

Echo uart

Generated by Doxygen 1.8.18



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Design Unit Hierarchy	1
<b>2 Design Unit Index</b>	<b>3</b>
2.1 Design Unit List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 Behavioral Architecture Reference	7
4.1.1 Member Function Documentation	8
4.1.1.1 PROCESS_0()	8
4.1.2 Member Data Documentation	8
4.1.2.1 Clk	8
4.1.2.2 Reset	8
4.1.2.3 Rx	8
4.1.2.4 RX_Data	8
4.1.2.5 RX_Ready	8
4.1.2.6 Tx	9
4.1.2.7 TX_Data	9
4.1.2.8 TX_Ready	9
4.1.2.9 TX_Start	9
4.1.2.10 uart_rx	9
4.1.2.11 uart_tx	9
4.2 Behavioral Architecture Reference	10
4.2.1 Member Function Documentation	10
4.2.1.1 RX_PROCESS()	10
4.2.1.2 TO_OUTPUT()	10
4.2.2 Member Data Documentation	11
4.2.2.1 count	11
4.2.2.2 data_buf	11
4.2.2.3 data_ready	11
4.2.2.4 freq_count	11
4.2.2.5 last_Rx	11
4.2.2.6 MAX_FREQ_COUNT	12
4.2.2.7 receiving	12
4.3 Behavioral Architecture Reference	12
4.3.1 Member Function Documentation	12
4.3.1.1 TX_PROCESS()	12
4.3.2 Member Data Documentation	13
4.3.2.1 count	13
4.3.2.2 freq_count	13

4.3.2.3 MAX_FREQ_COUNT . . . . .	13
4.4 main Entity Reference . . . . .	13
4.4.1 Member Data Documentation . . . . .	14
4.4.1.1 BAUD . . . . .	14
4.4.1.2 CLK_FREQUENCY . . . . .	14
4.4.1.3 Clk_input . . . . .	15
4.4.1.4 DATA_WIDTH . . . . .	15
4.4.1.5 IEEE . . . . .	15
4.4.1.6 NUMERIC_STD . . . . .	15
4.4.1.7 Reset_input . . . . .	15
4.4.1.8 Rx_input . . . . .	15
4.4.1.9 STD_LOGIC_1164 . . . . .	16
4.4.1.10 Tx_output . . . . .	16
4.5 UART_RX Entity Reference . . . . .	16
4.5.1 Detailed Description . . . . .	17
4.5.2 Member Data Documentation . . . . .	17
4.5.2.1 BAUD . . . . .	17
4.5.2.2 Clk . . . . .	17
4.5.2.3 CLK_FREQUENCY . . . . .	17
4.5.2.4 DATA_WIDTH . . . . .	18
4.5.2.5 IEEE . . . . .	18
4.5.2.6 NUMERIC_STD . . . . .	18
4.5.2.7 Reset . . . . .	18
4.5.2.8 Rx . . . . .	18
4.5.2.9 RX_Data_Out . . . . .	18
4.5.2.10 RX_Ready . . . . .	19
4.5.2.11 STD_LOGIC_1164 . . . . .	19
4.6 uart_tx Entity Reference . . . . .	19
4.6.1 Detailed Description . . . . .	20
4.6.2 Member Data Documentation . . . . .	20
4.6.2.1 BAUD . . . . .	20
4.6.2.2 Clk . . . . .	20
4.6.2.3 CLK_FREQUENCY . . . . .	20
4.6.2.4 DATA_WIDTH . . . . .	21
4.6.2.5 IEEE . . . . .	21
4.6.2.6 NUMERIC_STD . . . . .	21
4.6.2.7 Reset . . . . .	21
4.6.2.8 STD_LOGIC_1164 . . . . .	21
4.6.2.9 Tx . . . . .	21
4.6.2.10 TX_Data_In . . . . .	22
4.6.2.11 TX_Ready . . . . .	22
4.6.2.12 TX_Start . . . . .	22

---

<b>5 File Documentation</b>	<b>23</b>
5.1 elbertv2_pin.ucf File Reference . . . . .	23
5.1.1 Variable Documentation . . . . .	23
5.1.1.1 ""Clk_input[0]"" . . . . .	23
5.1.1.2 ""Rx_input[0]"" . . . . .	23
5.1.1.3 ""Tx_output[0]"" . . . . .	23
5.1.1.4 VCCAUX . . . . .	24
5.2 main.vhd File Reference . . . . .	24
5.3 uart_rx.vhd File Reference . . . . .	24
5.4 uart_tx.vhd File Reference . . . . .	24
<b>Index</b>	<b>25</b>



# Chapter 1

## Hierarchical Index

### 1.1 Design Unit Hierarchy

Here is a hierarchical list of all entities:

main . . . . .	13
UART_RX . . . . .	16
uart_tx . . . . .	19





## Chapter 2

# Design Unit Index

### 2.1 Design Unit List

Here is a list of all design unit members with links to the Entities they belong to:

architecture <a href="#">Behavioral</a> . . . . .	7
architecture <a href="#">Behavioral</a> . . . . .	10
architecture <a href="#">Behavioral</a> . . . . .	12
entity <a href="#">main</a> . . . . .	13
entity <a href="#">UART_RX</a>	
Definition of UART RX . . . . .	16
entity <a href="#">uart_tx</a>	
Definition of UART TX . . . . .	19



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">elbertv2_pin.ucf</a>	23
<a href="#">main.vhd</a>	24
<a href="#">uart_rx.vhd</a>	24
<a href="#">uart_tx.vhd</a>	24



# Chapter 4

## Class Documentation

### 4.1 Behavioral Architecture Reference

#### Processes

- `PROCESS_0( Clk )`

#### Signals

- `Clk std_logic`
- `Reset std_logic`
- `Rx std_logic`  
*signal for get*
- `Tx std_logic`  
*signal for transmitting*
- `RX_Data std_logic_vector(DATA_WIDTH - 1 downto 0 )`  
*Inputs from uart\_rx.*
- `RX_Ready std_logic`  
*Inputs from uart\_rx.*
- `TX_Data std_logic_vector( 7 downto 0 )`  
*Inputs from uart\_tx.*
- `TX_Ready std_logic:= ' 0 '`  
*Inputs from uart\_tx.*
- `TX_Start std_logic`  
*Outputs from uart\_tx.*

#### Instantiations

- `uart_rx UART_RX`  
*declaration uart\_rx*
- `uart_tx uart_tx`  
*declaration uart\_tx*

## 4.1.1 Member Function Documentation

### 4.1.1.1 PROCESS\_0()

```
PROCESS_0 (
    Clk )
```

## 4.1.2 Member Data Documentation

### 4.1.2.1 Clk

```
Clk std_logic [Signal]
```

### 4.1.2.2 Reset

```
Reset std_logic [Signal]
```

### 4.1.2.3 Rx

```
Rx std_logic [Signal]
```

signal for get

### 4.1.2.4 RX\_Data

```
RX_Data std_logic_vector (DATA_WIDTH - 1 downto 0 ) [Signal]
```

Inputs from uart\_rx.

### 4.1.2.5 RX\_Ready

```
RX_Ready std_logic [Signal]
```

Inputs from uart\_rx.

#### 4.1.2.6 Tx

`Tx` `std_logic` [Signal]

signal for transmitting

#### 4.1.2.7 TX\_Data

`TX_Data` `std_logic_vector( 7 downto 0 )` [Signal]

Inputs from `uart_tx`.

#### 4.1.2.8 TX\_Ready

`TX_Ready` `std_logic:= ' 0 '` [Signal]

Inputs from `uart_tx`.

#### 4.1.2.9 TX\_Start

`TX_Start` `std_logic` [Signal]

Outputs from `uart_tx`.

#### 4.1.2.10 uart\_rx

`uart_rx` `UART_RX` [Instantiation]

declaration `uart_rx`

#### 4.1.2.11 uart\_tx

`uart_tx` `uart_tx` [Instantiation]

declaration `uart_tx`

The documentation for this class was generated from the following file:

- `main.vhd`

## 4.2 Behavioral Architecture Reference

### Processes

- `RX_PROCESS( Clk , Reset )`  
*waiting for data frame*
- `TO_OUTPUT( Clk )`  
*receiving data frame*

### Constants

- `MAX_FREQ_COUNT` **positive:=CLK\_FREQUENCY/BAUD**  
*length of one bit in clock cycles*

### Signals

- `freq_count` **naturallrange 0 toMAX\_FREQ\_COUNT - 1**  
*used for counting clock cycles*
- `count` **naturallrange 0 toDATA\_WIDTH + 2**  
*counting received bits*
- `last_Rx` **std\_logic**  
*temporarily keeps last state of Rx input*
- `receiving` **std\_logic:= ' 0 '**  
*determinates if process is in receiving state*
- `data_buf` **std\_logic\_vector( 0 toDATA\_WIDTH + 2 )**  
*buffer for incoming uart frame, 'to' is used for reverse trick on output assignment*
- `data_ready` **std\_logic:= ' 0 '**  
*determinates if data is ready to send to the output*

### 4.2.1 Member Function Documentation

#### 4.2.1.1 RX\_PROCESS()

```
RX_PROCESS (
    Clk      ,
    Reset    )  [Process]
```

waiting for data frame

#### 4.2.1.2 TO\_OUTPUT()

```
TO_OUTPUT (
    Clk      )  [Process]
```

receiving data frame



## 4.2.2 Member Data Documentation

### 4.2.2.1 count

```
count naturalrange 0 to DATA_WIDTH + 2 [Signal]
```

counting received bits

### 4.2.2.2 data\_buf

```
data_buf std_logic_vector( 0 to DATA_WIDTH + 2 ) [Signal]
```

buffer for incoming uart frame, 'to' is used for reverse trick on output assignment

### 4.2.2.3 data\_ready

```
data_ready std_logic := ' 0 ' [Signal]
```

determinates if data is ready to send to the output

### 4.2.2.4 freq\_count

```
freq_count naturalrange 0 to MAX_FREQ_COUNT - 1 [Signal]
```

used for counting clock cycles

### 4.2.2.5 last\_Rx

```
last_Rx std_logic [Signal]
```

temporarily keeps last state of Rx input

#### 4.2.2.6 MAX\_FREQ\_COUNT

`MAX_FREQ_COUNT` `positive:=CLK_FREQUENCY /BAUD` [Constant]

length of one bit in clock cycles

#### 4.2.2.7 receiving

`receiving` `std_logic:=' 0 '` [Signal]

determinates if process is in receiving state

The documentation for this class was generated from the following file:

- `uart_rx.vhd`

### 4.3 Behavioral Architecture Reference

#### Processes

- `TX_PROCESS( Clk , Reset )`

#### Constants

- `MAX_FREQ_COUNT` `positive:=CLK_FREQUENCY /BAUD`  
*length of one bit in clock cycles*

#### Signals

- `freq_count` `naturallrange 0 toMAX_FREQ_COUNT - 1`  
*used for counting clock cycles*
- `count` `naturallrange 0 to 11 := 11`  
*counting received bits*

#### 4.3.1 Member Function Documentation

##### 4.3.1.1 TX\_PROCESS()

```
TX_PROCESS (
    Clk      ,
    Reset    ) [Process]
```

### 4.3.2 Member Data Documentation

#### 4.3.2.1 count

```
count naturalrange 0 to 11 := 11 [Signal]
```

counting received bits

#### 4.3.2.2 freq\_count

```
freq_count naturalrange 0 to MAX_FREQ_COUNT - 1 [Signal]
```

used for counting clock cycles

#### 4.3.2.3 MAX\_FREQ\_COUNT

```
MAX_FREQ_COUNT positive:=CLK_FREQUENCY /BAUD [Constant]
```

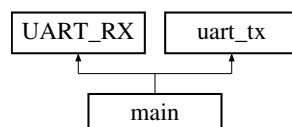
length of one bit in clock cycles

The documentation for this class was generated from the following file:

- [uart\\_tx.vhd](#)

## 4.4 main Entity Reference

Inheritance diagram for main:



### Entities

- [Behavioral](#) architecture

## Libraries

- [IEEE](#)  
*use standard library*

## Use Clauses

- [STD\\_LOGIC\\_1164](#)  
*use logic elements*
- [NUMERIC\\_STD](#)  
*use numeric elements*

## Generics

- [CLK\\_FREQUENCY](#) **positive:= 12000000**  
*clock frequency*
- [DATA\\_WIDTH](#) **positive:= 8**  
*UART message length.*
- [BAUD](#) **positive:= 19200**  
*UART baud rate.*

## Ports

- [Clk\\_input](#) in [std\\_logic\\_vector\( 0 downto 0 \)](#)
- [Reset\\_input](#) in [std\\_logic\\_vector\( 0 downto 0 \)](#)
- [Rx\\_input](#) in [std\\_logic\\_vector\( 0 downto 0 \)](#)  
*RX pin to get signals.*
- [Tx\\_output](#) out [std\\_logic\\_vector\( 0 downto 0 \)](#)  
*Tx pin for transmitting.*

## 4.4.1 Member Data Documentation

### 4.4.1.1 BAUD

[BAUD](#) **positive:= 19200** [Generic]

UART baud rate.

### 4.4.1.2 CLK\_FREQUENCY

[CLK\\_FREQUENCY](#) **positive:= 12000000** [Generic]

clock frequency

#### 4.4.1.3 Clk\_input

```
Clk_input in std_logic_vector( 0 downto 0 ) [Port]
```

#### 4.4.1.4 DATA\_WIDTH

```
DATA_WIDTH positive:= 8 [Generic]
```

UART message length.

#### 4.4.1.5 IEEE

```
IEEE [Library]
```

use standard library

#### 4.4.1.6 NUMERIC\_STD

```
NUMERIC_STD [use clause]
```

use numeric elements

#### 4.4.1.7 Reset\_input

```
Reset_input in std_logic_vector( 0 downto 0 ) [Port]
```

#### 4.4.1.8 Rx\_input

```
Rx_input in std_logic_vector( 0 downto 0 ) [Port]
```

RX pin to get signals.

#### 4.4.1.9 STD\_LOGIC\_1164

`STD_LOGIC_1164` [use clause]

use logic elements

#### 4.4.1.10 Tx\_output

`Tx_output` out std\_logic\_vector( 0 downto 0 ) [Port]

Tx pin for transmitting.

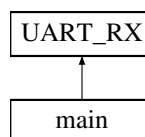
The documentation for this class was generated from the following file:

- [main.vhd](#)

## 4.5 UART\_RX Entity Reference

Definition of UART RX.

Inheritance diagram for UART\_RX:



### Entities

- [Behavioral](#) architecture

### Libraries

- [IEEE](#)  
*use standard library*

### Use Clauses

- [STD\\_LOGIC\\_1164](#)  
*use logic elements*
- [NUMERIC\\_STD](#)  
*use numeric elements*

## Generics

- `CLK_FREQUENCY` `positive:= 12000000`
- `DATA_WIDTH` `positive:= 8`  
*UART message length.*
- `BAUD` `positive:= 19200`  
*UART baud rate.*

## Ports

- `Clk` `in std_logic`
- `Reset` `in std_logic`
- `Rx` `in std_logic`  
*Rx pin for receiving.*
- `RX_Data_Out` `out std_logic_vector(DATA_WIDTH - 1 downto 0)`  
*received data*
- `RX_Ready` `out std_logic:= '0'`  
*determinates if data on output is ready*

### 4.5.1 Detailed Description

Definition of UART RX.

### 4.5.2 Member Data Documentation

#### 4.5.2.1 BAUD

`BAUD` `positive:= 19200` [Generic]

UART baud rate.

#### 4.5.2.2 Clk

`Clk` `in std_logic` [Port]

#### 4.5.2.3 CLK\_FREQUENCY

`CLK_FREQUENCY` `positive:= 12000000` [Generic]

#### 4.5.2.4 DATA\_WIDTH

`DATA_WIDTH` `positive:= 8` [Generic]

UART message length.

#### 4.5.2.5 IEEE

`IEEE` [Library]

use standard library

#### 4.5.2.6 NUMERIC\_STD

`NUMERIC_STD` [use clause]

use numeric elements

#### 4.5.2.7 Reset

`Reset` in `std_logic` [Port]

#### 4.5.2.8 Rx

`Rx` in `std_logic` [Port]

Rx pin for receiving.

#### 4.5.2.9 RX\_Data\_Out

`RX_Data_Out` out `std_logic_vector(DATA_WIDTH - 1 downto 0 )` [Port]

received data



#### 4.5.2.10 RX\_Ready

`RX_Ready` out `std_logic` := ' 0 ' [Port]

determinates if data on output is ready

#### 4.5.2.11 STD\_LOGIC\_1164

`STD_LOGIC_1164` [use clause]

use logic elements

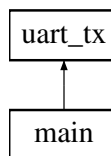
The documentation for this class was generated from the following file:

- [uart\\_rx.vhd](#)

## 4.6 uart\_tx Entity Reference

Definition of UART TX.

Inheritance diagram for `uart_tx`:



### Entities

- [Behavioral](#) architecture

### Libraries

- [IEEE](#)  
*use standart library*

### Use Clauses

- [STD\\_LOGIC\\_1164](#)  
*use logic elements*
- [NUMERIC\\_STD](#)  
*use numeric elements*

## Generics

- `CLK_FREQUENCY` `positive:= 12000000`
- `DATA_WIDTH` `positive:= 8`  
*UART message length.*
- `BAUD` `positive:= 19200`  
*UART baud rate.*

## Ports

- `Clk` `in std_logic`
- `Reset` `in std_logic`
- `TX_Data_In` `in std_logic_vector(DATA_WIDTH - 1 downto 0)`  
*data to transmit*
- `TX_Ready` `in std_logic`  
*definition when new data come*
- `Tx` `out std_logic:= '1'`  
*Tx pin for transmitting.*
- `TX_Start` `out std_logic:= '1'`  
*definition when new data can come*

### 4.6.1 Detailed Description

Definition of UART TX.

### 4.6.2 Member Data Documentation

#### 4.6.2.1 BAUD

`BAUD` `positive:= 19200` [Generic]

UART baud rate.

#### 4.6.2.2 Clk

`Clk` `in std_logic` [Port]

#### 4.6.2.3 CLK\_FREQUENCY

`CLK_FREQUENCY` `positive:= 12000000` [Generic]

#### 4.6.2.4 DATA\_WIDTH

`DATA_WIDTH` `positive:= 8` [Generic]

UART message length.

#### 4.6.2.5 IEEE

`IEEE` [Library]

use standart library

#### 4.6.2.6 NUMERIC\_STD

`NUMERIC_STD` [use clause]

use numeric elements

#### 4.6.2.7 Reset

`Reset` in `std_logic` [Port]

#### 4.6.2.8 STD\_LOGIC\_1164

`STD_LOGIC_1164` [use clause]

use logic elements

#### 4.6.2.9 Tx

`Tx` out `std_logic:= ' 1 '` [Port]

Tx pin for transmitting.

#### 4.6.2.10 TX\_Data\_In

`TX_Data_In in std_logic_vector(DATA_WIDTH - 1 downto 0 )` [Port]

data to transmit

#### 4.6.2.11 TX\_Ready

`TX_Ready in std_logic` [Port]

definition when new data come

#### 4.6.2.12 TX\_Start

`TX_Start out std_logic:=' 1 '` [Port]

definition when new data can come

The documentation for this class was generated from the following file:

- [uart\\_tx.vhd](#)

## Chapter 5

# File Documentation

### 5.1 elbertv2\_pin.ucf File Reference

#### Constraints

- `VCCAUX " 3 . 3 "`
- `"Clk_input[0]" LOC=P129|IOSTANDARD=LVCMOS33|PERIOD=12MHz`
- `"Rx_input[0]" LOC=P125|IOSTANDARD=LVCMOS33|SLEW=SLOW|DRIVE= 12`
- `"Tx_output[0]" LOC=P127|IOSTANDARD=LVCMOS33|SLEW=SLOW|DRIVE= 12`

#### 5.1.1 Variable Documentation

##### 5.1.1.1 `"Clk_input[0]"`

[Constraints]

##### 5.1.1.2 `"Rx_input[0]"`

[Constraints]

##### 5.1.1.3 `"Tx_output[0]"`

[Constraints]

#### 5.1.1.4 VCCAUX

[Constraints]

## 5.2 main.vhd File Reference

### Entities

- [main](#) entity
- [Behavioral](#) architecture

## 5.3 uart\_rx.vhd File Reference

### Entities

- [UART\\_RX](#) entity  
*Definition of UART RX.*
- [Behavioral](#) architecture

## 5.4 uart\_tx.vhd File Reference

### Entities

- [uart\\_tx](#) entity  
*Definition of UART TX.*
- [Behavioral](#) architecture

# Index

"Clk\_input[0]"  
    elbertv2\_pin.ucf, [23](#)  
"Rx\_input[0]"  
    elbertv2\_pin.ucf, [23](#)  
"Tx\_output[0]"  
    elbertv2\_pin.ucf, [23](#)

## BAUD

    main, [14](#)  
    UART\_RX, [17](#)  
    uart\_tx, [20](#)

## Behavioral, [7](#), [10](#), [12](#)

    Clk, [8](#)  
    count, [11](#), [13](#)  
    data\_buf, [11](#)  
    data\_ready, [11](#)  
    freq\_count, [11](#), [13](#)  
    last\_Rx, [11](#)  
    MAX\_FREQ\_COUNT, [11](#), [13](#)  
    PROCESS\_0, [8](#)  
    receiving, [12](#)  
    Reset, [8](#)  
    Rx, [8](#)  
    RX\_Data, [8](#)  
    RX\_PROCESS, [10](#)  
    RX\_Ready, [8](#)  
    TO\_OUTPUT, [10](#)  
    Tx, [8](#)  
    TX\_Data, [9](#)  
    TX\_PROCESS, [12](#)  
    TX\_Ready, [9](#)  
    TX\_Start, [9](#)  
    uart\_rx, [9](#)  
    uart\_tx, [9](#)

## Clk

    Behavioral, [8](#)  
    UART\_RX, [17](#)  
    uart\_tx, [20](#)

## CLK\_FREQUENCY

    main, [14](#)  
    UART\_RX, [17](#)  
    uart\_tx, [20](#)

## Clk\_input

    main, [14](#)

## count

    Behavioral, [11](#), [13](#)

## data\_buf

    Behavioral, [11](#)

## data\_ready

    Behavioral, [11](#)

## DATA\_WIDTH

    main, [15](#)  
    UART\_RX, [17](#)  
    uart\_tx, [20](#)

## elbertv2\_pin.ucf, [23](#)

    "Clk\_input[0]", [23](#)  
    "Rx\_input[0]", [23](#)  
    "Tx\_output[0]", [23](#)  
    VCCAUX, [23](#)

## freq\_count

    Behavioral, [11](#), [13](#)

## IEEE

    main, [15](#)  
    UART\_RX, [18](#)  
    uart\_tx, [21](#)

## last\_Rx

    Behavioral, [11](#)

## main, [13](#)

    BAUD, [14](#)  
    CLK\_FREQUENCY, [14](#)  
    Clk\_input, [14](#)  
    DATA\_WIDTH, [15](#)  
    IEEE, [15](#)  
    NUMERIC\_STD, [15](#)  
    Reset\_input, [15](#)  
    Rx\_input, [15](#)  
    STD\_LOGIC\_1164, [15](#)  
    Tx\_output, [16](#)

## main.vhd, [24](#)

## MAX\_FREQ\_COUNT

    Behavioral, [11](#), [13](#)

## NUMERIC\_STD

    main, [15](#)  
    UART\_RX, [18](#)  
    uart\_tx, [21](#)

## PROCESS\_0

    Behavioral, [8](#)

## receiving

    Behavioral, [12](#)

## Reset

    Behavioral, [8](#)

- UART\_RX, [18](#)
  - uart\_tx, [21](#)
- Reset\_input
  - main, [15](#)
- Rx
  - Behavioral, [8](#)
  - UART\_RX, [18](#)
- RX\_Data
  - Behavioral, [8](#)
- RX\_Data\_Out
  - UART\_RX, [18](#)
- Rx\_input
  - main, [15](#)
- RX\_PROCESS
  - Behavioral, [10](#)
- RX\_Ready
  - Behavioral, [8](#)
  - UART\_RX, [18](#)
- STD\_LOGIC\_1164
  - main, [15](#)
  - UART\_RX, [19](#)
  - uart\_tx, [21](#)
- TO\_OUTPUT
  - Behavioral, [10](#)
- Tx
  - Behavioral, [8](#)
  - uart\_tx, [21](#)
- TX\_Data
  - Behavioral, [9](#)
- TX\_Data\_In
  - uart\_tx, [21](#)
- Tx\_output
  - main, [16](#)
- TX\_PROCESS
  - Behavioral, [12](#)
- TX\_Ready
  - Behavioral, [9](#)
  - uart\_tx, [22](#)
- TX\_Start
  - Behavioral, [9](#)
  - uart\_tx, [22](#)
- UART\_RX, [16](#)
  - BAUD, [17](#)
  - Clk, [17](#)
  - CLK\_FREQUENCY, [17](#)
  - DATA\_WIDTH, [17](#)
  - IEEE, [18](#)
  - NUMERIC\_STD, [18](#)
  - Reset, [18](#)
  - Rx, [18](#)
  - RX\_Data\_Out, [18](#)
  - RX\_Ready, [18](#)
  - STD\_LOGIC\_1164, [19](#)
- uart\_rx
  - Behavioral, [9](#)
- uart\_rx.vhd, [24](#)
- uart\_tx, [19](#)
  - BAUD, [20](#)
  - Behavioral, [9](#)
  - Clk, [20](#)
  - CLK\_FREQUENCY, [20](#)
  - DATA\_WIDTH, [20](#)
  - IEEE, [21](#)
  - NUMERIC\_STD, [21](#)
  - Reset, [21](#)
  - STD\_LOGIC\_1164, [21](#)
  - Tx, [21](#)
  - TX\_Data\_In, [21](#)
  - TX\_Ready, [22](#)
  - TX\_Start, [22](#)
- uart\_tx.vhd, [24](#)
- VCCAUX
  - elbertv2\_pin.ucf, [23](#)