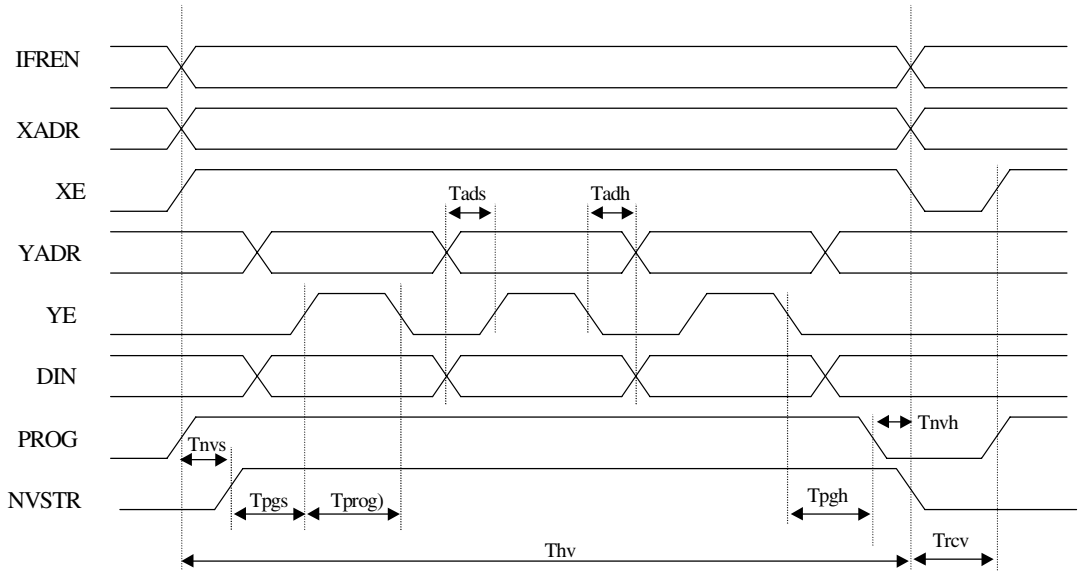


Fig. 6–3: Typical Flash Programming Cycle*

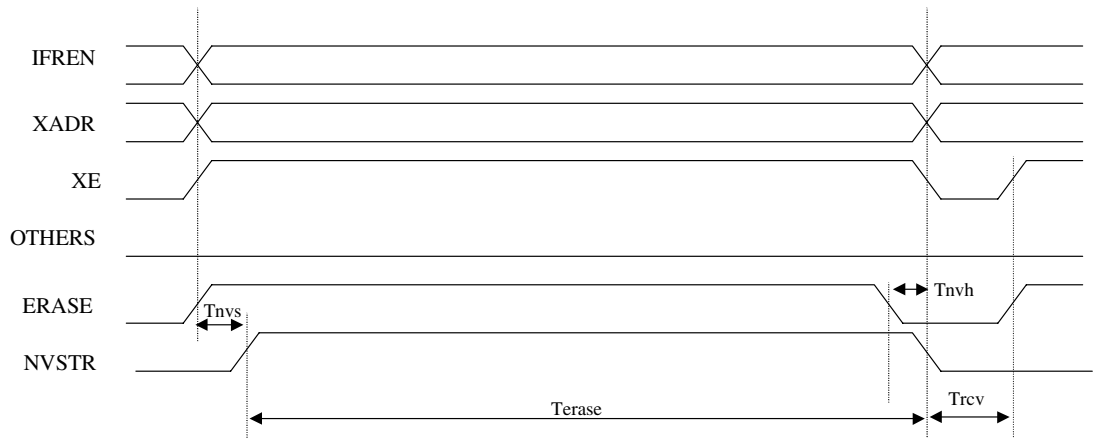


Fig. 6–4: Typical Flash Page Erase Cycle

* Programming of more than one word per program cycle is available via Parallel Flash interface

Table 6–1: Flash Programming Parameters

Symbol	Parameter	Min	Max	Units
Tnvs	PROG/ERASE to NVSTR	5	-	µs
Tnvh	Nvstr Hold Time	5	-	µs
Tpgs	Nvstr to Program Setup Time	10	-	µs
Tpgh	Program Hold Time	20	-	ns
Tprog	Program Time	20	40	µs
Tads	Address/data Setup Time	20	-	ns
Tadh	Address/data Hold Time	20	-	ns
Trcv	Recovery Time	1	-	µs
Thv	Cumulative high voltage programming time within the same row before next erase. The same address must not be programmed more than twice before next erase.	-	8	ms
Terase	ERASE Time	4	5	ms

7. List of Abbreviations

Abbreviation	Description
AHB	Advanced High-performance Bus; an AMBA bus
AMBA	Advanced Microcontroller Bus Architecture; a bus architecture specification from ARM Limited
APB	Advanced Peripheral Bus; an AMBA bus
ARM7TDMI	An ARM Processor Core developed by ARM Limited
CPSR	Current Processor Status Register; a status register on the ARM7TDMI core
CPU	Central Processing Unit; the ARM7TDMI core
DRM	Digital Rights Management
EP	Endpoint; source or sink of data on a USB peripheral
FIFO	First In First Out; a buffer strategy
FIQ	Fast Interrupt Request; an interrupt type with low latency on the ARM7TDMI core
GPIO	General Purpose Input/Output; the behavior of associated I/O-pins is programmable
I^2C	A serial control bus specification originated by Philips
I^2S	A serial bus specification originated by Philips; term also used for a variant by Sony
IrDA	Infrared Data Association; term is used for the association's infrared communication standards
IRQ	Interrupt Request; an interrupt type on the ARM7TDMI core
JTAG	Joint Test Access Group; a serial access protocol widely used for debugging and boundary scan.
LSB	Least Significant Bit; bit [0] within a data item
MPU	Memory Protection Unit
PLL	Phase-locked Loop
MSB	Most Significant Bit; bit with highest weight within a data item
PMU	Power Management Unit
PUC	Programmable Universal Controller
SIE	Serial Interface Engine; a serial/parallel conversion block within a USB peripheral
RTC	Real Time Clock
SPI	A serial protocol originated by Motorola
SPSR	Saved Processor Status Register; a status register on the ARM7TDMI core
SSP	Synchronous Serial Port; peripheral hardware for serial communication
TM	Test Mode
TPC	Test Port Controller; used for production testing
TTC	Triple Timer Counter module
UART	Universal Asynchronous Receiver Transmitter; a serial interface
USB	Universal Serial Bus
WDT	Watchdog Timer module

8. Data Sheet History

1. Advance Information: “PUC 303xA Programmable Universal Controller “Zenon”, Aug. 13, 2001. First release of the advance information.

Micronas GmbH
Hans-Bunte-Strasse 19
D-79108 Freiburg (Germany)
P.O. Box 840
D-79008 Freiburg (Germany)
Tel. +49-761-517-0
Fax +49-761-517-2174
E-mail: docservice@micronas.com
Internet: www.micronas.com

Printed in Germany
Order No. 6251-565-1AI