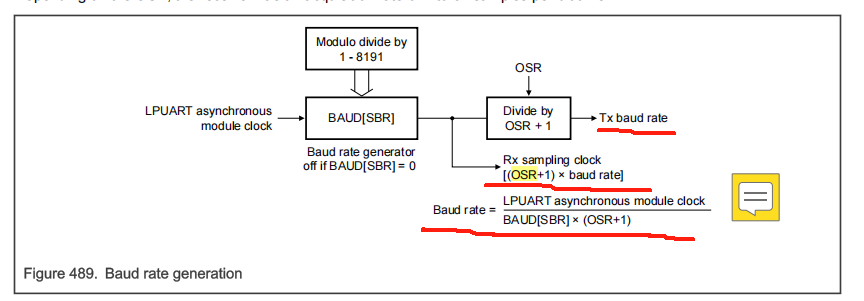
## Baud rate generator



Rx sampling clock = module\_clock / sbr= baud rate \* (osr+1)

Tx baud rate = module\_clock / (sbr \* (osr+1))

Need function in our lpuart.c：

s32k3x8\_lpuart\_update\_tolerance()

s32k3x8\_lpuart\_receive\_char()//consider tolerance to determine framing\_error and noise\_error

s32k3x8\_lpuart\_update\_clocks()//calculate Rx sampling clock、Tx baud rate

The transmitter and receiver operate independently, although they use the same baud rate generator.

## Transmitter function

Data register-->fifo-->shift-->transmitter

### 1. Writing to the DATA Register Stage

**Involved Register Fields:**

* **DATA Register:** Holds the data to be transmitted.
* **STAT[TDRE]:** Should be 1 before writing, indicating that the DATA register is empty and writable.
* **CTRL[TE]:** Must be 1 (enabled); otherwise, writing is not possible.
* **CTRL[M], CTRL[M7], BAUD[M10]:** These bits determine the valid data width for the DATA register (e.g., 8-bit, 9-bit).

**Operations:**

1. Software checks that STAT[TDRE] = 1.
2. Data is written into the DATA register.
3. The write operation clears STAT[TDRE] (if FIFO is not enabled or is full).

### 2. Data Entering the FIFO Stage

**Involved Register Fields:**

* **FIFO Control Register FIFO[TXFIFOSIZE]:** Defines FIFO depth.
* **FIFO[TXFLUSH]:** If set to 1, flushes FIFO contents.
* **FIFO[TXUFE]:** FIFO underflow error flag.
* **FIFO[TXFE]:** FIFO empty flag.
* **FIFO[TXOF]:** FIFO overflow flag.
* **FIFO[TXEMPT]:** FIFO empty status flag.
* **FIFO[TXFULL]:** FIFO full status flag.

**Operations:**

1. Data in the DATA register is automatically moved into the FIFO.
2. If the FIFO is full, FIFO[TXOF] is set.
3. FIFO[TXEMPT] becomes 0, indicating that FIFO is not empty.
4. If the FIFO is not full, STAT[TDRE] remains 1, meaning more data can be written.

### 3. Data Transfer from FIFO to Shift Register Stage

**Involved Register Fields:**

* **STAT[TDRE]:** Reflects whether FIFO can receive more data.
* **STAT[TC]:** Set to 0, indicating an ongoing transmission.
* **FIFO[TXEMPT]:** Updates when data leaves FIFO; set to 1 if FIFO becomes empty.
* **FIFO[TXCOUNT]:** Counts remaining data in FIFO; decrements as data moves out.

**Operations:**

1. When the shift register is idle, the next data in FIFO automatically moves into it.
2. FIFO status updates (TXCOUNT decrements, etc.).
3. If FIFO becomes empty, FIFO[TXEMPT] is set to 1.

### 4. Data Transmission via Shift Register Stage

**Involved Register Fields:**

* **CTRL[M], CTRL[M7], BAUD[M10]:** Configure shift register bit width.
* **BAUD[SBNS]:** Controls the number of stop bits (1 or 2).
* **BAUD[OSR] & BAUD[SBR]:** Configure transmission clock/baud rate.
* **CTRL[TXINV]:** Controls whether transmitted data is inverted.
* **STAT[TC]:** Set to 0 during transmission.
* **STAT[TDRE]:** Reflects FIFO status; 1 if FIFO can receive more data.

**Operations:**

1. The shift register transmits data bit by bit to the TXD pin at the configured baud rate.
2. The transmission includes:
   * A start bit (low level).
   * Data bits (8-10 bits).
   * Stop bit(s) (1-2 high levels).
3. Data bit order is controlled by CTRL[MSB] (LSB-first or MSB-first).
4. If parity is enabled (CTRL[PE]), a parity bit is also transmitted.

### 5. Transmission Completion Stage

**Involved Register Fields:**

* **STAT[TC]:** Set to 1 when all data is sent and no new data is waiting.
* **STAT[TDRE]:** Set to 1, indicating that new data can be written.
* **FIFO[TXEMPT]:** Set to 1, indicating that FIFO is empty.

**Operations:**

1. The last data bit is transmitted.
2. If FIFO is empty (no pending data), STAT[TC] is set to 1.
3. The TXD pin returns to its idle state (high or low, depending on CTRL[TXINV]).
4. The system waits for new data to be written or for TE to be disabled.

### Break character length

If your embedded application goes into an endless loop or abnormal state and can't respond to normal commands, you need a way to force a reset or go into a debug state

## **Receive function**

The entire LPUART reception process can be summarized in the following key steps:

#### ****1. Initial State Monitoring****

* The receiver continuously monitors the **RXD** pin, waiting for the falling edge of a **start bit** (logic 0).
* The received signal is sampled using the configured **oversampling rate (OSR)**.

#### ****2. Start Bit Detection****

* Upon detecting a falling edge, the receiver samples three consecutive logic 0 values to confirm a valid start bit.
* If confirmed, the receiver synchronizes with the bit timing.

#### ****3. Data Bit Sampling****

* Each data bit is sampled at the **middle point** of the bit period (OSR + 2).
* The receiver reads the data bits sequentially based on the configured bit length (**7, 8, 9, or 10 bits**).
* If **dual-edge sampling (BOTHEDGE = 1)** is enabled, data is sampled on both the rising and falling edges of the clock.

#### ****4. Stop Bit Detection****

* After receiving the data bits, the receiver checks for a valid **stop bit** (logic 1).
* It verifies whether the stop bit meets the expected conditions.

#### ****5. Data Processing****

* If the **Receive Data Register (RDRF = 0)** is empty, the received data is transferred to the **receive FIFO**.
* The **RDRF flag** is set to 1, indicating that data is available for reading.
* If any sampled bit is inconsistent, the **Noise Flag (NF)** is set to 1.

#### ****6. Overflow Handling****

* If the **receive FIFO** is full (RDRF = 1) and new data is received, the **Overflow Flag (OR)** is set to 1.
* The newly received data is lost, and the FIFO content remains unchanged.

#### ****7. Interrupt Generation****

* Based on the configured interrupt conditions, the hardware may trigger:
  + **Receive Complete Interrupt**
  + **FIFO Full Interrupt**
  + **Error Interrupt**

### ****Automatic Operation****

Once the receiver is enabled (RE = 1), the hardware automatically executes the above steps. The software only needs to **check the status flags** and **read the DATA register** to retrieve the received data.