CoreUARTapb v5.2 Handbook

Microsemi

Table of Contents

Introduction

General Description

CoreUARTapb is a serial communication controller with a flexible serial data interface that is intended primarily for embedded systems. CoreUARTapb can be used to interface directly to industry standard UARTs. CoreUARTapb is intentionally a subset of full UART capability to make the function cost-effective in a programmable device.

Core Version

This handbook applies to CoreUARTapb v5.2. The release notes provided with the core list known discrepancies between this handbook and the core release.

Supported Families

- IGLOO®
- IGLOOe
- IGLOO PLUS
- ProASIC®3
- ProASIC3E
- ProASIC3L
- SmartFusion®
- SmartFusion2
- Fusion
- ProASIC^PLUS®
- Axcelerator®
- RTAX-S
- SX-A
- RTSX-S

1 – Functional Block Description

Figure 1-1 shows the block diagram of the CoreUARTapb normal mode functionality. Figure 1-2 on page 6 shows the block diagram of CoreUARTapb with FIFO mode functionality. The baud generator creates a divided down clock enable that correctly paces the transmit and receive state machines.

The function of the receive and transmit state machines is affected by the control inputs BIT8, PARITY_EN, and ODD_N_EVEN. These signals indicate to the state machines how many bits should be transmitted or received. In addition, the signals suggest the type of parity and whether parity should be generated or checked. The activity of the state machines is paced by the outputs of the baud generator.

To transmit data, it is first loaded into the transmit data buffer in normal mode, and into the transmit FIFO in FIFO mode. Data can be loaded into the data buffer or transmit FIFO until the TXRDY signal is driven inactive. The transmit state machine will immediately begin to transmit data and will continue transmission until the data buffer is empty in normal mode, and until the transmit FIFO is empty in FIFO mode. The transmit state machine first transmits a START bit, followed by the data (LSB first), then the parity (optional), and finally the STOP bit. The data buffer is double-buffered in normal mode, so there is no loading latency.

The receive state machine monitors the activity of the RX signal. Once a START bit is detected, the receive state machine begins to store the data in the receive buffer in normal mode and the receive FIFO in FIFO mode. When the transaction is complete, the RXRDY signal indicates that valid data is available. Parity errors are reported on the PARITY_ERR signal (if enabled), and data overrun conditions are reported on the OVERFLOW signal. Framing errors are reported on the FRAMING_ERR signal.
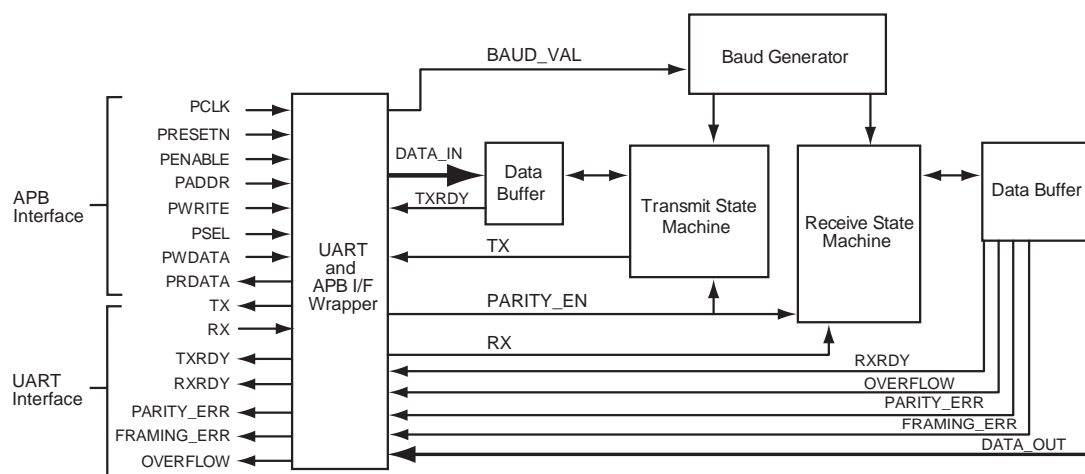
Figure 1-1 • **Block Diagram of CoreUARTapb Normal Functionality**
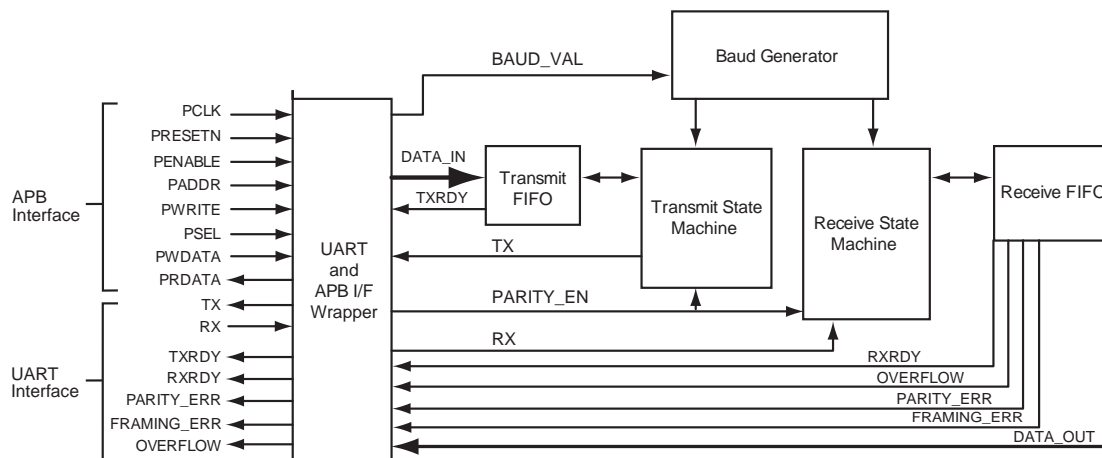
Figure 1-2 • **Block Diagram of CoreUARTapb with FIFO Functionality**

Device Utilization and Performance

Utilization statistics for targeted devices are listed in Table 1-1 through Table 1-2 on page 7.

Table 1-1 • **CoreUARTapb Utilization in FIFO Mode**

| Family | Cells or Tiles | | | Memory Blocks | Utilization | | Performance MHz |
	Sequential	Combinatorial	Total		Device	Total	
IGLOO IGLOOe IGLOO PLUS	140	240	380	2	AGL600V5	3%	67
ProASIC3 ProASIC3E ProASIC3L	140	240	380	2	M7A3P250	6%	131
SmartFusion	139	248	387	2	A2F500M3F	8%	117
SmartFusion2	TBD	TBD	TBD	TBD	TBD	TBD	TBD
Fusion	140	240	380	2	AFS600	3%	125
ProASIC<u>PLUS</u>	142	347	489	2	APA075	16%	72
Axcelerator®	194	237	431	2	AX250	10%	166
RTAX-S	222	216	438	2	RTAX250S	10%	153
SX-A	430	309	739	0	A54SX16A	51%	96
RTSX-S	432	308	740	0	RT54SX32S	26%	62

Notes:

1. *CoreUARTapb supports all standard baud rates, including 110, 300, 1,200, 2,400, 4,800, 9,600, 19,200, 38,400, 57,600, 115,200, 230,400, 460,800, and 921,600 baud.*

2. *The depth of the FIFO for SX-A and RTSX-S families is 16. The depth of the FIFO for SmartFusion2 devices is 128. For the other families, the depth of the FIFO is 256.*

3. *The numbers above reflect CoreUARTapb in programming mode.*

4. *Performance numbers are for –2 speed grade for each device in commercial range operating conditions.*

Table 1-2 • **CoreUARTapb Utilization in Normal Mode**

| Family | Cells or Tiles | | | Memory Blocks | Utilization | | Performance |
	Sequential	Combinatorial	Total		Device	Total	MHz
IGLOO IGLOOe IGLOO PLUS	108	213	321	0	AGL600	2%	128
ProASIC3 ProASIC3E ProASIC3L	108	213	321	0	M7A3P250	5%	197
SmartFusion	108	220	328	0	A2F500M3F	7%	200
SmartFusion2	TBD	TBD	TBD	TBD	TBD	TBD	TBD
Fusion	108	213	321	0	AFS600	2%	193
ProASIC^{PLUS}	109	322	431	0	APA075	14%	82
Axcelerator	109	133	242	0	AX250	5%	215
RTAX-S	109	133	242	0	RTAX250S	5%	153
SX-A	82	93	175	0	A54SX16S	12%	138
RTSX-S	80	92	172	0	RT54SX32S	6%	87

Notes:

1. *CoreUARTapb supports all standard baud rates, including 110, 300, 1,200, 2,400, 4,800, 9,600, 19,200, 38,400, 57,600, 115,200, 230,400, 460,800, and 921,600 baud.*

2. *The numbers above reflect CoreUARTapb in programmable mode.*

3. *Performance numbers are for –2 speed grade for each device in commercial range operating conditions.*

CoreUARTapb supports two modes: programmable and fixed. These modes enable the user to set parameters as fixed or as configurable during system operation.

Fixed Mode Options

There are four options in Fixed mode CoreUARTapb operation:

1. Character size
2. Parity
3. Baud rate
4. Fractional part of baud value

These values are hardwired and cannot be changed during runtime.

Character Size

The default value for the number of data bits is 7. The option PRG_BIT8 sets the serial bitstream to 8-bit data mode.

Parity

The PRG_PARITY parameter sets the parity enabled/disabled. It also sets parity even/odd.

Baud Rate

This baud value is a function of the system clock and the desired baud rate. The value should be set according to EQ 1-1.

$$\text{baud rate} = \frac{\text{clk}}{(\text{baudval} + 1) \times 16}$$

<div align="right">*EQ 1-1*</div>

where

$$\text{clk} = \text{the frequency of the system clock in hertz}$$

$$\text{baud rate} = \text{the desired baud rate}$$

and

$$\text{baudval} = \left(\frac{\text{clk}}{16 \times \text{baudrate}}\right) - 1$$

<div align="right">*EQ 1-2*</div>

The term baudval must be rounded to the nearest integer. For example, a system with a 33 MHz system clock and a desired baud rate of 9,600 should have a baud_value of 214 decimal or D6 hex. So, to get the desired baud rate, the user should assign 0xD6 to the baud_value (by writing appropriate values to Control Register 1 and Control Register 2).

Fractional Part of Baud Value

The BAUD_VAL_FRCTN parameter sets the fractional part of baud value. This option is only available if Enable Extra Precision is selected and in Fixed Mode. The baud value can be set with a precision of 0.125.

For example, a system with a 24 MHz system clock and a desired baud rate of 230,400 should have a baud_value of 5.51 decimal. Rounding the baud_value to the nearest integer, which is 6 decimal in this case, causes the percentage error to be higher than the allowed error of approx 4.54% . So, to get the desired baud rate, the user should assign 5 decimal to BAUD_VAL input, select **Enable Extra Precision** and assign the integer 4 to the BAUD_VAL_FRCTN parameter to achieve better precision (refer to Table 3-3 on page 15).

2 – Tool Flows

Licensing

CoreUARTapb is licensed in two ways. Depending on your license tool flow, functionality may be limited.

Obfuscated

Complete RTL code is provided for the core, allowing the core to be instantiated with SmartDesign. Simulation, Synthesis, and Layout can be performed within Libero® Integrated Design Environment (IDE). The RTL code for the core is obfuscated[1] and some of the testbench source files are not provided; they are precompiled into the compiled simulation library instead.

RTL

Complete RTL source code is provided for the core and testbenches.

---

1.   *Obfuscated means the RTL source files have had formatting and comments removed, and all instance and net names have been replaced with random character sequences.*

SmartDesign

CoreUARTapb is available for download in the SmartDesign IP deployment design environment. The core can be configured using the configuration GUI within SmartDesign, as shown in Figure 2-1 on page 10.

For more information on using SmartDesign to instantiate and generate cores, refer to the Using DirectCore in Libero® IDE User's Guide.
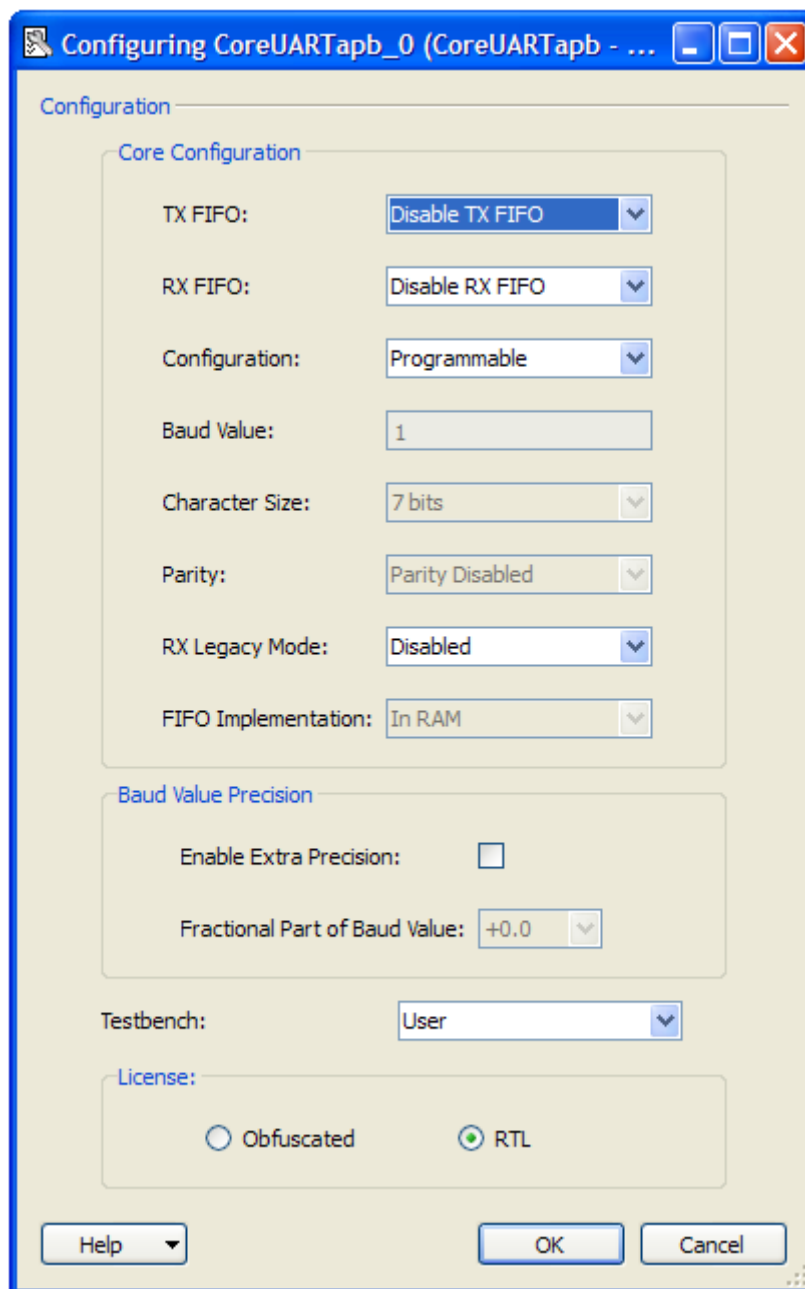
Figure 2-1 • **SmartDesign CoreUARTapb Configuration Window**

Simulation Flows

The user testbench for CoreUARTapb is included in all releases.

To run simulations, select the user testbench flow within SmartDesign and click **Generate Design** under the SmartDesign menu. The user testbench is selected through the Core Testbench Configuration GUI.

When SmartDesign generates the Libero IDE project, it will install the user testbench files.

To run the user testbench, set the design root to the CoreUARTapb instantiation in the Libero IDE Design Hierarchy pane and click the Simulation icon in the Libero IDE Design Flow window. This will invoke ModelSim® and automatically run the simulation.

User Testbench

An example user testbench is included with CoreUARTapb for both VHDL and Verilog. The testbench is provided as an obfuscated bus functional model (BFM) connected as shown in Figure 2-2 to two CoreUARTapb blocks that are in turn connected via their serial UART interfaces. The user can examine and change the testbench by modifying the *.bfm file and generating a *.vec APB master vector file, as shown in Figure 2-2.
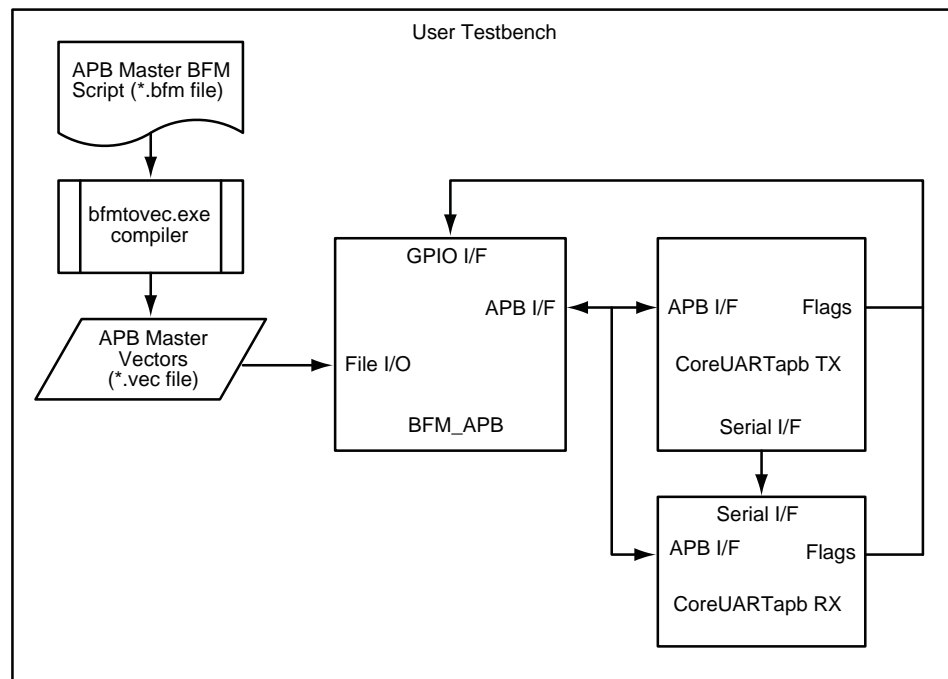
***Figure 2-2 •* CoreUARTapb User Testbench**

As shown in Figure 2-2, the user testbench instantiates a Microsemi DirectCore AMBA bus functional model (BFM) module to emulate an APB master that controls the operation of CoreUARTapb via reads and writes to access internal registers. A BFM ASCII script source file with comments is included in the directory <proj>/simulation, where <proj> represents the path to your Libero IDE project. The BFM source file, coreuartapb_usertb_apb_master.bfm, controls the APB master processor. This BFM source file is automatically recompiled each time the simulation is invoked from Libero IDE by the bfmtovec.exe executable, if running on a Windows® platform, or by the bfmtovec.lin executable, if running on a Linux® platform. The coreuartapb_usertb_apb_master.vec vector file, created by the bfmtovec executable, is read in by the BFM module for simulation in ModelSim.

You can alter the BFM script, if desired. Refer to the *DirectCore AMBA BFM User's Guide* for more information.

Synthesis in Libero IDE

Click the **Synthesis** icon in Libero IDE. The Synthesis window appears, displaying the Synplify$^®$ project. Set Synplify to use the Verilog 2001 standard if Verilog is being used. To run Synthesis, select the **Run** icon.

Place-and-Route in Libero IDE

Click the **Layout** icon in Libero IDE to invoke Designer. CoreUARTapb requires no special place-and-route settings.

3 – Core Interfaces

Signal descriptions for CoreUARTapb are defined in Table 3-1. The APB interface allows access to the CoreUARTapb internal registers, FIFO, and internal memory. This interface is synchronous to the clock.

Table 3-1 • **CoreUARTapb Signals**

Name*	Type	Description
PCLK	In	Master clock input
PRESETN	In	Active low asynchronous reset
PWRITE	In	APB write/read enable, active high
PADDR[4:2]	In	APB address
PSEL	In	APB select
PENABLE	In	APB enable
PWDATA[7:0]	In	APB data input
PRDATA[7:0]	Out	APB data output
TXRDY	Output	Status bit; when set to logic 0, indicates that the transmit data buffer/FIFO is not available for additional transmit data.
RXRDY	Output	Status bit; when set to logic 1, indicates that data is available in the receive data buffer/FIFO to be read by the system logic. The data buffer must be read through APB via the Receive Data Register (0x04) to prevent an overflow condition from occurring.
PARITY_ERR	Output	Status bit; when set to logic 1, indicates a parity error during a receive transaction. When RX FIFO is enabled, this bit is self clearing between bytes. Otherwise, this bit is synchronously cleared by performing a read operation on the Receive Data Register via the APB slave interface.
FRAMING_ERR	Output	Status bit; when set to logic 1, indicates a framing error (that is, a missing stop bit) during the last received transaction. When RX FIFO is enabled, this bit is self clearing between bytes. Otherwise, this bit is synchronously cleared by performing a read operation on the Receive Data Register via the APB slave interface.
OVERFLOW	Output	Status bit; when set to logic 1, indicates that a receive overflow has occurred. This bit is synchronously cleared by performing a read operation on the Receive Data Register via the APB slave interface.
RX	Input	Serial receive data
TX	Output	Serial transmit data

*Note: *All signals are active high unless otherwise indicated.*

Core Parameters

CoreUARTapb Configurable Options

There are a number of configurable options that apply to CoreUARTapb, as shown in Table 3-2. If a configuration other than the default is required, the user should use the configuration dialog box in SmartDesign to select appropriate values for the configurable options.

Table 3-2 • **CoreUARTapb Configurable Options**

Configurable Options	Default Setting	Description
TX_FIFO	Disable TX_FIFO	Enables or disables transmit FIFO
RX_FIFO	Disable RX_FIFO	Enables or disables receive FIFO
FAMILY	ProASIC3	Selects target family. Must be set to match the supported FPGA family. 8 – 54SXA 9 – RTSXS 11 – Axcelerator 12 – RTAX-S 14 – ProASIC^{PLUS} 15 – ProASIC3 16 – ProASIC3E 17 – Fusion 18 – SmartFusion 19 – SmartFusion2 20 – IGLOO 21 – IGLOOe 22 – ProASIC3L 23 – IGLOO PLUS
FIXEDMODE	Programmable	0 – Programmable 1 – Fixed Fixed or Programmable mode. In Fixed mode, the parameters BAUD_VALUE, Character Size, and Parity are hardwired. In Programmable mode they are programmed by the control registers.
BAUD_VALUE	1	Baud value is set only when configuration is set to fixed mode.
BAUD_VAL_FRCTN	+0.0	This parameter is only relevant when the parameter FIXEDMODE is set to Fixed and parameter BAUD_VAL_FRCTN_EN has been enabled. The value chosen here is added to the baud value to give a precise baud value.

Table 3-2 • **CoreUARTapb Configurable Options (continued)**

Configurable Options	Default Setting	Description
BAUD_VAL_FRCTN_EN	Disabled	When parameter FIXEDMODE is set to Programmable, enabling this parameter enables an additional control register (Control Register 3) that can be used to set a fractional part for the baud value. The baud value can be set with a precision of 0.125. When parameter FIXEDMODE is set to Fixed, enabling this parameter allows you to set a fixed fractional part for the baud value. The size of the fractional part is specified by the BAUD_VAL_FRCTN parameter.
PRG_BIT8	7 bits	This option can only be set when configuration mode is set to fixed mode. This option defines the number of valid data bits in the serial bitstream. Character size can be 8 bits or 7 bits.
PRG_PARITY	Parity disabled	This option can only be set when configuration mode is set to Fixed mode. The options for parity are as follows: Parity Disable, Even Parity, or Odd Parity.
RX_LEGACY_MODE	Disabled	When disabled, the RXRDY signal is synchronized with the FRAMING_ERR output, which occurs after the stop bit. When enabled (legacy mode), the RXRDY signal is asserted after all data bits have been received, but before the stop bit.
USE_SOFT_FIFO	Disabled	When disabled, the FIFO is implemented using a device-specific hard macro. When enabled, a 16-byte FIFO is implemented in FPGA logic instead. 54SXA and RTSX-S devices use this soft FIFO by default.

Table 3-3 shows the fraction that baud value will be modified by when in Fixed Mode and BAUD_VAL_FRCTN is used.

Table 3-3 • **Baud Value Fraction**

| BAUD_VAL_FRCTN	Precision
0	+0.0
1	+0.125
2	+0.25
3	+0.375
4	+0.5
5	+0.625
6	+0.75
7	+0.875

4 – Timing Diagrams

The UART waveforms can be broken down into a few basic functions: transmit data, receive data, and errors. Figure 4-1 shows serial transmit signals, and Figure 4-2 on page 17 shows serial receive signals. Figure 4-3 on page 18 and Figure 4-4 on page 18 show the parity and overflow error cycles, respectively. The number of clock cycles required is equal to the clock frequency divided by the baud rate.
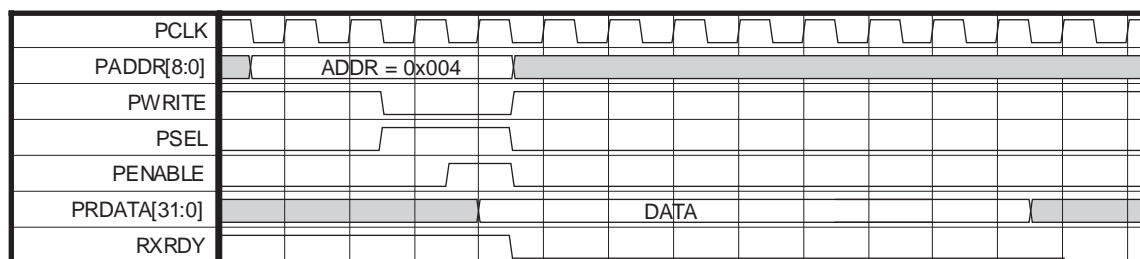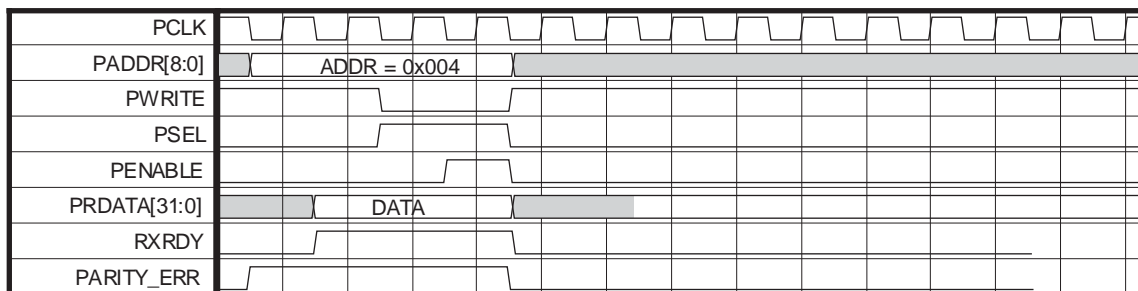
Serial Transmit
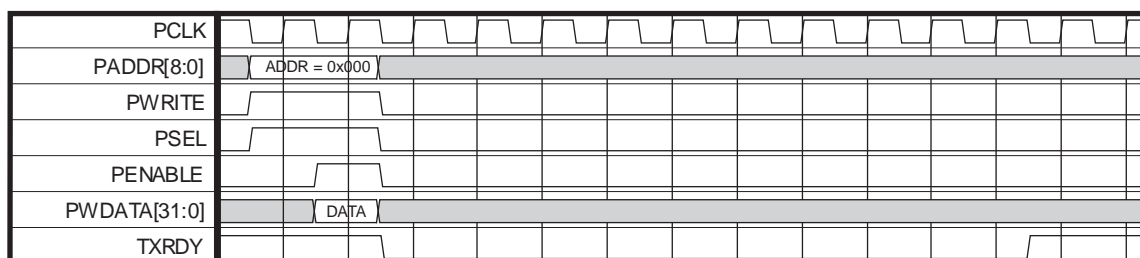
Figure 4-1 • **Serial Transmit**

Note: A serial transmit is initiated by writing data into CoreUARTapb. This is accomplished by providing valid data and asserting the PWRITE, PSEL, and PENABLE signals.
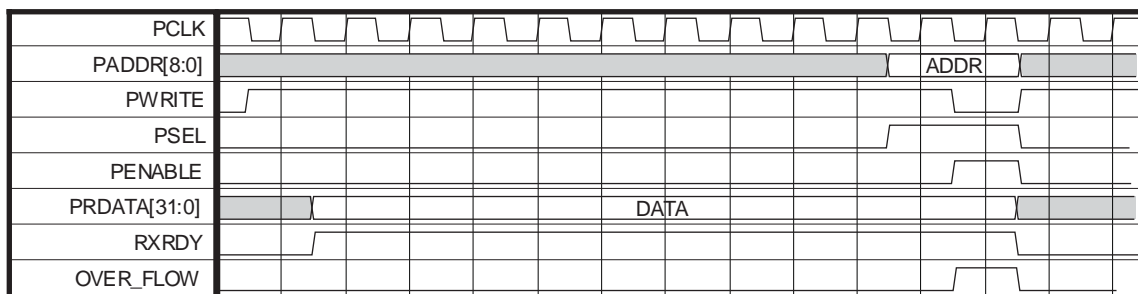
Serial Receive

Figure 4-2 • **Serial Receive**

Parity Error

Figure 4-3 • **Parity Error**

Note: When a parity error occurs (mismatch in parity between transmitted data and receiver), PARITY_ERR will be asserted in the receiver. To clear the PARITY_ERR signal, as shown, simply perform a read operation on the receive data register.
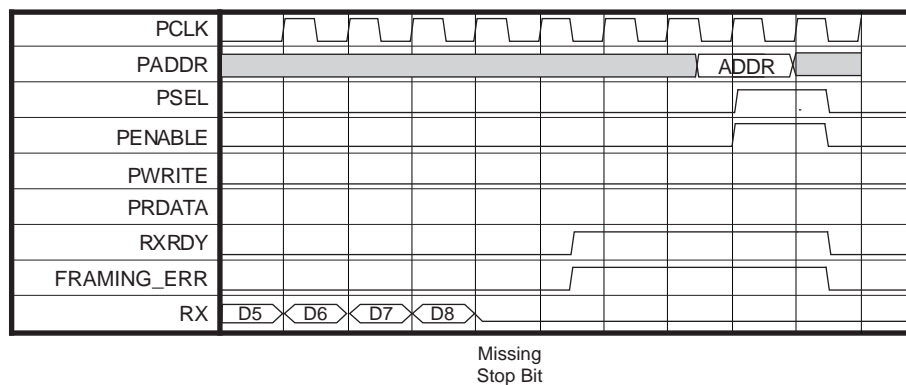
Overflow Error

Figure 4-4 • **Overflow Error**

Note: When a data overflow error occurs, the overflow signal is asserted.

Framing Error (no legacy mode)

Figure 4-5 • **Framing Error (no legacy mode)**
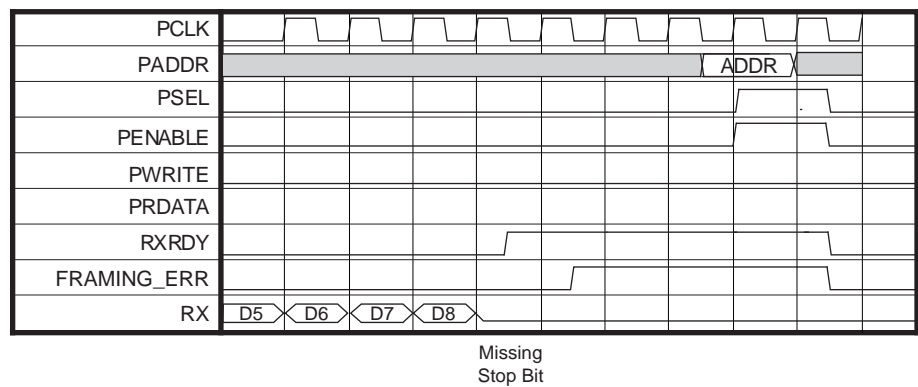
Framing Error (legacy mode)

Figure 4-6 • **Framing Error (legacy mode)**

5 – CoreUARTapb Configuration Registers

CoreUARTapb Programmer's Model

Table 5-1 lists the registers for CoreUARTapb.

Table 5-1 • **CoreUARTapb Registers**

Address	Type	Width	Reset Value	Name	Description
Base + 0x000	Write	8	0x00	TxData	Transmit Data register
Base + 0x004	Read	8	0x00	RxData	Receive Data register
Base + 0x008	Read/Write	8	0x00	Ctrl1	Control register 1
Base + 0x00C	Read/Write	8	0x00	Ctrl2	Control register 2
Base + 0x010	Read	8	0x01	Status	Status register
Base + 0x014	Read/Write	3	0x00	Ctrl3	Control register 3

Transmit Data Register

The Transmit Data Register contains the 7- or 8-bit transmit data.

Receive Data Register

The Receive Data Register contains the 7- or 8-bit receive data.

Control Register 1

Control Register 1 contains a single field, baud value, used to set the baud rate for CoreUARTapb. The baud value should be set according to EQ 5-1:

$$\text{baud val} = \frac{clk}{16 \times \text{baud rate}} - 1$$

EQ 5-1

where clk is the system clock frequency in hertz.

The result of this calculation must be rounded to the nearest integer and converted to hexadecimal to obtain the value that should be written to Control register 1 and Control register 2, shown in Table 5-1. For example, when the clock frequency is 10 MHz and a baud rate of 9,600 is desired, 0x40 should be written to Control register 1 and 0x00 should be written to Control register 2. When the clock frequency is 50 MHz and a baud rate of 381 is desired, 0xFF should be written to Control register 1, and 0x1F should be written to the top 5 bits of Control register 2.

Table 5-2 • **Control Register 1**

Bit(s)	Name	Type	Function
7:0	Baud value	Read/write	Bits 7:0 of 13-bit baud value

Control Register 2

Table 5-3 shows Control Register 2, which is used to assign values to the configuration inputs available on CoreUARTapb.

Table 5-3 • **Control Register 2**

Bit(s)	Name	Type	Function
0	BIT8	Read/write	Data width setting: BIT8 = 0: 7-bit data BIT8 = 1: 8-bit data
1	PARITY_EN	Read/write	Parity is enabled when this bit is set to 1.
2	ODD_N_EVEN	Read/write	Parity is set as follows: ODD_N_EVEN = 0: even ODD_N_EVEN = 1: odd
7:3	BAUD_VALUE	Read/write	Bits 12:8 of 13-bit baud value

Control Register 3

Table 5-4 shows Control Register 3, which is used to assign values to the configuration inputs available on CoreUARTapb.

Table 5-4 • **Control Register 3**

Bit(s)	Name	Type	Function
2:0	BAUD_VAL_FRACTION	Read/write	When Configuration is set to Programmable, this register can be used to set a fractional part for the baud value. The baud value can be set with a precision of 0.125.

Note: *BAUD_VAL_FRCTN_EN must be enabled to enable this register.*

Set the fractional part of the baud value according to Table 5-5.

Table 5-5 • **Fractional Baud Value Settings**

Bit(s)	Extra Precision
000	+0.0
001	+0.125
010	+0.25
011	+0.375
100	+0.5
101	+0.625
110	+0.75
111	+0.875

Status Register

Table 5-6 shows the Status Register, which provides information on the status of CoreUARTapb.

Table 5-6 • **Status Register**

Bit(s)	Name	Type	Function
0	TXRDY	Read only	When Low, the transmit data buffer/FIFO is not available for additional transmit data.
1	RXRDY	Read only	When High, data is available in the receive data buffer/FIFO. This bit is cleared by reading the Receive Data Register.
2	PARITY_ERR	Read only	When High, a parity error has occurred during a receive transaction. This bit is cleared by reading the Receive Data Register.
3	OVERFLOW	Read only	When High, a receive overflow occurs. This bit is cleared by reading the Receive Data Register.
4	FRAMING_ERR	Read only	When High, a framing error has occurred during a receive transaction. This bit is cleared by reading the Receive Data Register.
7:5	–	–	Unused

Note: *When RX_FIFO is enabled, PARITY_ERR is asserted when a parity error occurs, but deasserted before CoreUARTapb receives the next byte. It is the user's responsibility to monitor the PARITY_ERR signal (for example, treat it as an interrupt signal), as it is non-persistent when RX_FIFO = 1. Similarly, when RX_FIFO is enabled, FRAMING_ERR is asserted when a framing error occurs, but deasserted before CoreUARTapb receives the next byte. It, too, should be treated in the same manner as an interrupt signal*

A – List of Document Changes

The following table lists critical changes that were made in revisions of the document.

Revision	Changes	Page
Revision 5 (September 2012)	The "Core Version" was revised to v5.2. SmartFusion2 was added to the supported families (SAR 37393).	3
	The FIFO depth for SmartFusion2 devices was added to the second note for Table 1-1 • CoreUARTapb Utilization in FIFO Mode (SAR 37393).	6
	SmartFusion2 was added to Table 3-3 • Baud Value Fraction (SAR 37393).	15
Revision 4 (March 2012)	The "Core Version" was revised to v5.1.	3
	The "fractional part of a baud value" option was added to the "Fixed Mode Options" section. The "Fractional Part of Baud Value" section is new (SAR 37392).	14
	Figure 2-1 • SmartDesign CoreUARTapb Configuration Window was replaced (SAR 37392).	10
	BAUD_VAL_FRCTN and BAUD_VAL_FRCTN_EN were added to Table 3-2 • CoreUARTapb Configurable Options. Table 3-3 • Baud Value Fraction is new (SAR 37392).	14
	Control register 3 was added to Table 5-1 • CoreUARTapb Registers. The "Control Register 3" section is new (SAR 37392).	21
Revision 3 (October 2010)	The core version was revised to v4.2.	3
	SmartFusion, SX-A, and RTSX-S were added to the "Supported Families" section.	3
	Signal names have been changed to all upper case letters.	5 and others
	FRAMING_ERR as an output signal from the UART and APB I/F Wrapper was added to Figure 1-1 • Block Diagram of CoreUARTapb Normal Functionality and Figure 1-2 • Block Diagram of CoreUARTapb with FIFO Functionality.	5, 6
	SmartFusion was added to Table 1-1 • CoreUARTapb Utilization in FIFO Mode and Table 1-2 • CoreUARTapb Utilization in Normal Mode.	6, 7
	EQ 1-2 is new.	8
	The "Tool Flows" chapter was rewritten.	9
	The "Testbench Operation" chapter was deleted. The new "Tool Flows" chapter contains a "User Testbench" section.	11

Revision	Changes	Page
Revision 3 (continued)	Table 3-1 • CoreUARTapb Signals was revised.	13
	The RXRDY signal description was revised in to include the statement, "The data buffer must be read through APB via the Receive Data Register (0x04) to prevent an overflow condition from occurring."	
	The PARITY_ERR signal description was revised to state, "When RX FIFO is enabled, this bit is self clearing between bytes. Otherwise, this bit is synchronously cleared by performing a read operation on the Receive Data Register via the APB slave interface."	
	The FRAMING_ERR signal was added.	
	The OVERFLOW signal description was revised to state, "This bit is synchronously cleared by performing a read operation on the Receive Data Register via the APB slave interface."	
	The table note was revised from "Active low signals are designated with a trailing lower case n" to "All signals are active high unless otherwise indicated."	
	SmartFusion was added to Table 3-2 • CoreUARTapb Configurable Options.	14
	Table 5-1 • CoreUARTapb Registers was revised. The reset value for the Transmit Data register was changed from 0x01 to 0x00. The reset value for the Status register was changed from 0x00 to 0x01.	21
	The "Control Register 1" section was revised to state, in the example of a desired clock frequency of 10 MHz and baud rate of 9,600, that 0x40 (rather than 0x41) should be written to Control register 1. The second example was changed to read, "a baud rate of 381 is desired," rather than a baud rate of 762.	21
	The description for baud_value in Table 5-3 • Control Register 2 was changed from "Bits 12:7 of 13-bit baud value" to "Bits 12:8 of 13-bit baud value."	22
	The note in Table 5-6 • Status Register was revised to add, "Similarly, when RX_FIFO is enabled, FRAMING_ERR is asserted when a framing error occurs, but deasserted before CoreUARTapb receives the next byte. It, too, should be treated in the same manner as an interrupt signal."	23

B – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 650.318.8044

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Customer Support website (www.microsemi.com/soc/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

Website

You can browse a variety of technical and non-technical information on the SoC home page, at www.microsemi.com/soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Index

B
baud generator 5
BAUD_VAL_FRACTN 15
BAUD_VAL_FRCTN 22
block diagram 5
   FIFO mode 6
   normal mode 5
buffers
   receive 5
   transmit 5

C
configurable options 14
contacting Microsemi SoC Products Group
   customer service 27
   email 27
   web-based technical support 27
control inputs 5
Control register 3 22
core versions 3
customer service 27

D
description, general 3
device
   utilization and performance 6
double-buffering 5

F
FIFO mode 5
FIFOs
   receive 5
   transmit 5
fractional part of baud value 8

G
general description 3

I
I/O signals 13

M
Microsemi SoC Products Group
   email 27
   web-based technical support 27
   website 27

N
normal mode 5

P
parity
   errors 5
product support
   customer service 27
   email 27
   My Cases 28
   outside the U.S. 28
   technical support 27
   website 27

S
signals, I/O 13
state machines
   receive 5
   transmit 5
supported FPGA families 3

T
tech support
   ITAR 28
   My Cases 28
   outside the U.S. 28
technical support 27
timing diagrams
   overflow error 18
   parity error 18
   serial receive 17
   serial transmit 17

V
versions, core 3

W
web-based technical support 27

50200101-5/9.12