

IOB-UART, a RISC-V UART

Software User Guide, V0.1 , Build 4d20429



May 24, 2022



1 Module Documentation	1
1.1 IOb-UART Software Drivers	1
1.1.1 Introduction	1
1.1.2 IOb-UART Macro Values	2
1.1.2.1 ACK	2
1.1.2.2 ENQ	2
1.1.2.3 EOT	2
1.1.2.4 ETX	3
1.1.2.5 FRX	3
1.1.2.6 FTX	3
1.1.2.7 STX	3
1.1.2.8 UART_PROGNAME	3
1.1.3 IOb-UART Functions	3
1.1.3.1 uart_finish()	3
1.1.3.2 uart_getc()	4
1.1.3.3 uart_init()	4
1.1.3.4 uart_putc()	5
1.1.3.5 uart_puts()	5
1.1.3.6 uart_recvfile()	6
1.1.3.7 uart_rxwait()	6
1.1.3.8 uart_sendfile()	7
1.1.3.9 uart_txwait()	7



Chapter 1

Module Documentation

1.1 IOb-UART Software Drivers

IOb-UART Software Drivers.

1.1.1 Introduction

IOb-UART Software Drivers.

The present IOb-UART software drivers implement a way to interface with the IOb-UART peripheral for serial communication.

The present drivers provide base functionalities such as:

- initialization and setup
- basic control functions
- single character send and receive functions
- simple protocol for multi byte transfers

IOb-UART Defined Macros

- `#define UART_PROGNAME "IOb-UART"`
- `#define STX 2`
- `#define ETX 3`
- `#define EOT 4`
- `#define ENQ 5`
- `#define ACK 6`
- `#define FTX 7`
- `#define FRX 8`



IOb-UART Function Signatures

- void `uart_init` (int base_address, uint16_t div)
Initialize UART.
- void `uart_finish` ()
Close transmission.
- void `uart_txdwait` ()
Wait for TX.
- void `uart_putc` (char c)
Print char.
- void `uart_puts` (const char *s)
Print string.
- void `uart_sendfile` (char *file_name, int file_size, char *mem)
Send file.
- void `uart_rxdwait` ()
Wait for RX Data.
- char `uart_getc` ()
Get char.
- int `uart_rcvfile` (char *file_name, char **mem)
Receive file.

1.1.2 IOb-UART Macro Values

1.1.2.1 ACK

```
#define ACK 6
```

Acknowledge. Signal reception of incoming message.

1.1.2.2 ENQ

```
#define ENQ 5
```

Enquiry. Signal start of UART connection.

1.1.2.3 EOT

```
#define EOT 4
```

End of transmission. Signal end of UART connection.



1.1.2.4 ETX

```
#define ETX 3
```

End text. Signal end of data sequence to be printed.

1.1.2.5 FRX

```
#define FRX 8
```

File reception. Signal file reception request.

1.1.2.6 FTX

```
#define FTX 7
```

File transfer. Signal file transfer request.

1.1.2.7 STX

```
#define STX 2
```

Start text. Signal start of data sequence to be printed.

1.1.2.8 UART_PROGNAME

```
#define UART_PROGNAME "IOb-UART"
```

Prefix to IOb-Uart specific prints.

1.1.3 IOb-UART Functions

1.1.3.1 uart_finish()

```
void uart_finish ( )
```

Close transmission.

Send end of transmission (EOT) command via UART. Active wait until TX transfer is complete. Use this function to close console program.

Returns

void.



1.1.3.2 uart_getc()

```
char uart_getc ( )
```

Get char.

Active wait and receive char/byte from UART.

Returns

received byte from UART.

1.1.3.3 uart_init()

```
void uart_init (
    int base_address,
    uint16_t div )
```

Initialize UART.

Reset UART, set IOb-Uart base address and set the division factor. The division factor is the number of clock cycles per simbol transfered.

For example, for a case with fclk = 100 Mhz for a baudrate of 115200 we should have $div = (100 * 10^6 / 115200) = (868)$.

The following code is a simple usage example:

```
#include "iob-uart.h"
#define UART_BASE (0x80000000)
#define FREQ (100000000)
#define BAUD (115200)
int main()
{
    //init uart
    uart_init(UART_BASE, FREQ/BAUD);
    uart_puts("\n\nHello world!\n\n");
    uart_finish();
}
```

The IOb-UART is inicialized with UART_BASE as the memory address and $div = (FREQ/BAUD)$.

Parameters

<i>base_address</i>	IOb-Uart instance base address in the system.
<i>div</i>	Equal to round (fclk/baudrate).



Returns

void.

1.1.3.4 uart_putc()

```
void uart_putc (
    char c )
```

Print char.

Send character via UART to be printed by in console program.

Parameters

c	Character to print.
---	---------------------

Returns

void.

1.1.3.5 uart_puts()

```
void uart_puts (
    const char * s )
```

Print string.

Send string via UART to be printed by in console program.

Parameters

s	Pointer to char array to be printed.
---	--------------------------------------

Returns

void.

1.1.3.6 uart_recvfile()

```
int uart_recvfile (
    char * file_name,
    char ** mem )
```

Receive file.

Request variable size file via UART. Order of commands:

1. Send file receive (FRX) command.
2. Send file_name.
3. Receive file_size (in little endian format).
4. Send ACK command.
5. Receive file.

If memory pointer is not initialized, allocates memory for incoming file.

Parameters

<i>file_name</i>	Pointer to file name string.
<i>mem</i>	Pointer in memory to store incoming file.

Returns

Size of received file.

1.1.3.7 uart_rxwait()

```
void uart_rxwait ( )
```

Wait for RX Data.

Active wait for RX incoming data.

Returns

void.



1.1.3.8 uart_sendfile()

```
void uart_sendfile (
    char * file_name,
    int file_size,
    char * mem )
```

Send file.

Send variable size file via UART. Order of commands:

1. Send file transmit (FTX) command.
2. Send file_name.
3. Send file_size (in little endian format).
4. Send file.

Parameters

<i>file_name</i>	Pointer to file name string.
<i>file_size</i>	Size of file to be sent.
<i>mem</i>	Pointer to file.

Returns

void.

1.1.3.9 uart_txwait()

```
void uart_txwait ( )
```

Wait for TX.

Active wait until TX is ready to process new byte to send.

Returns

void.