



# Esterel Technologies Training Material

Lab #1 - A Simple UART  
For Academic Program Only

# Restricted Usage Conditions

- This material is provided to help academics build Esterel Studio and Esterel language training courses
- Usage is exclusively restricted to teaching activities for university and training institution students
- No industrial usage authorized
- Please, give us your feed-back, tell us about further needs and share other training material and labs.



**Sylvie Aldebert**

Academic Program Coordinator

Esterel Technologies

TWINS 1 - 679 avenue Julien Lefèbvre

06270 VILLENEUVE-LOUBET

FRANCE

**Phone:** +33 (0)4 92 02 40 40

**Phone:** +33 (0)4 92 02 40 64

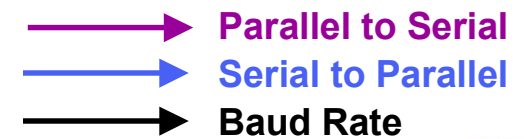
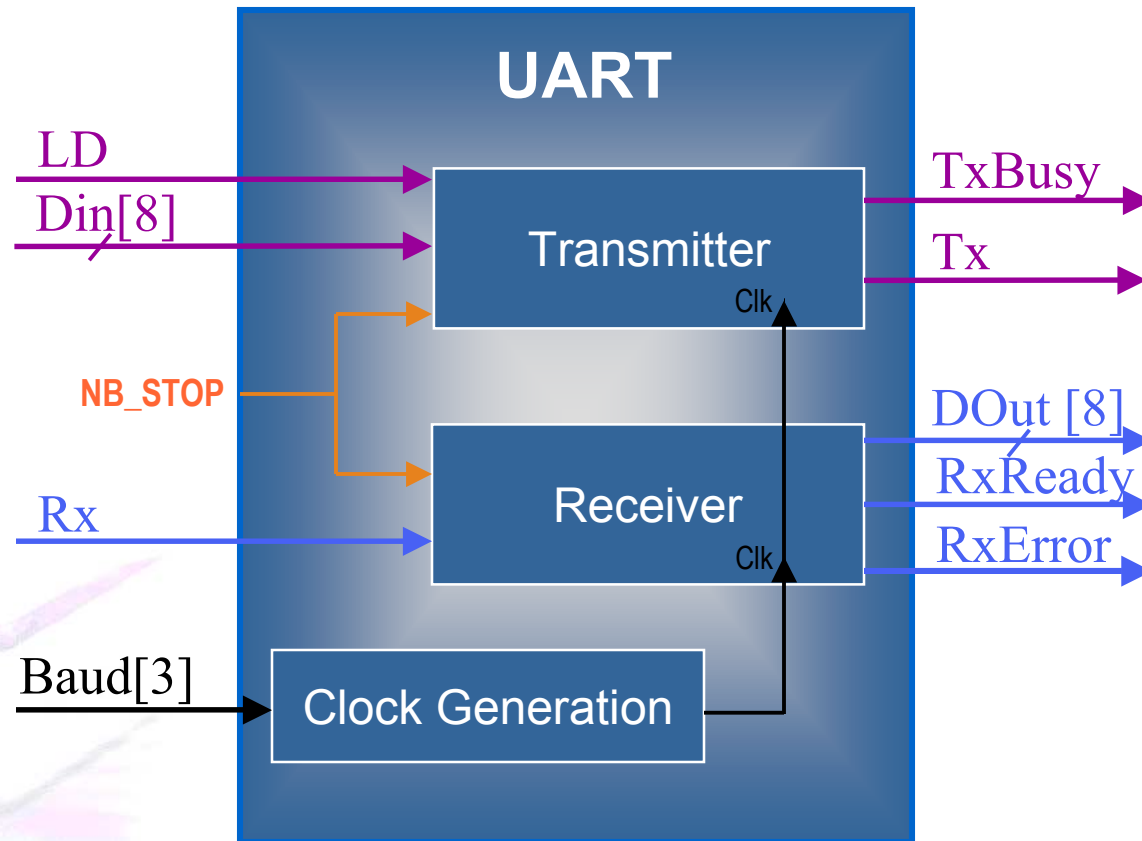
**Email:** [academic@esterel-technologies.com](mailto:academic@esterel-technologies.com)



# A Simple UART

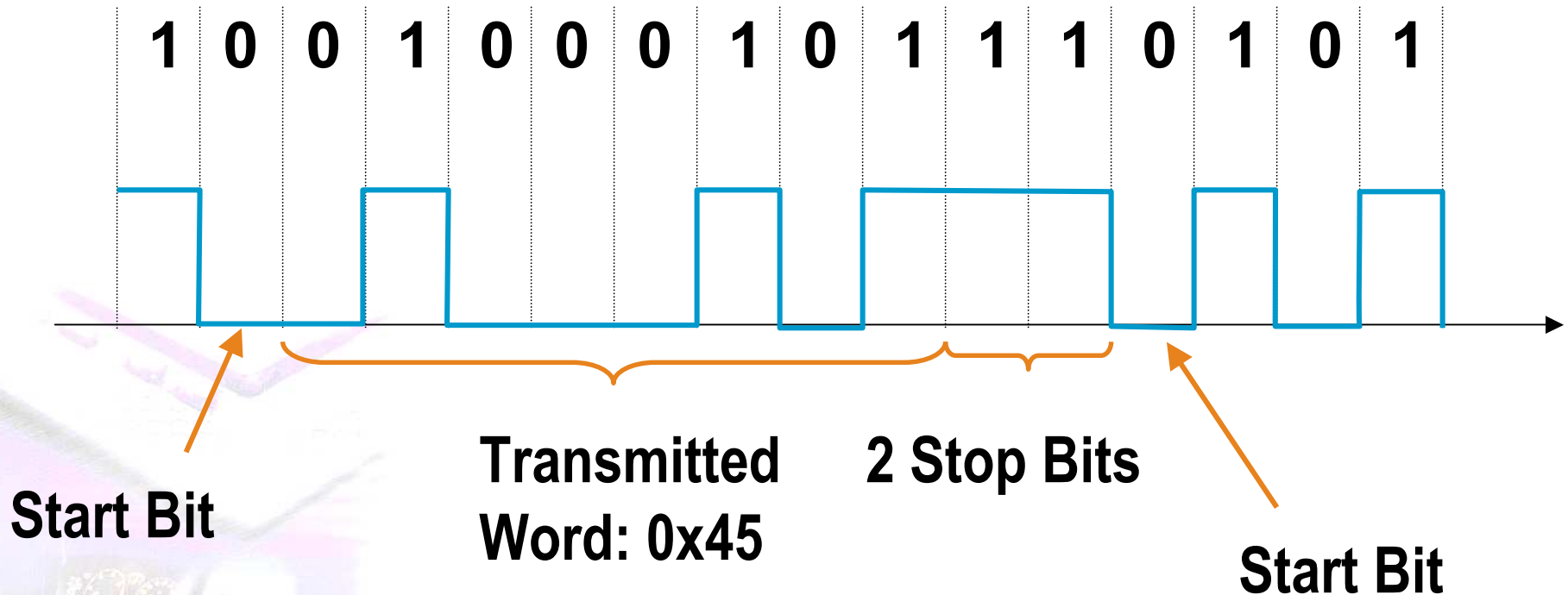


## UART Architecture



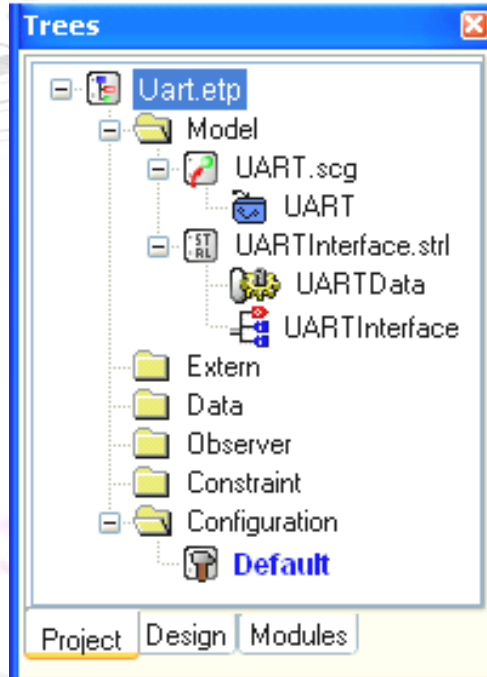


## Serial Transmitted Data





## UART Interface



**data** **UARTData:**

**constant** DATA\_SIZE: unsigned<> = 8;

**constant** NB\_STOP: unsigned<> = 3;

**end data**

**interface** **UARTInterface:**

**extends** UARTData;

**input** DIn: bool[DATA\_SIZE];

**input** LD;

**input** Rx;

**input** Baud : bool[3];

**output** Tx;

**output** TxBusy;

**output** DOut: bool[DATA\_SIZE];

**output** RxReady;

**output** RxError;

**end interface**






## Exercise 1

- 
- Design the receiver and transmitter modules.  
Consider only one start bit and one stop bit



## Exercise 2

- 
- Update your model to take into account one start bit, NB\_STOP stop bits and the parity bit management



## Exercise 3

- 
- Connect together 2 instances of the UART



## Exercise 4

- 
- Verify that the data sent are always received



## Exercise 5

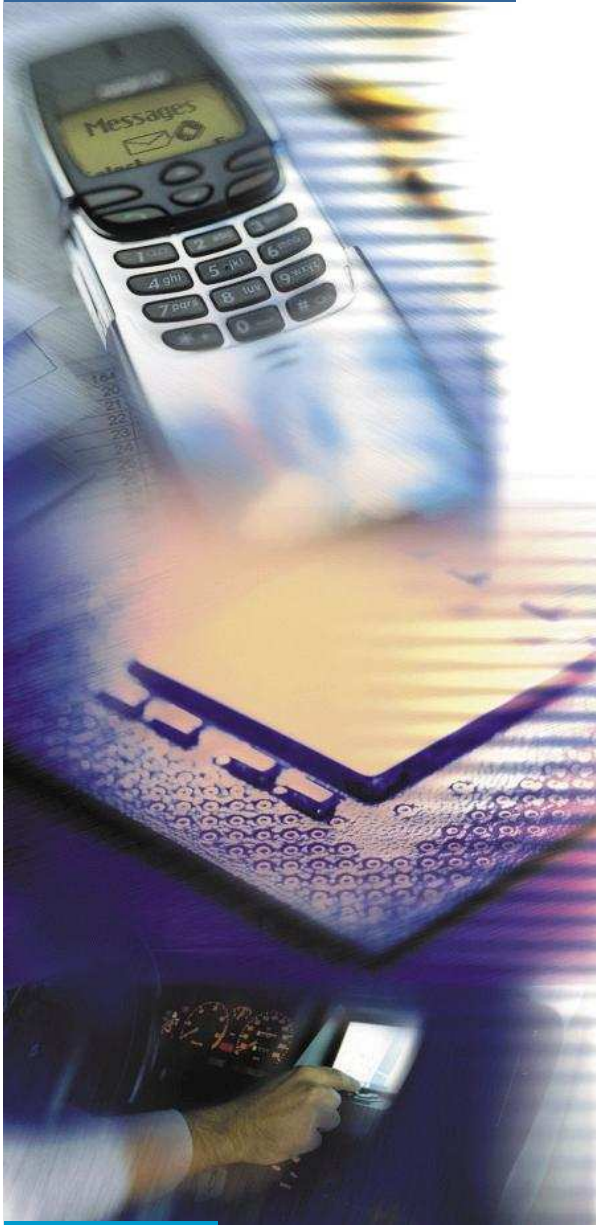
- 
- Generate different clocks to play with the communication Baud Rate





# Lab Solutions

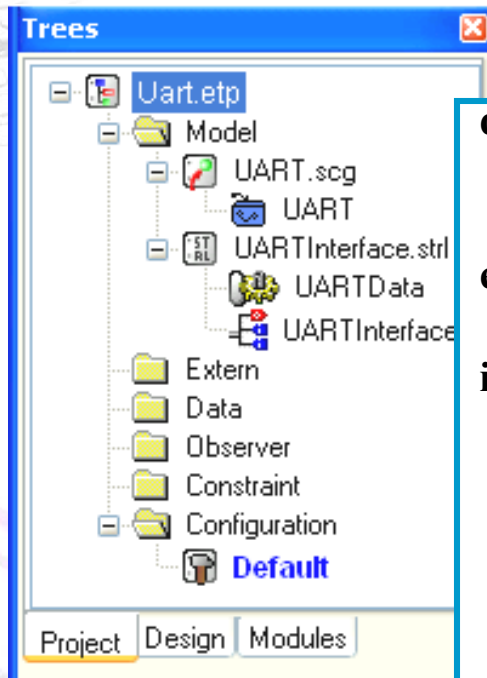
## A Simple UART



# Simple UART: Solutions

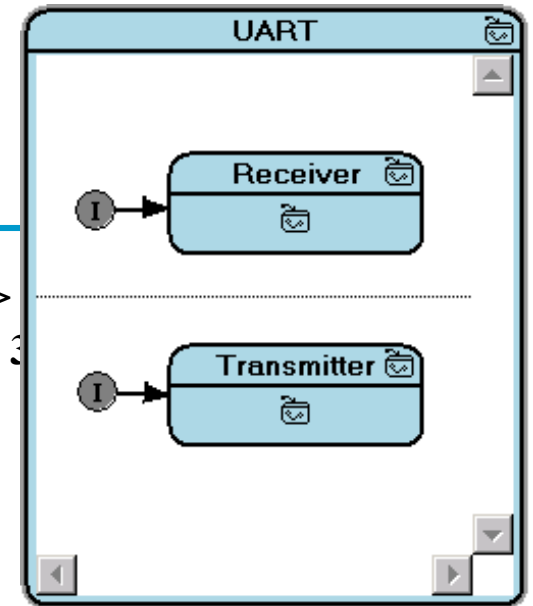


## Exercise 1



```
data UARTData:  
  constant DATA_SIZE: unsigned<>  
  constant NB_STOP: unsigned<> = 3  
end data
```

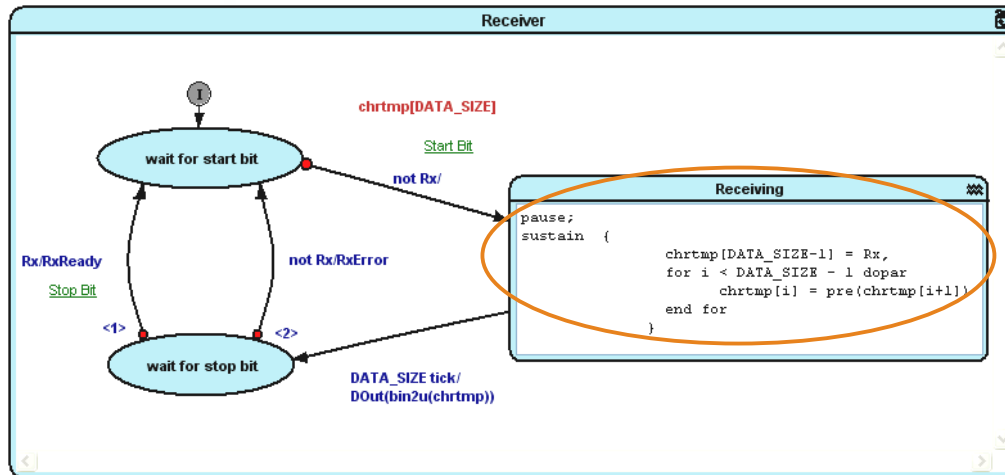
```
interface UARTInterface:  
  extends UARTData;  
  input DIn: bool[DATA_SIZE];  
  input LD;  
  input Rx;  
  input Baud: bool[3];  
  output Tx;  
  output TxBusy;  
  output DOut: bool[DATA_SIZE];  
  output RxReady;  
  output RxError;  
end interface
```



# Simple UART: Solutions



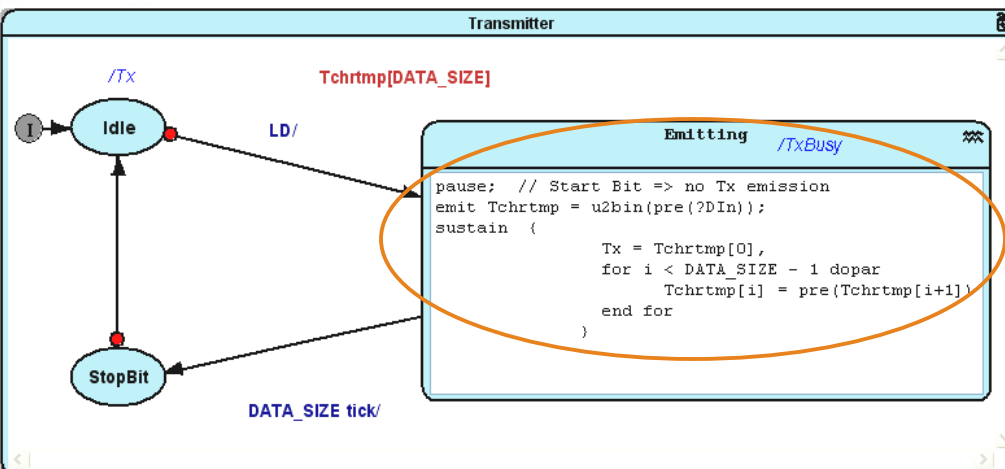
## Exercise 1



```

    pause;
    sustain {
        chrtmp[DATA_SIZE-1] = Rx,
        for i < DATA_SIZE - 1 dopar
            chrtmp[i] = pre(chrtmp[i+1])
        end for
    }
  
```

## Shifted Registers



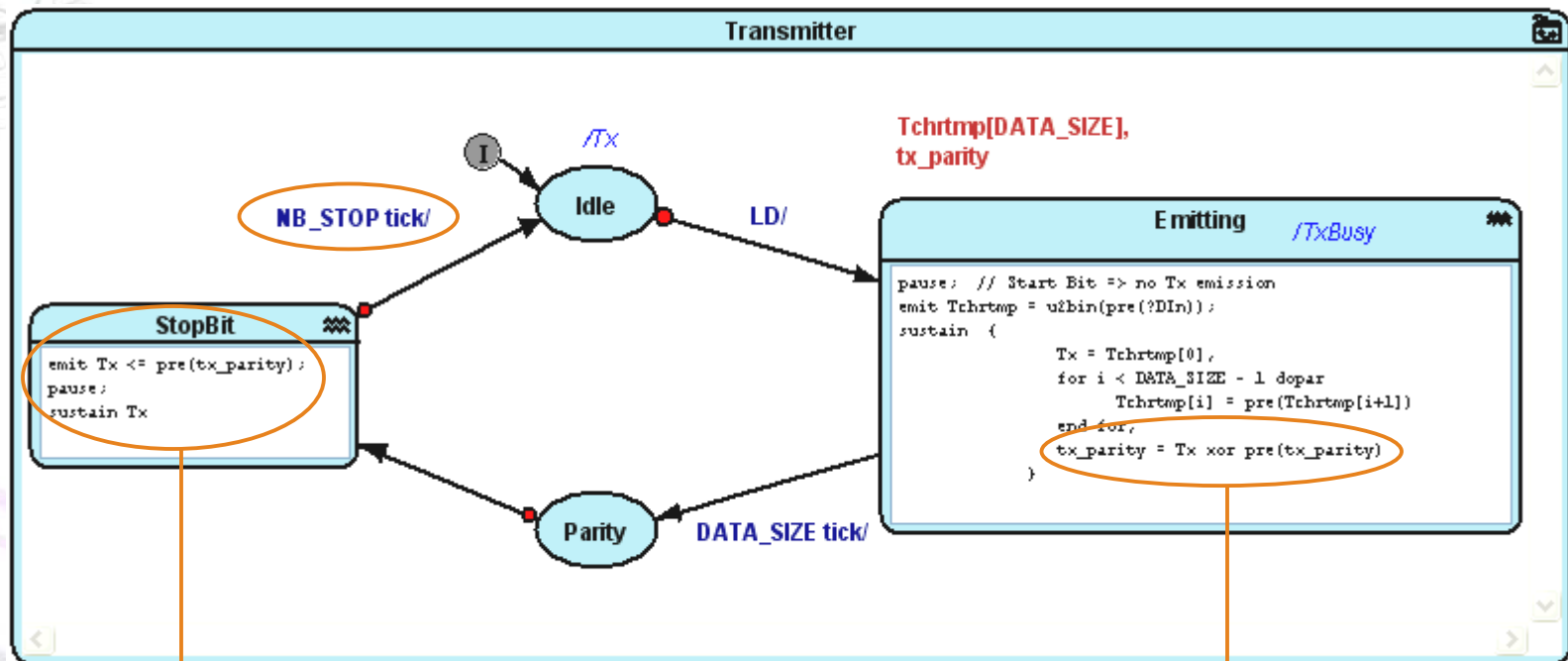
```

    pause; // Start Bit => no Tx emission
    emit ?Tchrtmp = pre(?DIn);
    sustain {
        Tx = ?Tchrtmp[0],
        for i < DATA_SIZE - 1 dopar
            ?Tchrtmp[i] = pre(Tchrtmp[i+1])
        end for
    }
  
```

# UART with Parity Bit and Stop bits



## Exercise 2



**Parity Bit & Stop Bits Emission**

```
emit Tx <= pre(tx_parity);
pause;
sustain Tx
```

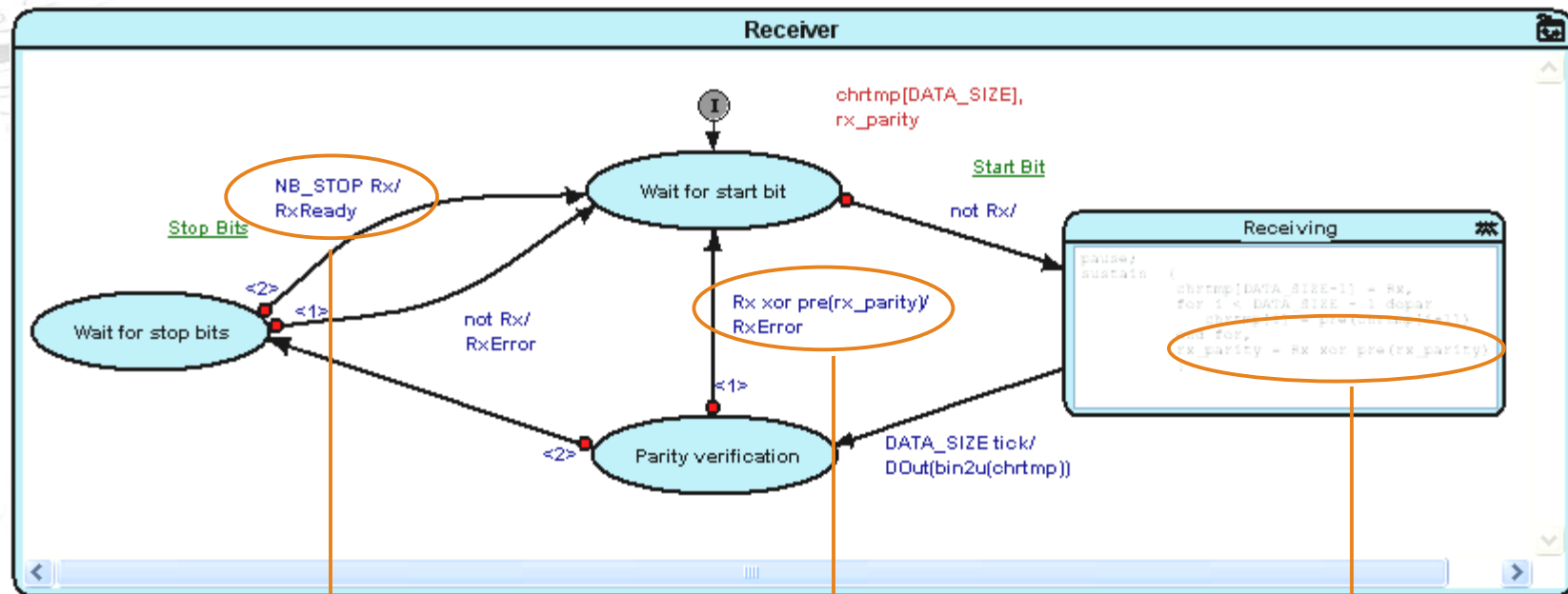
**Parity Bit Computation**

```
tx_parity = Tx xor pre(tx_parity)
```

# UART with Parity Bit and Stop bits



## Exercise 2



Stop Bits  
Detection

Parity Bit  
Verification

Parity Bit  
Computation

$rx\_parity = Rx \text{ xor } pre(rx\_parity)$

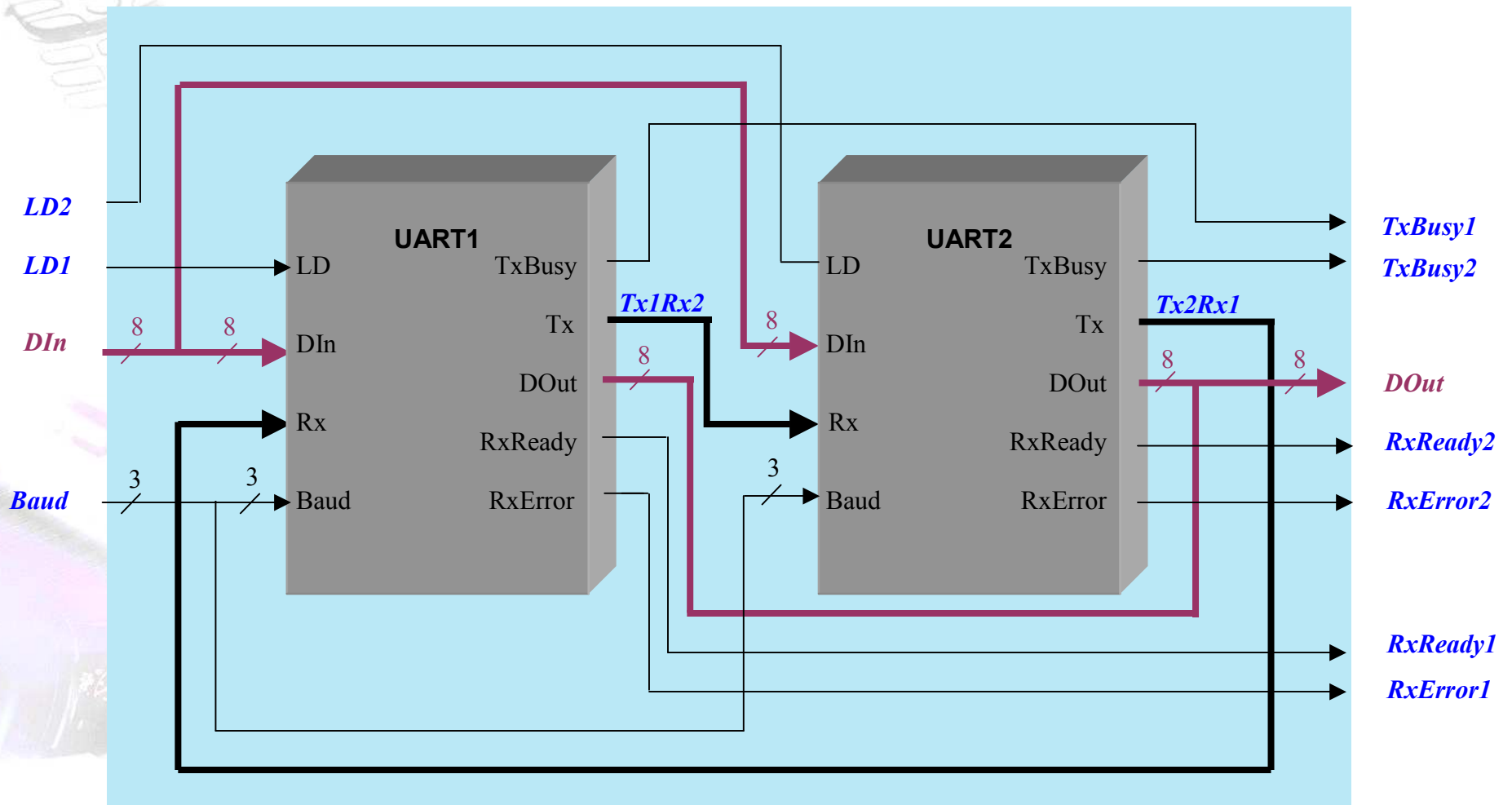
# Two instances of the UART



## Exercise 3

### Structural Diagram / Renaming

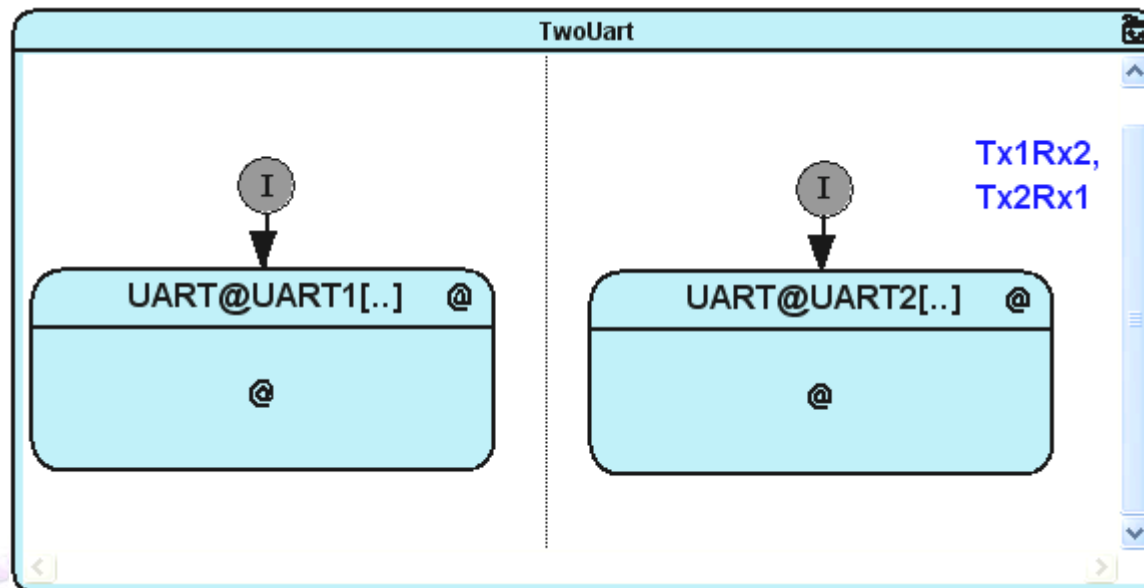
 *Renaming*  
 *In/Out flow*



# Two instances of the UART



## Exercise 3

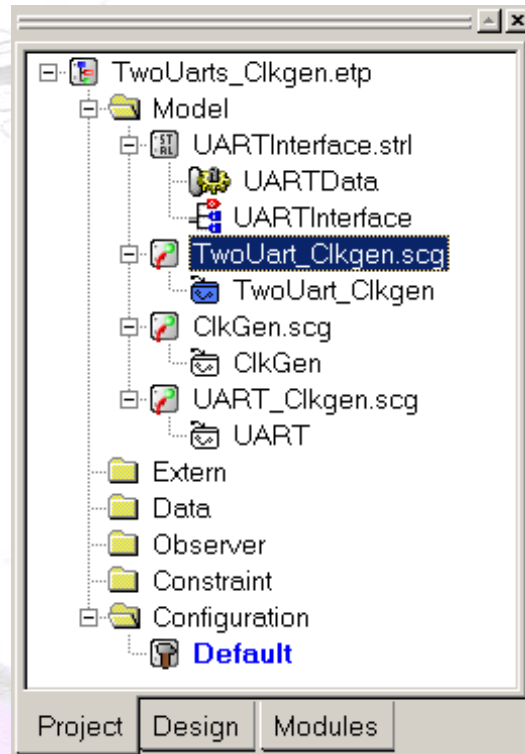




# Two UARTs with Loop Back



## Top Level



## Top Level Interface

extends data UARTData;

input **DIn**: bool[DATA\_SIZE];

input **LD1,LD2**;

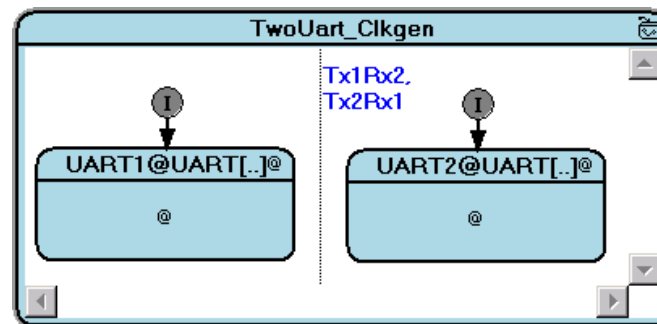
input **Baud**[3];

output **DOut**: value bool[DATA\_SIZE];

output **TxBusy1,TxBusy2**;

output **RxReady1,RxReady2**;

output **RxError1,RxError2**;



## Renamings

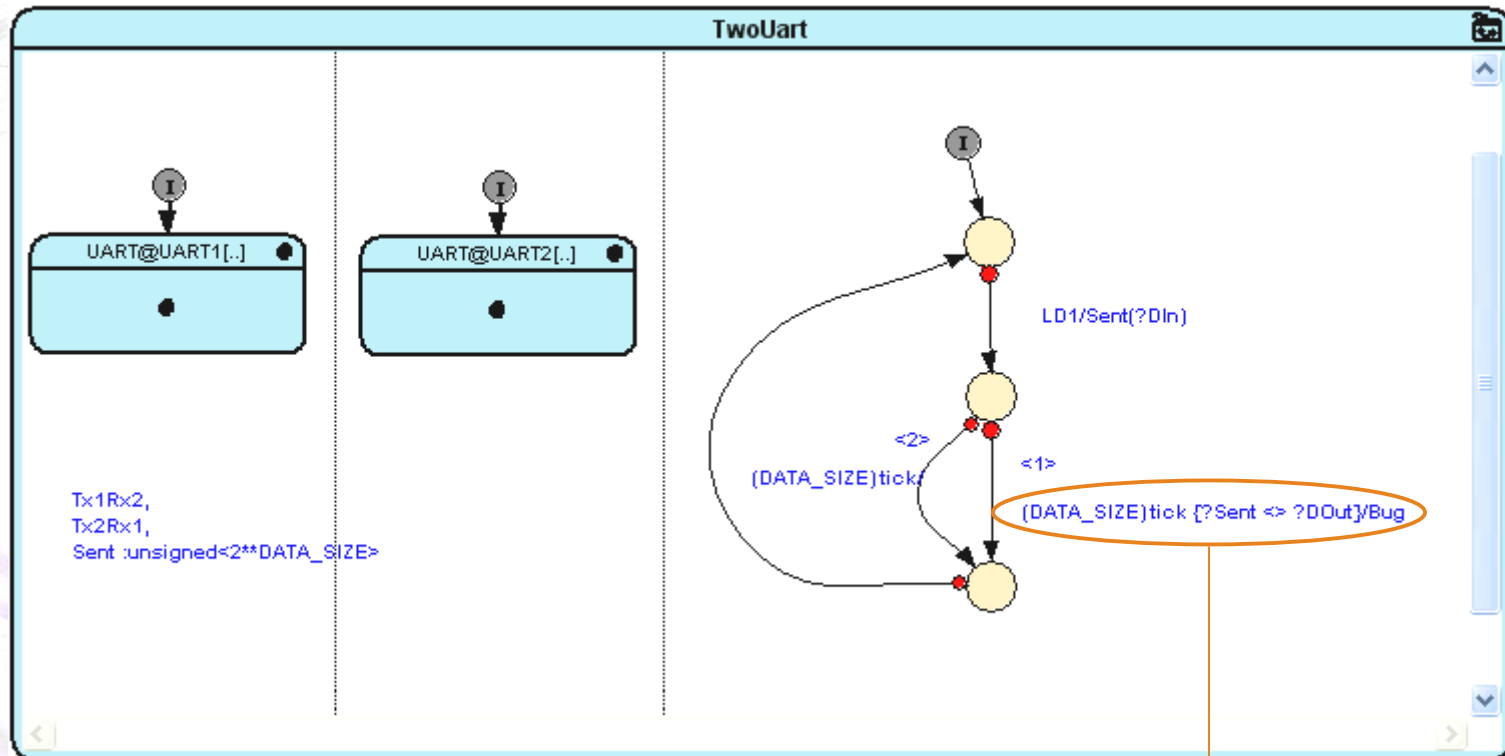
LD1/LD,  
Tx2Rx1/Rx,  
Tx1Rx2/Tx,  
TxBusy1/TxBusy,  
RxReady1/RxReady,  
RxError1/RxError

LD2/LD,  
Tx1Rx2/Rx,  
Tx2Rx1/Tx,  
TxBusy2/TxBusy,  
RxReady2/RxReady,  
RxError2/RxError

# The data sent is always received



## Exercise 4:



**Emit a signal Bug when the  
Sent data is different from  
the Received one**

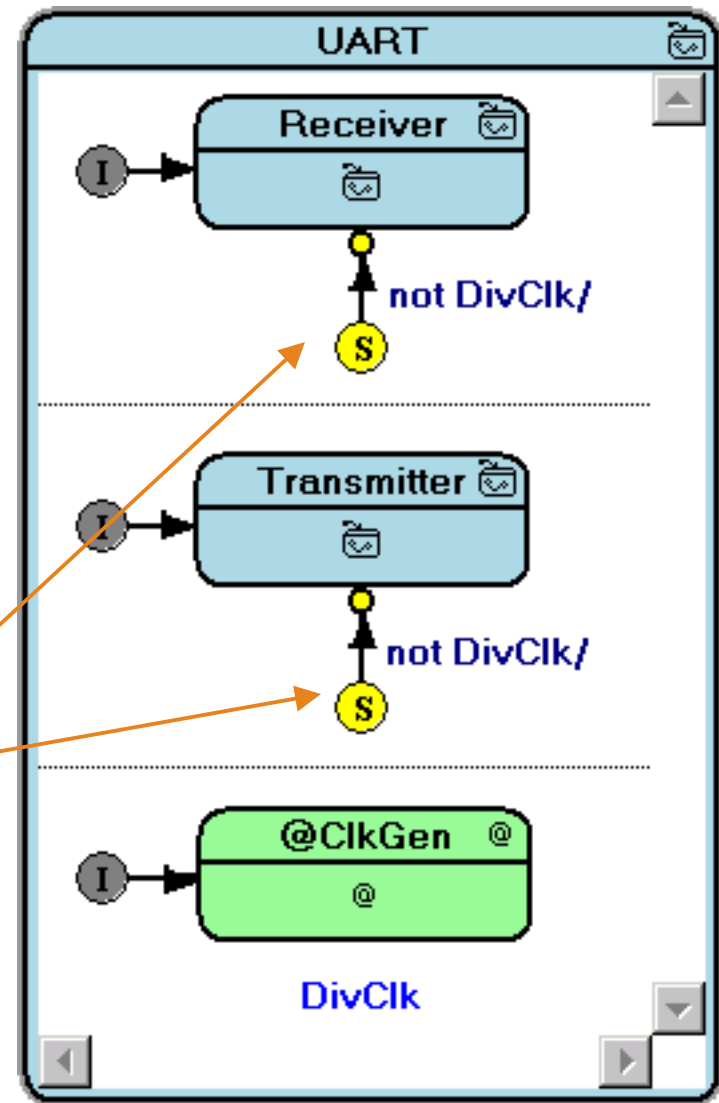
# UART with Baud Rate Selection



## Exercise 5

- ✚ Introduce the notion of multi-clock
- ✚ We generate different signals which are different clocks signals
- ✚ We suspend the emission of those generated clocks using suspend connectors.

**Behavior is suspended when the Divided Clock is not present**



# UART with Baud Rate Selection



## Exercise 5

### UART Clock Generation

```
constant DIV0: unsigned<2> = 1 ;  
constant DIV1: unsigned<3> = 2 ;  
constant DIV2: unsigned<4> = 3 ;  
constant DIV3: unsigned<7> = 6 ;  
constant DIV4: unsigned<13> = 12 ;  
constant DIV5: unsigned<25> = 24 ;  
constant DIV6: unsigned<49> = 48 ;  
constant DIV7: unsigned<97> = 96 ;
```

```
input Baud[3];  
output DivClk;
```

Divided Clock

