



DW APB ARC EM Starter Kit UART User Manual

Version 6284-001 March 2013

Copyright Notice and Proprietary Information Notice

Copyright © 2013 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <http://www.synopsys.com/Company/Pages/Trademarks.aspx>.

All other product or company names may be trademarks of their respective owners.

Synopsys, Inc.
700 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Contents

Customer Support.....	4
Accessing SolvNet.....	4
Contacting the Synopsys Technical Support Center.....	4
1 Introduction	5
Definitions	5
2 DW APB UART Driver.....	6
DW_apb_uart Features	6
Software Architecture.....	7
Error Codes.....	8
Functions	9
msf_io_fifo_buffer_read_begin()	9
_dw_apb_uart_polled_init().....	10
_dw_apb_uart_polled_deinit().....	10
_dw_apb_uart_polled_getc()	11
_dw_apb_uart_polled_putc()	11
_dw_apb_uart_polled_status().....	12
_dw_apb_uart_ioctl()	12
_dw_apb_uart_int_install().....	13
_dw_apb_uart_int_init()	13
_dw_apb_uart_int_enable()	14
_dw_apb_uart_int_deinit()	14
_dw_apb_uart_int_putc()	14
_dw_apb_uart_int_isr()	15
dw_apb_uart_reconfigure().....	15
Data Types.....	16
bsp_dw_apb_uart_init.....	16
I/O Controllers and I/O Structures.....	17
IO_IOCTL_SERIAL_GET_STOP_BITS/ IO_IOCTL_SERIAL_SET_STOP_BITS	17
IO_IOCTL_SERIAL_GET_DATA_BITS/ IO_IOCTL_SERIAL_SET_DATA_BITS	18
IO_IOCTL_SERIAL_GET_BAUD/ IO_IOCTL_SERIAL_SET_BAUD.....	18
IO_IOCTL_SERIAL_GET_STATS/ IO_IOCTL_SERIAL_CLEAR_STATS	18

IO_IOCTL_SERIAL_GET_CONFIG..... 19

IO_IOCTL_SERIAL_GET_HW_SIGNAL/ IO_IOCTL_SERIAL_SET_HW_SIGNAL/
IO_IOCTL_SERIAL_CLEAR_HW_SIGNAL.....20

IO_IOCTL_SERIAL_GET_PARITY/ IO_IOCTL_SERIAL_SET_PARITY.....20

Default Configuration for DW APB UART 21

3 Test Application.....22

Customer support is available through SolvNet online customer support and through contacting the Synopsys Technical Support Center.

Accessing SolvNet

SolvNet includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNet also gives you access to a wide range of Synopsys online services, which include downloading software, viewing Documentation on the Web, and entering a call to the Support Center.

To access SolvNet:

1. Go to the SolvNet Web page at <http://solvnet.synopsys.com/>.
2. If prompted, enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.)

If you need help using SolvNet, click SolvNet Help in the Support Resources section.

Contacting the Synopsys Technical Support Center

If you have problems, questions, or suggestions, you can contact the Synopsys Technical Support Center in the following ways:

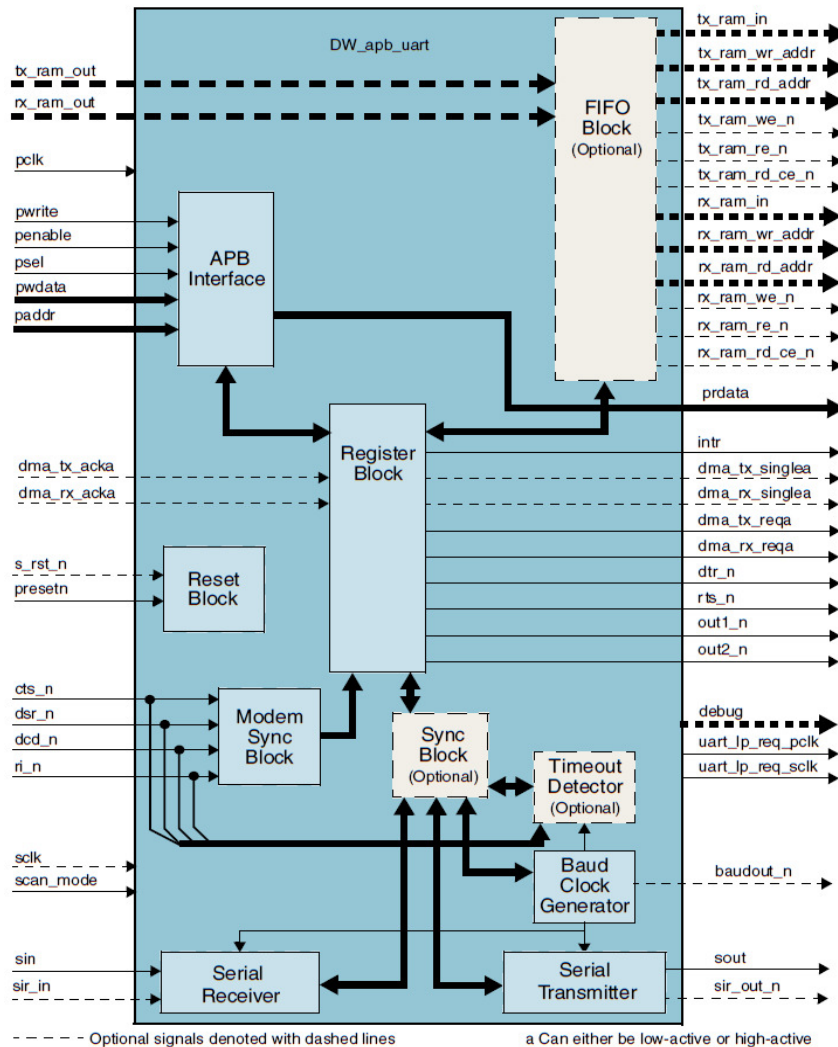
- Open a call to your local support center from the Web by going to <http://solvnet.synopsys.com/> (Synopsys user name and password required), then clicking “Enter a Call to the Support Center.”
- Send an e-mail message to your local support center.
 - E-mail support_center@synopsys.com from within North America.
 - Find other local support center e-mail addresses at <http://www.synopsys.com/Support/GlobalSupportCenters>.
- Telephone your local support center.
 - Call (800) 245-8005 from within the continental United States.
 - Call (650) 584-4200 from Canada.
 - Find other local support center telephone numbers at <http://www.synopsys.com/Support/GlobalSupportCenters>.

This document describes the usage of the UART driver for writing applications. It aims to make the development of embedded application easier for developers. This document contains the descriptions of all the data types and API functions.

Definitions

Term	Definition
ARC 700	Family of 32 bit ARC processor cores with MMU support
AOM	Automated Overlay Manager
HLA	High-Level Architecture
MAS	Micro-Architecture Specification
ARChitect™	System Configuration tool developed by ARC. ARChitect can be run from the command line, or as a GUI to provide point-and-click definition of a processor architecture, and generates Verilog source code of the configured design.
HDL	Hardware Description Language — a programming language used to describe the hardware implementation of a system. HDL is usually synthesized to produce a chip (ASIC). Two common HDLs are Verilog and VHDL.
MetaWare	C/C++ development toolkit for ARC processors
ML509	Xilinx FPGA development platform used as a reference platform by ARC
MQX	RTOS supplied by ARC

This chapter describes the UART device drivers. The following is the functional block diagram of the DW APB UART.



DW_apb_UART Features

- The DW_apb_uart is a programmable Universal Asynchronous Receiver/Transmitter (UART). This component is an AMBA 2.0-compliant Advanced Peripheral Bus (APB) slave device.

- The DW_apb_uart is modeled after the industry-standard 16550. However, the register address space is relocated to 32-bit data boundaries for APB bus implementation.
- The DW_apb_uart is used for serial communication with the following:
 - Peripherals
 - Modems (data carrier equipment, DCE)
 - Data sets

Data is written from a master (CPU) over the APB bus to the UART. It is then converted to serial and transmitted to the destination device. Serial data is also received by the UART and stored for master (CPU) to read back.

The DW_apb_uart contains registers that control the following parameters:

- Character length
- Baud rate
- Parity generation/checking
- Interrupt generation

Although the DW_apb_uart has only one interrupt output signal (intr), several prioritized interrupt types can assert it. Each of the interrupt types can be separately enabled or disabled by the control registers.

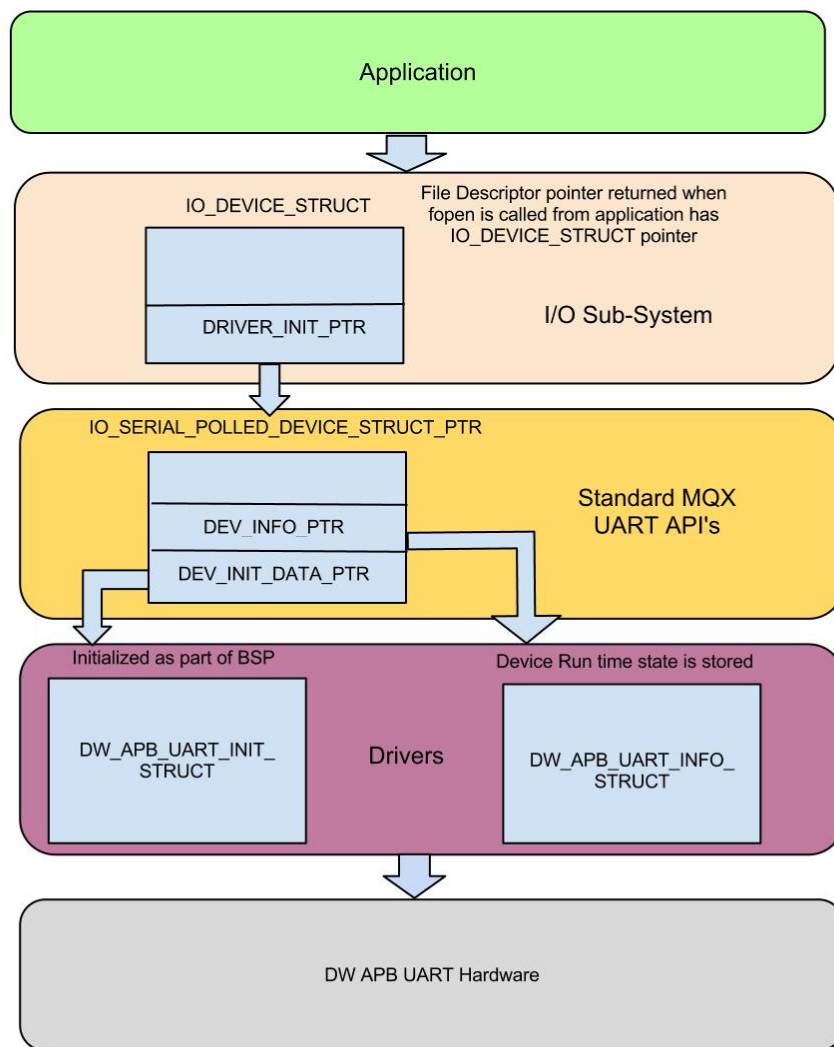
Software Architecture

The DW APB driver is written using the standard MQX UART API based on the I/O subsystem. Standard MQX UART APIs are responsible for data flow between the I/O subsystem and low-level driver.

All low-level functionality such as data transmission, reception, and interrupts is handled by the low-level driver. Standard MQX UART APIs are responsible for maintaining the data provided by the low-level driver.

For more information on the driver structures and members, see the *MQX® RTOS I/O Drivers User's Guide*.

The following diagram provides an overview of the software architecture.



Error Codes

The DW APB UART driver returns standard MQX defined ERROR codes for each error condition handled by the driver.

The following are the standard MQX error defines used in the driver:

Error Code	Description
MQX_INVALID_PARAMETER	Invalid baud rate passed as argument to driver
MQX_OUT_OF_MEMORY	Memory allocation failed

MQX_ETIMEDOUT	Timed out while waiting for data to be received
IO_ERROR_INVALID_IOCTL_CMD	Invalid IOCTL

Functions

msf_io_fifo_buffer_read_begin()

Description This function installs the polling mode of the driver. It can be called either from the BSP or from the application.

Syntax

```
uint_32 _dw_apb_uart_polled_install(char_ptr identifier,  
pointer init_data_ptr, _mqx_uint queue_size);
```

Parameters `identifier`
This variable is a `char_ptr` input parameter to the `_dw_apb_uart_polled_install()` function and it holds string that identifies the device during `fopen()`.

`init_data_ptr`
This variable is a `void` input parameter to the `_dw_apb_uart_polled_install()` function. The default configuration of the device is provided through this parameter.

`queue_size`
This variable is an integer. input parameter to the `_dw_apb_uart_polled_install()` function. It holds the size of the queue used by the standard MQX UART API's.

`_dw_apb_uart_polled_init()`

Description This function initializes the DW APB UART hardware and provides a handle to the user.

This function also initializes required resources for the device. It must be called prior to the use of any other functions.

Syntax

```
_mqx_uint _dw_apb_uart_polled_init(DW_APB_UART_INIT_STRUCT_PTR
io_init_ptr, pointer _PTR_ io_info_ptr_ptr, char_ptr
open_name_ptr);
```

Parameters `io_init_ptr`
This variable is a structure pointer of the `DW_APB_UART_INIT_STRUCT_PTR` type. The default configuration is provided by this parameter.

`io_info_ptr_ptr`

This variable is `void double pointer`, and its output argument to `_dw_apb_uart_polled_init()`. It holds the address to store device-specific information.

`open_name_ptr`

This variable is a `char_ptr` input parameter to the `_dw_apb_uart_polled_init()` function, and it holds a string that identifies the device during `fopen()`.

`_dw_apb_uart_polled_deinit()`

Description This function de-initializes the DW APB UART hardware, terminates modem control signals, and releases all obtained resources. After calling this function, users have no control over the UART.

Syntax

```
_mqx_uint
_dw_apb_uart_polled_deinit(DW_APB_UART_INIT_STRUCT_PTR
io_init_ptr, DW_APB_UART_INFO_STRUCT_PTR io_info_ptr);
```

Parameters `io_init_ptr`
This variable is a structure pointer of the `DW_APB_UART_INIT_STRUCT_PTR` type. The default configuration is provided by this parameter.

`io_info_ptr`

This variable is a structure pointer of the `DW_APB_UART_INFO_STRUCT_PTR` type,

which is a private data structure. It is an input argument to `_dw_apb_uart_polled_deinit`

`_dw_apb_uart_polled_getc()`

Description This function returns a character when the character is available. This function polls the device for input. If the device does not receive any character by the specified timeout period, `_mqx_fatal_error` with `MQX_ETIMEDOUT` is called.

Syntax

```
char _dw_apb_uart_polled_getc( DW_APB_UART_INFO_STRUCT_PTR  
io_info_ptr);
```

Parameters `io_info_ptr`
This variable is a structure pointer of the `DW_APB_UART_INFO_STRUCT_PTR` type, which is a private data structure. It is an input argument to `_dw_apb_uart_polled_deinit()`.

`_dw_apb_uart_polled_putc()`

Description This function writes the provided character to the transmit FIFO.

Syntax

```
void _dw_apb_uart_polled_putc( DW_APB_UART_INFO_STRUCT_PTR  
io_info_ptr, char data);
```

Parameters `io_info_ptr`
This variable is a structure pointer of the `DW_APB_UART_INFO_STRUCT_PTR` type, which is a private data structure. It is an input argument to `_dw_apb_uart_polled_putc()`.

`data`

Data to be written to the TX FIFO. Input argument.

`_dw_apb_uart_polled_status()`

Description This function returns `TRUE` if a character is available on the on I/O device, otherwise it returns `FALSE`.

Syntax

```
boolean _dw_apb_uart_polled_status (DW_APB_UART_INFO_STRUCT_PTR
io_info_ptr);
```

Parameters `io_info_ptr`
This variable is a structure pointer of the `DW_APB_UART_INFO_STRUCT_PTR` type, which is a private data structure. It is an input argument to `_dw_apb_uart_polled_status()`.

`_dw_apb_uart_ioctl()`

Description This function performs miscellaneous services for the I/O device. The following are the actions performed.

- Set a particular hardware configuration
- Get the current configuration of the hardware
- Adjust baud rate
- Get the statistical information of the device

Syntax

```
uint_32 _dw_apb_uart_ioctl(DW_APB_UART_INFO_STRUCT_PTR
io_info_ptr, _mqx_uint cmd, pointer param_ptr);
```

Parameters `io_info_ptr`
This variable is a structure pointer of the `DW_APB_UART_INFO_STRUCT_PTR` type, which is private data structure. It is an input argument to `_dw_apb_uart_polled_ioctl()`.

`cmd`

Any define that is described in the I/O controls and structures can be used. The function behavior and a type of `param_ptr` is determined according to this parameter.

`param_ptr`

This parameter is taken as a `void` pointer and type-casted to one of the control-data structures depending on the command type. The type of `param_ptr` is determined according to the command value.

`_dw_apb_uart_int_install()`

Description This function installs Interrupt mode of driver. It can be called either from the BSP or from the application.

Syntax

```
uint_32 _dw_apb_uart_int_install(char_ptr identifier,  
pointer init_data_ptr, _mqx_uint queue_size);
```

Parameters `identifier`
This variable is a `char_ptr`, It is an input parameter to the `_dw_apb_uart_int_install()` function and it holds string that identifies the device during `fopen()`.

`init_data_ptr`
This variable is a `void` input parameter to the `_dw_apb_uart_int_install()` function. Default configuration of the device is provided through this parameter.

`queue_size`
This variable is an integer, input parameter to the `_dw_apb_uart_int_install()` function. Its holds the size of the queue used by the standard MQX UART APIs.

`_dw_apb_uart_int_init()`

Description This function initializes the DW APB UART hardware for interrupt mode and provides a handle to the user. This function also initializes required resources for the device. It must be called prior to the use of any other functions.

Syntax

```
uint_32 _dw_apb_uart_int_init(IO_SERIAL_INT_DEVICE_STRUCT_PTR  
int_io_dev_ptr, char _PTR_ open_name_ptr);
```

Parameters `int_io_dev_ptr`
This variable is pointer of type `IO_SERIAL_INT_DEVICE_STRUCT_PTR`. It is an input argument to `_dw_apb_uart_int_init()`. It holds the interrupt I/O initialization information.

`open_name_ptr`
This variable is a `char_ptr` an input parameter to the `_dw_apb_uart_int_init()` function and holds string that identifies the device during `fopen()`.

`_dw_apb_uart_int_enable()`

Description This function enables the UART interrupts.

Syntax

```
uint_32 _dw_apb_uart_int_enable (DW_APB_UART_INFO_STRUCT_PTR  
io_info_ptr);
```

Parameters `io_info_ptr`
This variable is a structure pointer of the `DW_APB_UART_INFO_STRUCT_PTR` type, which is a private data structure. It is an input argument to `_dw_apb_uart_int_enable()`.

`_dw_apb_uart_int_deinit()`

Description This function de-initializes the DW APB UART hardware, terminates modem control signals, and releases all obtained resources. After calling this function, users have no control over the UART.

Syntax

```
uint_32 _dw_apb_uart_int_deinit (DW_APB_UART_INIT_STRUCT_PTR  
io_init_ptr, DW_APB_UART_INFO_STRUCT_PTR  
io_info_ptr);
```

Parameters `io_init_ptr`
This variable is a structure pointer of the `DW_APB_UART_INIT_STRUCT_PTR` type. The default configuration is provided by this parameter.

`io_info_ptr`

This variable is a structure pointer of the `DW_APB_UART_INFO_STRUCT_PTR` type, which is a private data structure. It is an input argument to `_dw_apb_uart_int_deinit()`.

`_dw_apb_uart_int_putc()`

Description This function is called to write out the first character, when the output serial device and output ring buffers are empty.

Syntax

```
void _dw_apb_uart_int_putc( IO_SERIAL_INT_DEVICE_STRUCT_PTR  
int_io_dev_ptr, char data);
```

Parameters `info_io_dev_ptr`

This variable is pointer of the `IO_SERIAL_INT_DEVICE_STRUCT_PTR` type. It is an input argument to `_dw_apb_uart_int_init()`. It holds the interrupt I/O initialization information.

`data`

Data to be written to the TX FIFO. Input argument.

`_dw_apb_uart_int_isr()`

Description Interrupt handler for the serial I/O interrupts.

Syntax `void _dw_apb_uart_int_isr(pointer parameter);`

Parameters `parameter`

This argument is a void pointer input argument to interrupt handler. It holds the address of the interrupt I/O initialization information (`IO_SERIAL_INT_DEVICE_STRUCT_PTR`).

`dw_apb_uart_reconfigure()`

Description This function re-initializes the UART with the values provided in `DW_APB_UART_INIT_STRUCT`. This function is also used to set a particular hardware configuration using IOCTLs.

Syntax `uint_32 dw_apb_uart_reconfigure(DW_APB_UART_INFO_STRUCT_PTR io_info_ptr)`

Parameters `io_info_ptr`

This variable is a structure pointer of the `DW_APB_UART_INFO_STRUCT_PTR` type, which is a private data structure of the driver. It is an input argument to `dw_apb_uart_reconfigure()`.

`data`

Data to be written in TX FIFO. Input argument.

Data Types

bsp_dw_apb_uart_init

```
typedef struct dw_apb_uart_init_struct
{
    pointer    COM_PORT_ADDRESS;
    _mqx_uint  QUEUE_SIZE;
    uint_32    BAUD_RATE;
    _mqx_uint  BITS_PER_CHARACTER;
    _mqx_uint  PARITY;
    _mqx_uint  STOP_BITS;
    _mqx_uint  VECTOR;
}DW_APB_UART_INIT_STRUCT, _PTR_ DW_APB_UART_INIT_STRUCT_PTR;
```

Field Name	Description
COM_PORT_ADDRESS	Base address of the UART memory where the device is mapped. This value is used by the driver to access the registers.
QUEUE_SIZE	Size of the queue used by the driver to buffer incoming and outgoing data.
BAUD_RATE	Baud rate at which the UART is to be initialized. Most UARTs are initialized to the values 9600 and 115200.
BITS_PER_CHARACTER	Number of bits in a character. Possible values are 5, 6, 7, and 8.
PARITY	Parity bits at which the UART is to be initialized. Possible parity values are none, even, and odd.
STOP_BITS	Number of stop bits at which device is to be initialized. Possible values are 1, 1.5, and 2.
VECTOR	Interrupt vector number for this UART.

I/O Controllers and I/O Structures

IOCTLS are used to interact with the device, to check the status, and to change any settings if required.

The following IOCTLS are supported by the driver.

- IO_IOCTL_SERIAL_GET_STOP_BITS
- IO_IOCTL_SERIAL_SET_STOP_BITS
- IO_IOCTL_SERIAL_GET_DATA_BITS
- IO_IOCTL_SERIAL_SET_DATA_BITS
- IO_IOCTL_SERIAL_GET_BAUD
- IO_IOCTL_SERIAL_SET_BAUD
- IO_IOCTL_SERIAL_GET_STATS
- IO_IOCTL_SERIAL_CLEAR_STATS
- IO_IOCTL_SERIAL_GET_CONFIG
- IO_IOCTL_SERIAL_GET_HW_SIGNAL
- IO_IOCTL_SERIAL_SET_HW_SIGNAL
- IO_IOCTL_SERIAL_CLEAR_HW_SIGNAL
- IO_IOCTL_SERIAL_GET_PARITY
- IO_IOCTL_SERIAL_SET_PARITY

IO_IOCTL_SERIAL_GET_STOP_BITS/ IO_IOCTL_SERIAL_SET_STOP_BITS

This IOCTL is used to get and set the current stop bits of the UART.

```
uint_32_ptr  ret_buf
```

Field Name	Description
ret_buf	This field can have the following values to set the stop bits <pre>#define IO_SERIAL_STOP_BITS_1 (1) #define IO_SERIAL_STOP_BITS_1_5 (2) #define IO_SERIAL_STOP_BITS_2 (3)</pre>

IO_IOCTL_SERIAL_GET_DATA_BITS/ IO_IOCTL_SERIAL_SET_DATA_BITS

This IOCTL is used to get and set the current data bits of the UART.

```
uint_32_ptr  ret_buf
```

Field Name	Description
ret_buf	This field can have the following values to set the stop bits: <pre>#define DW_ABP_UART_DATA_BITS_5 (1) #define DW_ABP_UART_DATA_BITS_6 (2) #define DW_ABP_UART_DATA_BITS_7 (3)</pre>

IO_IOCTL_SERIAL_GET_BAUD/ IO_IOCTL_SERIAL_SET_BAUD

This IOCTL is used to get and set the current baud rate of the UART.

```
uint_32_ptr  ret_buf
```

Field Name	Description
ret_buf	This field is used to pass the baud rate as an argument; any valid baud rate can be passed. Standard baud rates are 9600 and 115200.

IO_IOCTL_SERIAL_GET_STATS/ IO_IOCTL_SERIAL_CLEAR_STATS

This IOCTL is used to get and clear the current statistical information of the UART. Statistical information is maintained in a private data structure of the UART driver.

The following are the members of the `DW_ABP_UART_INFO_STRUCT` structure used to maintain the statistical information.

```
typedef struct dw_apb_uart_info_struct
{
    .
    .
    uint_32      INTERRUPTS;
    uint_32      RX_DROPPED_INPUT;
    uint_32      RX_GOOD_CHARS;
    uint_32      RX_CHARS;
    uint_32      TX_CHARS;
    uint_32      RX_PARITY_ERRORS;
    uint_32      RX_FRAMING_ERRORS;
```

```

uint_32          RX_OVERRUNS;
uint_32          RX_BREAKS;
} DW_APB_UART_INFO_STRUCT, _PTR_ DW_APB_UART_INFO_STRUCT_PTR;

```

Field Name	Description
INTERRUPTS	Number of Interrupts occurred
RX_GOOD_CHARS	Total number of received characters without any errors
RX_CHARS	Total number of characters received
TX_CHARS	Total number of transmitted characters
RX_PARITY_ERRORS	Number of parity errors
RX_FRAMING_ERRORS	Number of framing errors
RX_OVERRUNS	Number of overrun errors
RX_BREAKS	Number of break errors

IO_IOCTL_SERIAL_GET_CONFIG

This IOCTL is used to get the current configuration of the UART.

```

typedef struct dw_apb_uart_init_struct
{
    pointer      COM_PORT_ADDRESS;
    _mqx_uint    QUEUE_SIZE;
    uint_32      BAUD_RATE;
    _mqx_uint    BITS_PER_CHARACTER;
    _mqx_uint    PARITY;
    _mqx_uint    STOP_BITS;
    _mqx_uint    VECTOR;
}DW_APB_UART_INIT_STRUCT, _PTR_ DW_APB_UART_INIT_STRUCT_PTR;

```

Field Name	Description
COM_PORT_ADDRESS	Base address of the UART memory where device is mapped. This value is used by the driver to access the registers.
QUEUE_SIZE	Size of the queue used by the driver to buffer incoming and outgoing data.

BAUD_RATE	Baud rate at which the UART is to be initialized. Most UARTs are initialized to the values 9600 and 115200.
BITS_PER_CHARACTER	Number of bits in a character. Possible values are 5, 6, 7, and 8.
PARITY	Parity bits at which the UART is to be initialized. Possible parity values are none, even, and odd.
STOP_BITS	Number of stop bits at which device is to be initialized. Possible values are 1, 1.5, and 2.
VECTOR	Interrupt vector number for this UART.

IO_IOCTL_SERIAL_GET_HW_SIGNAL/ IO_IOCTL_SERIAL_SET_HW_SIGNAL/ IO_IOCTL_SERIAL_CLEAR_HW_SIGNAL

This IOCTL is used to get, set, and clear the hardware signals (modem control signals) of the UART.

```
uint_32_ptr  ret_buf
```

Field Name	Description
ret_buf	This field can have the following values: <pre>#define IO_SERIAL_CTS (1) #define IO_SERIAL_RTS (2) #define IO_SERIAL_DTR (4) #define IO_SERIAL_DSR (8)</pre>

IO_IOCTL_SERIAL_GET_PARITY/ IO_IOCTL_SERIAL_SET_PARITY

This IOCTL is used to get and set the parity of the UART.

```
uint_32_ptr  ret_buf
```

Field Name	Description
ret_buf	This field can have the following values:

	#define IO_SERIAL_PARITY_NONE	(1)
	#define IO_SERIAL_PARITY_ODD	(2)
	#define IO_SERIAL_PARITY_EVEN	(3)

Default Configuration for DW APB UART

The default values with which the DW APB UART is configured during initialization is explained in the table below. These values are passed to the driver using the `struct dw_apb_uart_init_struct`.

3

Test Application

A test application is provided with the UART driver. It tests all features of the MQX driver for the DW APB UART. The result of each test is displayed on the program output window of the debugger.