HB0101
Handbook
CoreUARTapb v5.7

Microsemi

a Microchip company

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

50200101. 14.0 9/20

Contents

Figures

Tables

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 14.0

Added PolarFire® SoC support.

1.2 Revision 13.0

The following is a summary of the changes made in this revision.

- Updated core version from 5.5 to 5.6.
- Added PolarFire device values in Table 1 and Table 2.

1.3 Revision 12.0

The following is a summary of the changes made in this revision.

- Updated core version from 5.4 to 5.5.
- Added RTG4 device values in Table 1 and Table 2.

1.4 Revision 11.0

The following is a summary of the changes made in this revision.

- Formatted the document as per the new HB specifications.
- Added a note in Table 1.
- Updated Baud Rate, page 9.
- Updated Table 10.

1.5 Revision 10.0

The following is a summary of the changes made in this revision.

- Added RTG4 information to Supported Families, page 5, FAMILY parameter and to all Device Utilization tables.
- Changed core version from 5.3 to 5.4 in numerous places.

1.6 Revision 9.0

Updated Note in Framing Error (no legacy mode), page 18 and Note in Framing Error (legacy mode), page 19 section (SAR 56623).

1.7 Revision 8.0

Added Note (SAR 56293).

1.8 Revision 7.0

Figure 6, Figure 7, and Figure 8 were updated (SAR 55499).

1.9 Revision 6.0

The following is a summary of the changes made in this revision.

- The “Core Version” section was revised to v5.3.
- IGLOO2 was added to “Supported Families” section.
- Table 1 was updated.
- Table 2 was updated.
- IGLOO2 was added to Table 10.

- Figure 7 was updated (SAR 38130).

1.10 Revision 5.0

The following is a summary of the changes made in this revision.

- The "Core Version" was revised to v5.2. SmartFusion2 was added to the supported families (SAR 37393).
- The FIFO depth for SmartFusion2 devices was added to the second note for Table 1 CoreUARTapb Utilization in FIFO Mode (SAR 37393).
- SmartFusion2 was added to Table 11 Baud Value Fraction (SAR 37393).

1.11 Revision 4.0

The following is a summary of the changes made in this revision.

- The "Core Version" was revised to v5.1.
- 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).
- Figure 9 SmartDesign CoreUARTapb Configuration Window was replaced (SAR 37392).
- BAUD_VAL_FRCTN and BAUD_VAL_FRCTN_EN were added to Table 10 CoreUARTapb Configurable Options. Table 11 Baud Value Fraction is new (SAR 37392).
- Control register 3 was added to Table 3 CoreUARTapb Registers. The "Control Register 3" section is new (SAR 37392).

1.12 Revision 3.0

The following is a summary of the changes made in this revision.

- The core version was revised to v4.2.
- SmartFusion, SX-A, and RTSX-S were added to the "Supported Families" section.
- Signal names have been changed to all upper case letters.
- FRAMING_ERR as an output signal from the UART and APB I/F Wrapper was added to Figure 1 • Block Diagram of CoreUARTapb Normal Functionality and Figure 2 • Block Diagram of CoreUARTapb with FIFO Functionality.
- SmartFusion was added to Table 1 • CoreUARTapb Utilization in FIFO Mode and Table 2 • CoreUARTapb Utilization in Normal Mode.
- EQ 2 is new.
- The "Tool Flows" chapter was rewritten.
- The "Testbench Operation" chapter was deleted. The new "Tool Flows" chapter contains a "User Testbench" section.
- Table 9 o CoreUARTapb Signals was revised.
- 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 10 • CoreUARTapb Configurable Options.
- Table 3 • 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.
- 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.

- The description for baud_value in Table 5 • Control Register 2 was changed from “Bits 12:7 of 13-bit baud value” to “Bits 12:8 of 13-bit baud value.”
- The note in Table 8 • 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.”

1.13 Revision 1.0

The first publication of this document.

2 Preface

2.1 About this Document

This handbook provides details about the CoreUART APB IP core and how to use it.

2.2 Intended Audience

Designers using Libero® System-on-Chip (SoC) or Libero Integrated Design Environment (IDE).

2.3 References

Microsemi Publications.

3 Introduction

3.1 Overview

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.

3.2 Key Features

CoreUARTapb is a highly configurable core and has the following features:

- Asynchronous mode to interface with industry standard UART
- Optional transmit and receive FIFOs
- Advanced peripheral bus (APB) interface
- Fixed and programmable modes of operation

3.3 Core Version

This handbook applies to CoreUARTapb v5.7. The release notes provided with the core list known discrepancies between this handbook and the core release.

3.4 Supported Families

- PolarFire® SoC
- PolarFire®
- IGLOO®2
- RTG4™
- SmartFusion®2
- SmartFusion®
- IGLOO®
- IGLOOe
- IGLOO PLUS
- ProASIC®3
- ProASIC3E
- ProASIC3L
- Fusion
- ProASICPLUS®
- Axcelerator®
- RTAX-S
- SX-A
- RTSX-S

3.5 Device Utilization and Performance

Utilization statistics for targeted devices are listed in Table 1 through Table 2.

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

	Cells or Tiles				Utilization		
Family	**Sequential**	**Combinatorial**	**Total**	**Memory Blocks**	**Device**	**Total**	**Performance MHz**
PolarFire	268	273	541	2	PA5M500	0.11	332
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	241	280	521	2	M2S150T	0.35%	250
Fusion	140	240	380	2	AFS600	3%	125
ProASIC^{PLUS}	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
IGLOO2	241	280	521	2	M2GL150T	0.35%	250
RTG4	262	356	518	2	RT4G150	0.4%	156

Note: FPGA resources and performance data for the PolarFire SoC family is similar to PolarFire family.

Note:

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/IGLOO2 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.
5. CoreUART does not support baud value of zero.

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

	Cells or Tiles				Utilization		
Family	**Sequential**	**Combinatorial**	**Total**	**Memory Blocks**	**Device**	**Total**	**Performance MHz**
PolarFire	114	162	276	2	PA5M500	0.05	304
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	111	140	251	0	M2S150T	0.2%	250
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
IGLOO2	111	140	251	0	M2GL150T	0.2%	250
RTG4	114	180	294	0	RT4G150	0.2%	151

Note: FPGA resources and performance data for the PolarFire SoC family is similar to PolarFire family.

Note:

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

4 Functional Block Description

Figure 1 shows the block diagram of the CoreUARTapb normal mode functionality. Figure 2 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 • **Block Diagram of CoreUARTapb Normal Functionality**

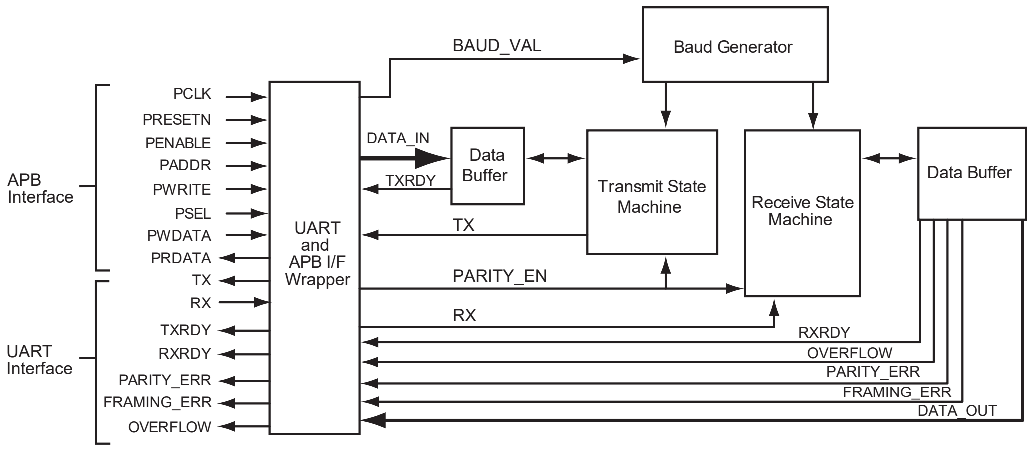
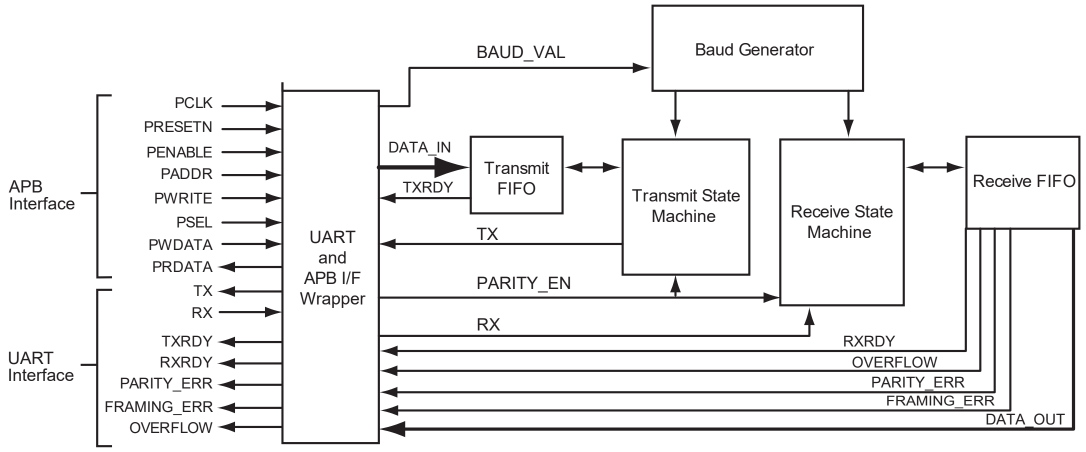


Figure 2 • **Block Diagram of CoreUARTapb with FIFO Functionality**



4.1 Fixed Mode Options

There are four options in Fixed mode CoreUARTapb operation:

- Character size
- Parity
- Baud rate
- Fractional part of baud value

These values are hardwired and cannot be changed during runtime.

4.1.1 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.

4.1.2 Parity

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

4.1.3 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.

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

EQ 1

where

clk = the frequency of the system clock in hertz

baud rate = the desired baud rate

and

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

EQ 2

The term baudval must be rounded to the nearest integer and must be greater than or equal to 1 or less than or equal to 8191. For example, a system with a 33 MHz system clock and a desired baud rate of 9,600 must have a baud_value of 214 decimal or D6 hex. So, to get the desired baud rate, you must assign 0xD6 to the baud_value (by writing appropriate values to Control Register 1 and Control Register

2). More accurate baud rates can be achieved when fractional part of baud value is enabled and desired precision is selected.

4.1.4 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 11 on page 17).

5 Operation

5.1 CoreUARTapb Configuration Registers

5.1.1 CoreUARTapb Programmer's Model

Table 3 lists the registers for CoreUARTapb.

Table 3 • **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

5.1.2 Transmit Data Register

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

5.1.3 Receive Data Register

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

5.1.4 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 3:

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

EQ 3

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 3. 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 4 • **Control Register 1**

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

5.1.5 Control Register 2

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

Table 5 • **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

5.1.6 Control Register 3

Table 6 shows Control Register 3, which is used to assign values to the configuration inputs available on CoreUARTapb.

Table 6 • **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 7.

Table 7 • **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

5.1.7 Status Register

Table 8 shows the Status Register, which provides information on the status of CoreUARTapb.

Table 8 • **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.

6 Interface Description

Signal descriptions for CoreUARTapb are defined in Table 9. The APB interface allows access to the CoreUARTapb internal registers, FIFO, and internal memory. This interface is synchronous to the clock.

Table 9 • **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
PREADY	Output	Ready. The Slave uses this signal to extend an APB transfer.
PSLVERR	Output	This signal indicates a transfer failure.

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

6.1 Core Parameters

6.1.1 CoreUARTapb Configurable Options

There are a number of configurable options that apply to CoreUARTapb, as shown in Table 10. 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 10 • **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- ProASICPLUS 15- ProASIC3 16- ProASIC3E 17- Fusion 18- SmartFusion 19- SmartFusion2 20- IGLOO 21- IGLOOe 22- ProASIC3L 23- IGLOO PLUS 24- IGLOO2 25- RTG4 26- PolarFire 27- PolarFire SoC
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. **Note:** BAUD_VALUE = 0 is not supported.
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.
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.

Table 10 • **CoreUARTapb Configurable Options *(continued)***

Configurable Options	Default Setting	Description
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 11 shows the fraction that baud value will be modified by when in Fixed Mode and BAUD_VAL_FRCTN is used.

Table 11 • **Baud Value Fraction**

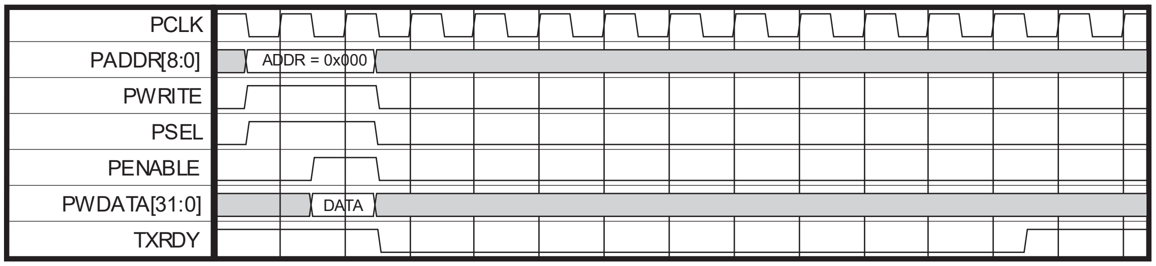
BAUD_VAL_FRCTN	Precision
0	+0.0
1	+0.125
2	+0.25
3	+0.375
4	+0.5
5	+0.625
6	+0.75
7	+0.875

7 Timing Diagrams

The UART waveforms can be broken down into a few basic functions: transmit data, receive data, and errors. Figure 3 shows serial transmit signals, and Figure 4 shows serial receive signals. Figure 5 and Figure 6 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.

7.1 Serial Transmit

Figure 3 • **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.
- It is recommended that after a reset, the data is not written into the transmitter channel register until 11 baud clock cycles. However if the reset is held for 11 baud clock cycles or more, the data can be written into the transmitter channel register straight after the deassertion of the reset.

7.2 Serial Receive

Figure 4 • **Serial Receive**

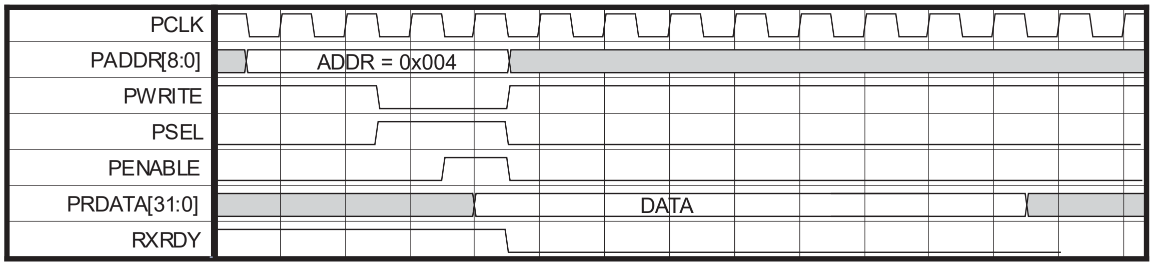
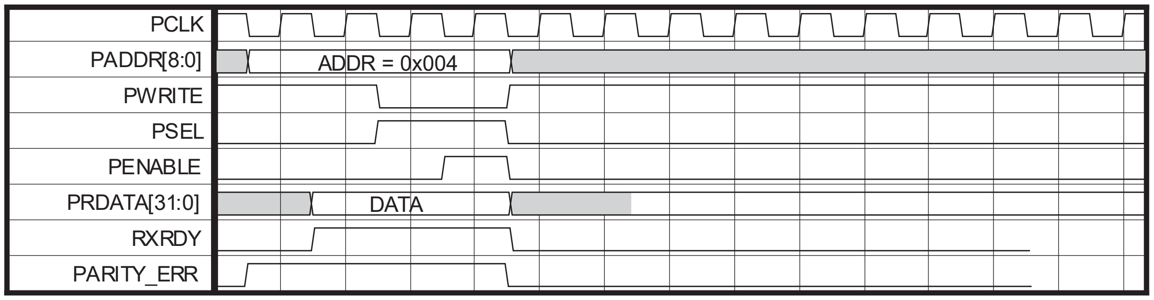


Figure 5 • **Parity Error**

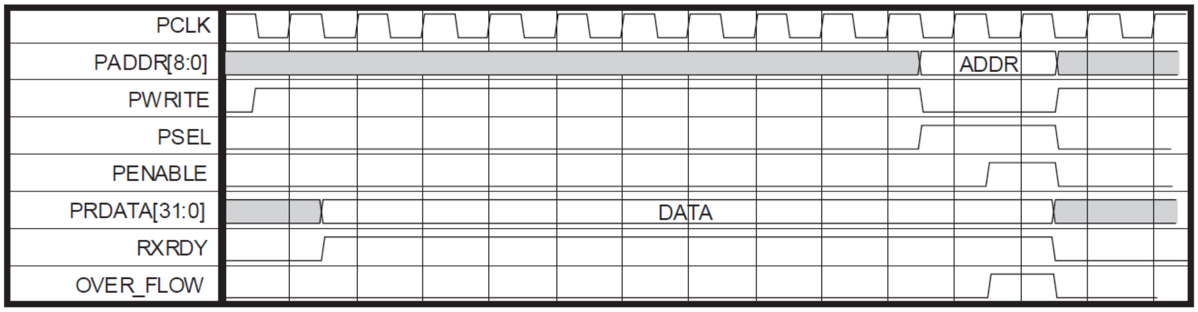


Note:

1. 1.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.
2. 2.When RX_FIFO=1, the data comes out in first in first out (FIFO). However, if a parity error occurs, the data which caused the parity error goes to the output (if the read request enabled) until the parity error is de-asserted. This gives a chance to read the data that caused the parity error when it occurs. This data will not be stored in the RX_FIFO because it is invalid. This timing diagram shows the data that caused the parity error being read. During a parity error, no data will be read from the RX_FIFO so no valid data is lost.

7.3 Overflow Error

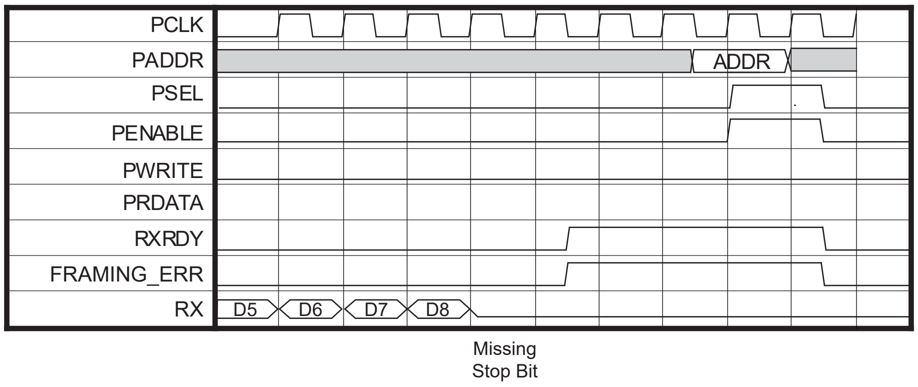
Figure 6 • **Overflow Error**



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

7.4 Framing Error (no legacy mode)

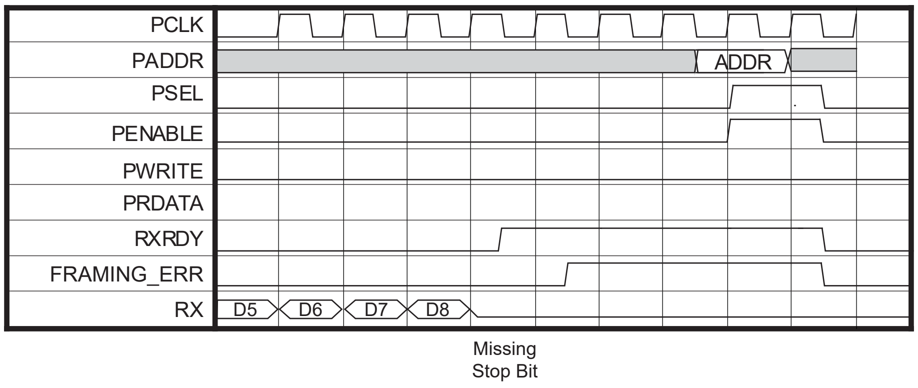
Figure 7 • **Framing Error (no legacy mode)**



Note: In Normal (non-legacy) mode, RXRDY and FRAMING_ERR are one system clock cycle apart. The FRAMING_ERR signal gets asserted before one system clock cycle of the RXRDY signal assertion. The error is cleared using a read operation.

7.5 Framing Error (legacy mode)

Figure 8 • **Framing Error (legacy mode)**



Note: In Legacy mode, the FRAMING_ERR signal gets asserted after one frame bit period of the RXRDY signal assertion. When the data is available, the RXRDY signal gets asserted and then the check for missing stop bit occurs. The error is cleared using a read operation.

8 Tool Flows

8.1 License

No license required.

8.1.1 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 SoC. 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.

8.1.2 RTL

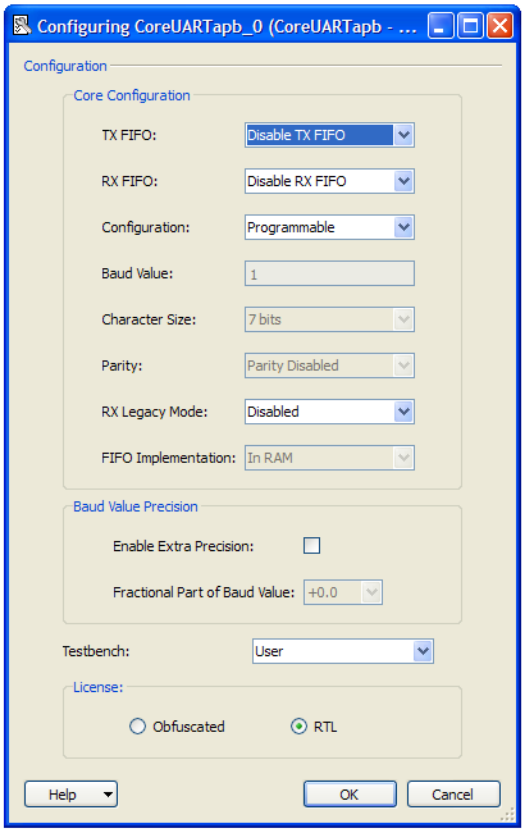
Complete RTL source code is provided for the core and testbenches.

8.2 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 9.

To know how to create SmartDesign project using the IP cores, refer to *Libero SoC documents page* and use the latest SmartDesign user guide.

Figure 9 • **SmartDesign CoreUARTapb Configuration Window**



Configuring CoreUARTapb_0 (CoreUARTapb - ...

Configuration

Core Configuration

TX FIFO: Disable TX FIFO

RX FIFO: Disable RX FIFO

Configuration: Programmable

Baud Value: 1

Character Size: 7 bits

Parity: Parity Disabled

RX Legacy Mode: Disabled

FIFO Implementation: In RAM

Baud Value Precision

Enable Extra Precision: ☐

Fractional Part of Baud Value: +0.0

Testbench: User

License:

Obfuscated [✓ RTL]

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.

8.3 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 project, it will install the user testbench files.

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

8.4 Synthesis in Libero

Click the **Synthesis** icon in Libero. 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.

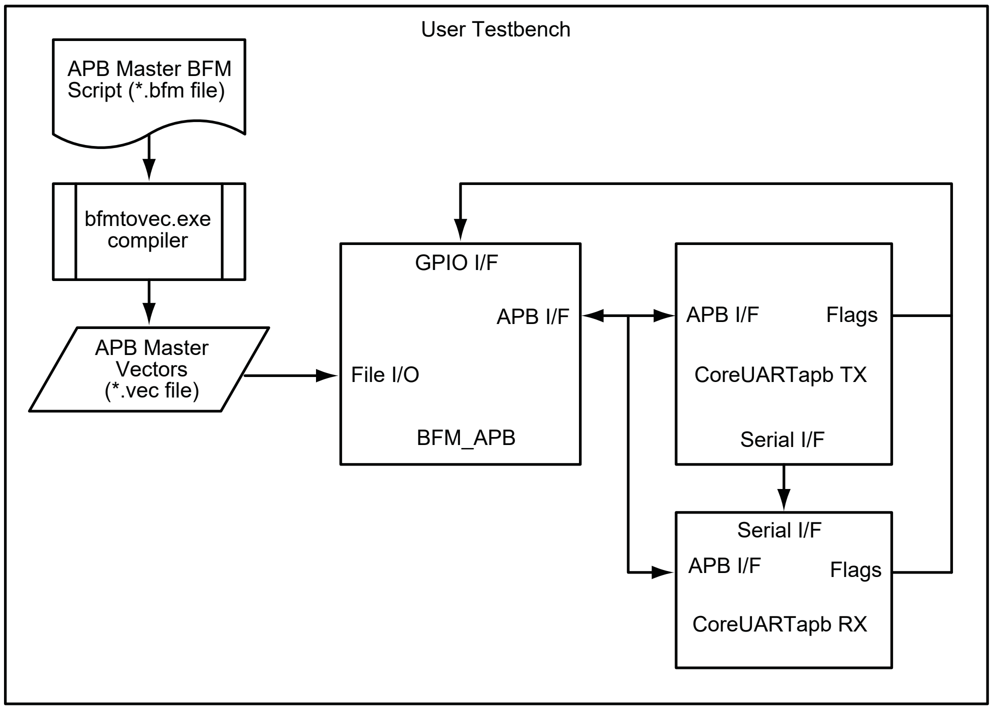
8.5 Place-and-Route in Libero

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

9 Testbench Operation and Modification

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 10 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 10.

Figure 10 • **CoreUARTapb User Testbench**



As shown in Figure 10, 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 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 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. For more information, refer to the *DirectCore AMBA BFM User's Guide*.