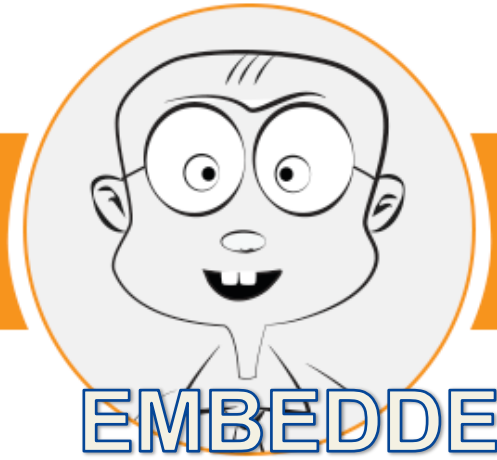


# EMBEDDED SYSTEM COURSE

## LECTURE 8: PERIPHERALS UART

# Learning Goals

- Understanding basis concepts about UART and how to transmit/receive data via UART connection.
- Introduce about RS232 protocol.
- Understanding on how to configure the KE16Z LPUART module.

# Table of contents

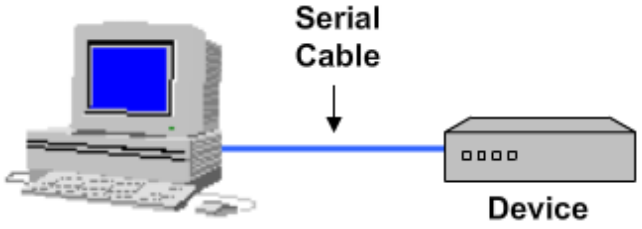
# Table of contents

# Introduction to UART

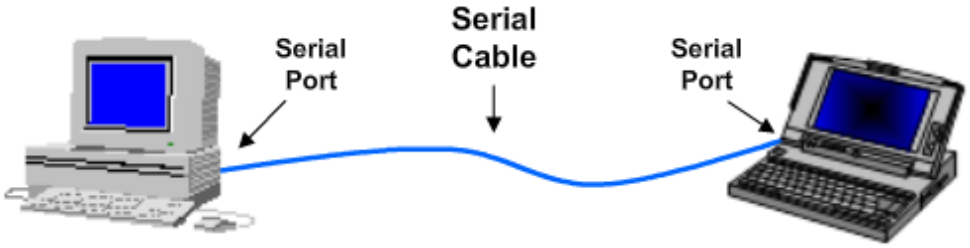
- UART: Universal Asynchronous Receiver/Transmitter
- A UART maybe used when:
  - High speed is not required
  - An inexpensive communication link between two devices is required
- Very widely used
  - Low hardware cost.
  - Relatively simple hardware.
  - Single wire for each direction (plus ground wire)
  - Asynchronous because no clock signal is transmit.

# Table of contents

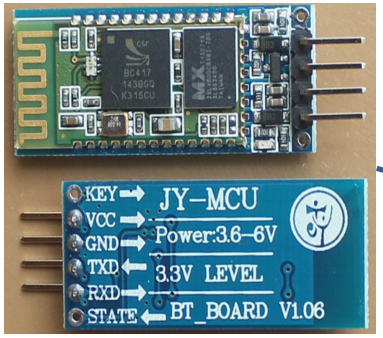
# UART applications




PC to Device




PC to PC




Bluetooth module

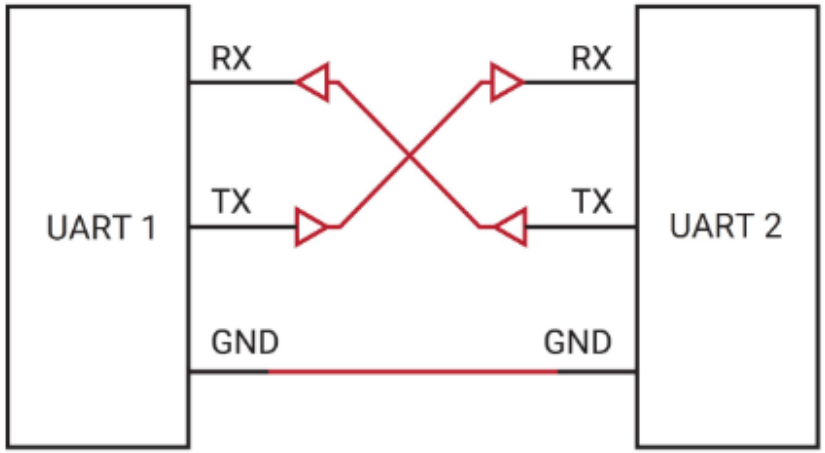
Microcontroller

GSM and GPRS Modems

# UART applications

- Use in applications that do not require fast communication.

# UART Connection

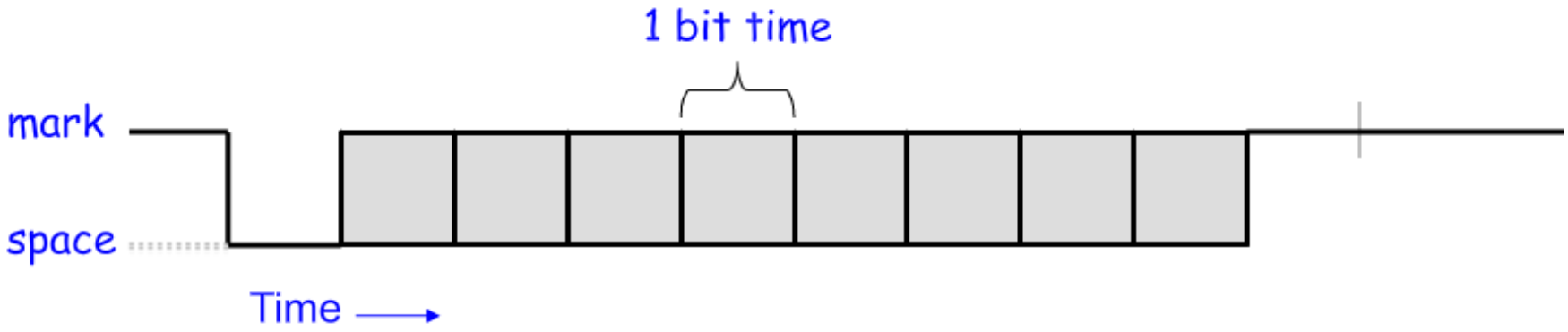


# UART functions

- **Transmitter**
  - ✓Convert from parallel to serial
  - ✓Add Start and Stop bits
  - ✓Add parity bit
- **Receiver**
  - ✓Convert from serial to parallel
  - ✓Remove Start and Stop bits
  - ✓Check and remove parity bit

# Table of contents

# Data transmission

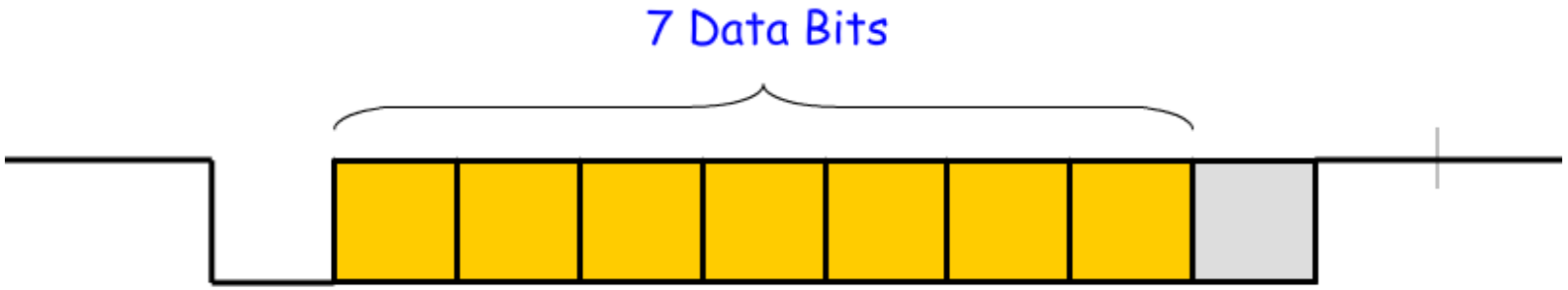



- Use a single wire for transmission
- Each bit has fixed time duration determined by the transmission rate.
  - ✓ Example: a 1200 bps (bit per second) UART will have a 1/1200 s or about 833.3 us bit duration.

# Data transmission


- The Start bit marks the beginning of a new transmission.
- When detected, the receiver synchronizes with the new data stream.

# Data transmission

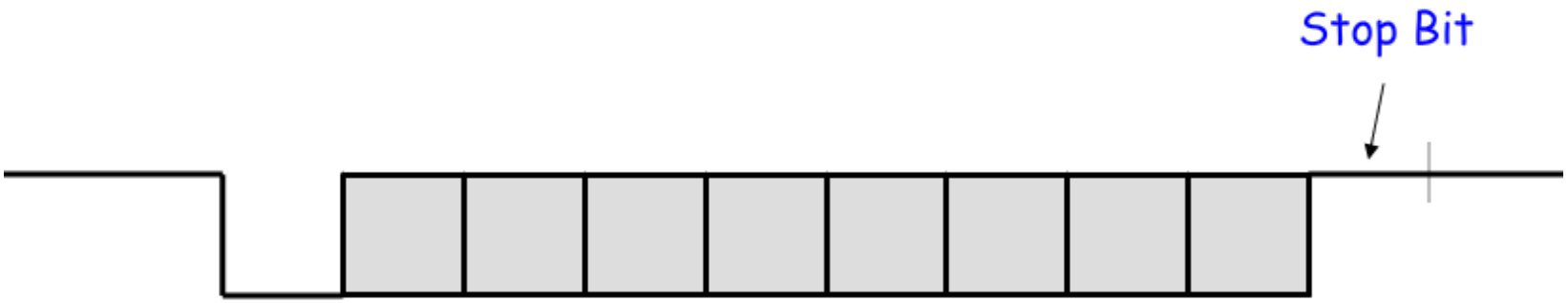



- Next follow the data bits (7 or 8).
- The least significant bit is sent first.

# Data transmission

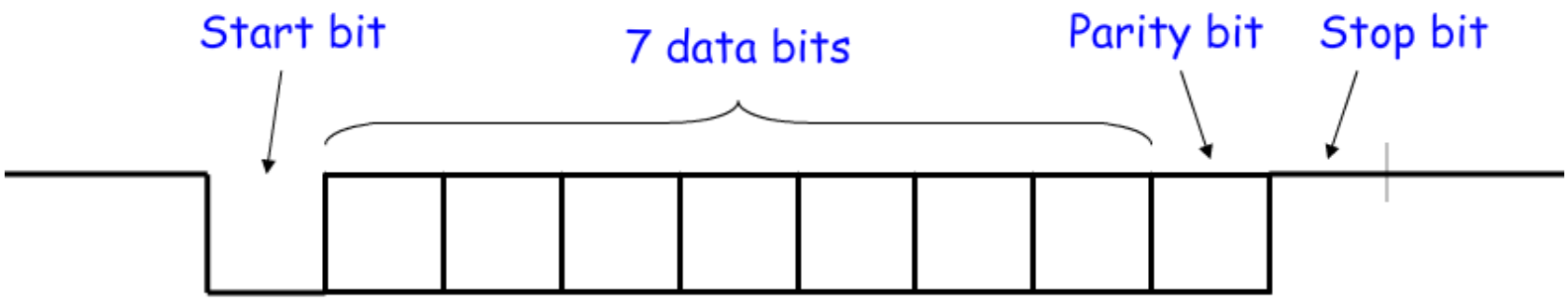

- The parity bit  is added to make the number of 1's even (even parity) or odd (odd parity).
- This is can be used by the receiver to check for transmission errors.
- Use of parity bit is optional.

# Data transmission



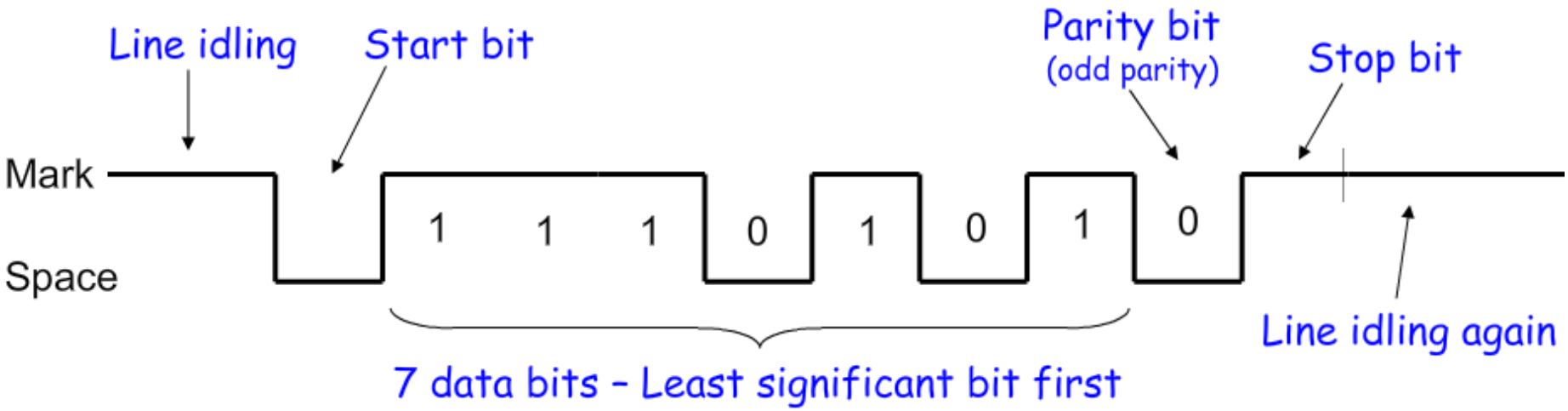

- The Stop bit marks the end of transmission.
- Receiver checks to make sure it is “1”.
- Separates one word from the start bit of the next word.

# Data transmission




- In the configuration shown, it takes 10 bits to send 7bits of data.
- Transmission efficiency is 70%.

# Data transmission

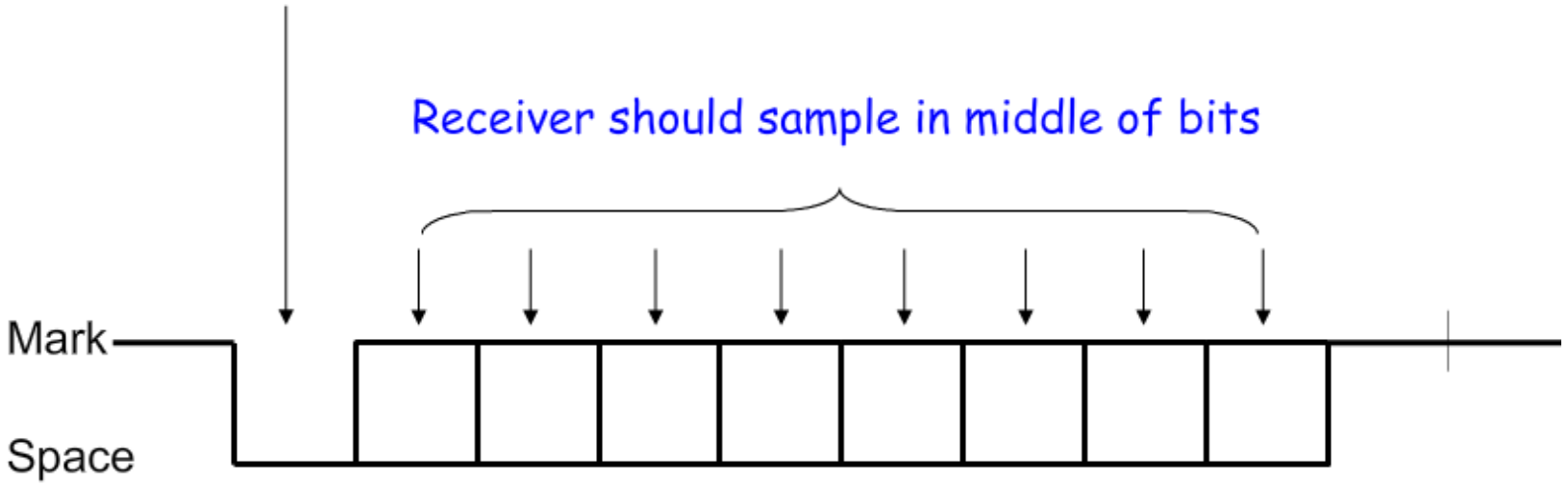



- An example of sending the ASCII letter “W”- 1010111.

# Table of contents

# Data Reception

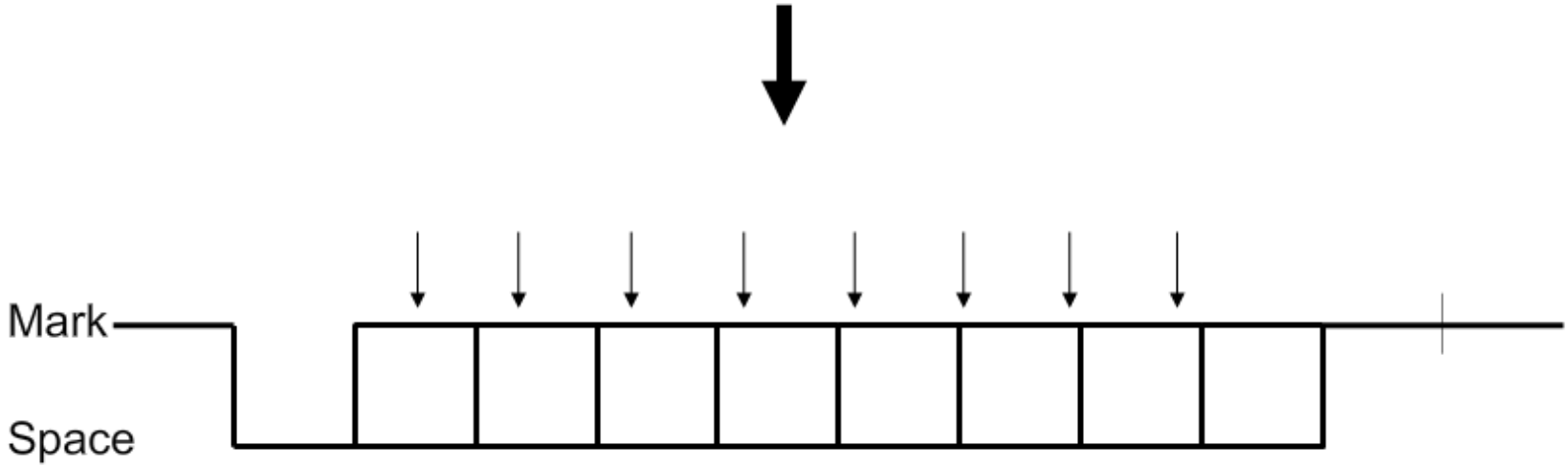
Start bit says a character is coming,
receiver resets its timers




Receiver uses a timer (counter) to time when it samples.
Transmission rate (i.e., bit duration) must be known!

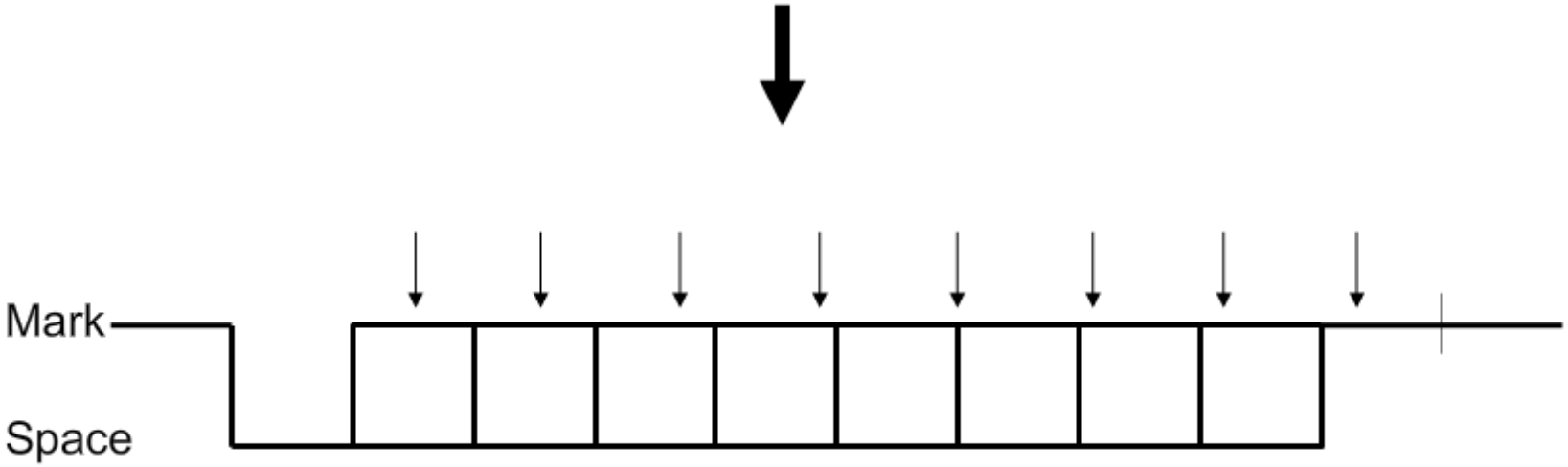
# Data Reception

If receiver samples too quickly, see what happens…



# Data Reception

If receiver samples too slowly, see what happens…




Receiver resynchronizes on every start bit.
Only has to be accurate enough to read 9 bits.

# Data Reception

- Receiver  also verifies that stop bit is “1”, if not, reports “framing error” to host system.
- New start bit can appear immediately after Stop bit.
- Receive will resynchronize for each Start bit.

# UART options

- UARTs usually have programmable options:
  - ✓ Data: 7 or 8 bits.
  - ✓ Parity: even, odd, none.
  - ✓ Stop bits: 1, 1.5, 2.
  - ✓ Baud rate: 300, 1200, 2400, 4800, 9600 , 19.2k, 38.4k, 57.6k. 115.2k…

# UART Data throughput

- Data throughput Example
  - ✓Assume 19200 baud, 8 data bits, no parity, 1 stop bit
    - 19200 baud→19.2kbps
    - 1 start bit + 8 data bits + 1 stop bit = 10 bits → It takes 10 bits to send 8 bits of data.
    - 19.2 kbps * 8/10 = 15.36 kbps.
    - How many KB (kilobytes) per second:

    =15.36 * 1KB / 8192 bits = 1.875 KB/s

# Table of contents

# RS232

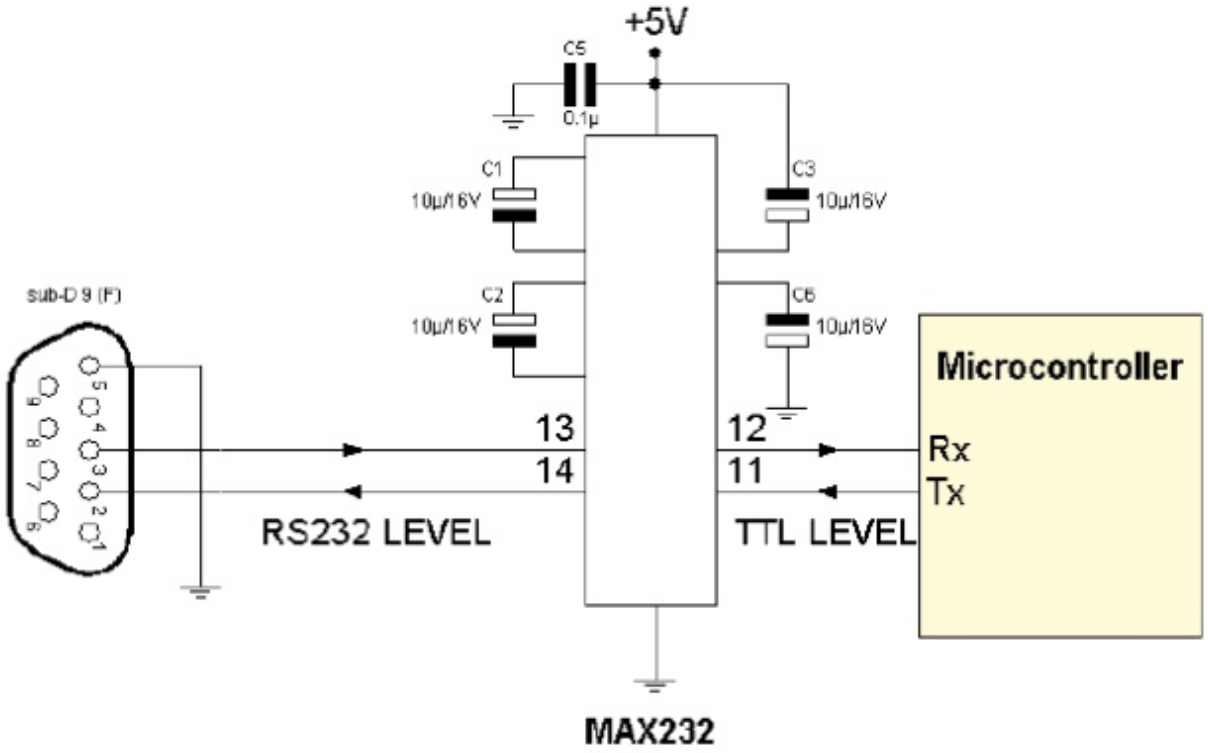
- RS232 is the most common UART standard.
  - Used by PC serial ports.
- RS232 does not use positive logic.
  - Logic 1 is any signal from -25V to -3V
  - Logic 0 is any signal from +3V to 25V
  - The range -3V to 3V is a transition region that is not assigned to a logic level.
- RS232 connecting between a *DTE* (Data Terminal Equipment) and a *DCE* (Data Circuit-terminating Equipment).

# RS232 signals

| Signal | | | Origin | | DB-25 pin |
|---|---|---|---|---|---|
| Name | Typical purpose | Abbreviation | DTE | DCE | |
| Data Terminal Ready | Indicates presence of DTE to DCE. | DTR | • | | 20 |
| Data Carrier Detect | DCE is connected to the telephone line. | DCD | | • | 8 |
| Data Set Ready | DCE is ready to receive commands or data. | DSR | | • | 6 |
| Ring Indicator | DCE has detected an incoming ring signal on the telephone line. | RI | | • | 22 |
| Request To Send | DTE requests the DCE prepare to receive data. | RTS | • | | 4 |
| Clear To Send | Indicates DCE is ready to accept data. | CTS | | • | 5 |
| Transmitted Data | Carries data from DTE to DCE. | TxD | • | | 2 |
| Received Data | Carries data from DCE to DTE. | RxD | | • | 3 |
| Common Ground | | GND | common | | 7 |
| Protective Ground | | PG | common | | 1 |

- Hardware flow control = RTS/CTS handshaking

# Connect MCU with PC over RS232



# Table of contents

# Freedom KE16Z LPUART Block Diagram

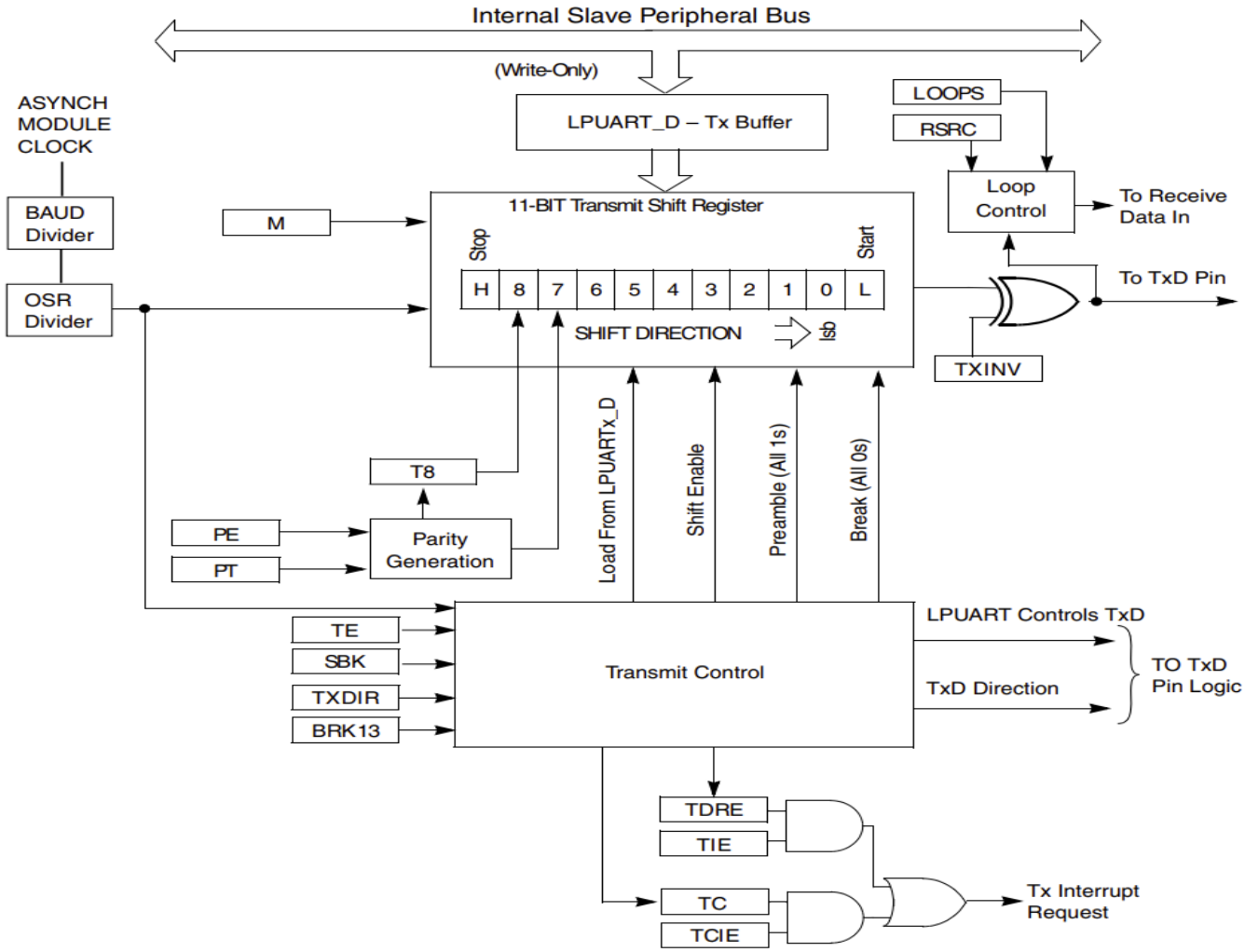



**Figure 41-1. LPUART transmitter block diagram**

# Freedom KE16Z LPUART Block Diagram

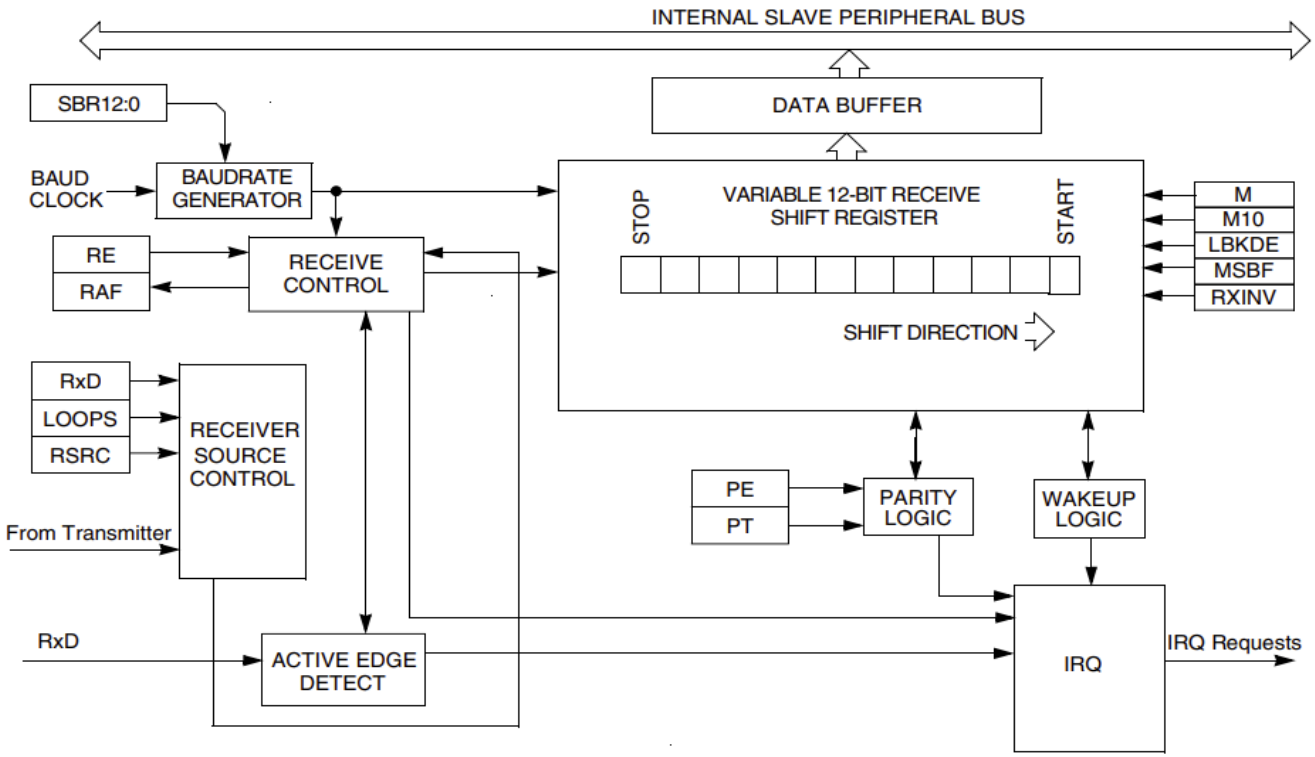



**Figure 41-2. LPUART receiver block diagram**

# Features

- Full-duplex operation
- Programmable baud rates (13-bit modulo divider and oversampling ratio from 4x to 32x).
- Programmable 7-bit, 8-bit, 9-bit or 10-bit data length.
- Programmable 1-bit or 2-bit stop bits
- Hardware parity generation and checking
- Selectable transmitter output and receiver input polarity
- Hardware flow control support for request to send (RTS) and clear to send (CTS) signals
- Independent FIFO structure for transmit and receive

# LPUART Baud Rate Register

- **BAUD**
  - ✓ M10: 10-bit Mode select
  - ✓ OSR: 5 bits Oversampling Ratio 4->32
  - ✓ SBNS: Select one or two stop bits
  - ✓ SBR: 13 bits set division ratio for baud rate generator

$$Baud\ rate = \frac{LPUART\ module\ clock}{SBR[12:0] * (OSR + 1)}$$

# LPUART Status Register

- **STAT**
  - ✓ MSBF: Configure MSB First
  - ✓ RXINV: Configure Receive Data Inversion
  - ✓ TDRE: Transmit Data Register Empty Flag
  - ✓ TC: Transmission Complete Flag
  - ✓ RDRF: Receive Data Register Full Flag
  - ✓ OR : Receiver Overrun Flag
  - ✓ NF: Noise Flag
  - ✓ FE: Framing Error Flag
  - ✓ PF: Parity Error Flag

# LPUART Control Register

- **CTRL**
  - ✓ R8T9: Receive Bit 8 / Transmit Bit 9
  - ✓ R9T8: Receive Bit 9 / Transmit Bit 8
  - ✓ TXINV: Configure Transmit Data Inversion
  - ✓ TIE: Configure Transmit Interrupt Enable
  - ✓ TCIE: Transmission Complete Interrupt Enable
  - ✓ RIE: Receiver Interrupt Enable
  - ✓ TE: Transmitter Enable
  - ✓ RE: Receiver Enable

# LPUART Control Register (continue)

- **CTRL**
  - ✓ M7: 7-Bit Mode Select
  - ✓ M: 9-Bit or 8-Bit Mode Select
  - ✓ PE: Parity Enable
  - ✓ PT: Parity Type

# LPUART Data Register

- **DATA**
  - ✓ RxTx[9:0]: 10 bits data to send or receive
    - ✓ Reads return the contents of the read-only receive data buffer.
    - ✓ Writes go to the write only transmit data buffer.

# LPUART FIFO Register

- **FIFO**:
  - ✓ TXEMPT: Transmit Buffer/FIFO Empty
  - ✓ RXEMPT: Receive Buffer/FIFO Empty
  - ✓ TXFE: Transmit FIFO Enable
  - ✓ TXFIFOSIZE: Transmit FIFO Buffer Depth
  - ✓ RXFE: Receive FIFO Enable
  - ✓ RXFIFOSIZE: Receive FIFO Buffer Depth

# LPUART Watermark Register

- **WATER**:
  - ✓ RXCOUNT: Receive Counter
  - ✓ RXWATER: set the number of datawords in the receive FIFO/buffer will generate interrupt.
  - ✓ TXCOUNT: Transmit Counter
  - ✓ TXWATER: set the number of datawords in the transmit FIFO/buffer will generate interrupt.

# Example LPUART Init

**1.** Enable the clock for the PORT associated with the LPUART pins you want to use (**PCC_PORTx**)

**2.** Enable LPUART pins (**PORTx_PCRn**)

**3.** Configure the **divider** and **enable** the clock source to be used for the LPUART (**FIRC**: SCG_FIRCDIV; **SIRC**: SCG_SIRCDIV; **SOSC**: SCG_SOSCCFG[RANGE, EREFS], SCG_SOSCDIV, SCG_SOSCCSR[SOSCEN])

**4.** Select clock source and enable clock for LPUART module (**PCC_LPUARTx**)

**5.** Configure the LPUART control registers for the desired data format

- Disable TX RX via bits **CTRL**[RE, TE]
- Number of data bits **CTRL**[M, M7] and **BAUD**[M10]
  - Parity and parity type **CTRL**[PE,PT]
  - MSB or LSB first **STAT**[MSBF]
  - Data polarity **STAT**[RXINV] and **CTRL**[TXINV]
  - Configure number of Stop bit **BAUD**[SBNS]

# Example LPUART Init(continue)

**6.** Configure the baud rate **BAUD**[SBR, OSR]

**7.** Enable the receiver and/or transmitter **CTRL**[RE, TE]

**8.** To transmit or receive data, check status of transmit buffer or receive buffer via the flags **STAT**[TDRE, RDRF] and read/write to **DATA** register.

# Reference


- http://www.powershow.com/view/af8e1-ZTk2Z/20_UART_flash_ppt_presentation
- http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter
- KE16Z Reference Manual

# Summary

- Understanding about the basic concepts regarding UART such as its applications, how the data has been transmitted as well as received.
- Understanding about RS232 standard.
- Understand on how to configure the Freedom KE16Z LPUART module.

# Question & Answer

Thanks for your attention !

# Copyright


- This course including **Lecture Presentations**, **Quiz**, **Mock Project**, **Syllabus**, **Assignments**, **Answers** are copyright by FPT Software Corporation.
- This course also uses some information from external sources and non-confidential training document from Freescale, those materials comply with the original source licenses.