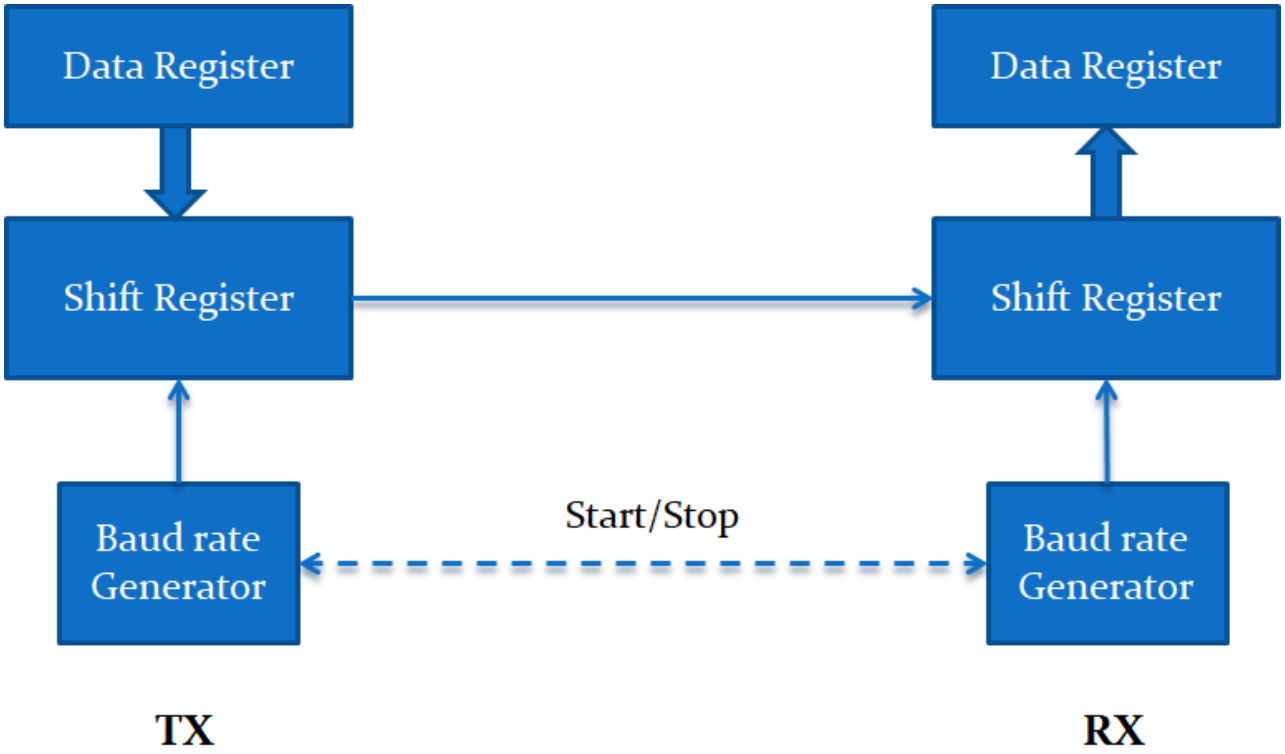
Module 7

UART interface (Part 1)

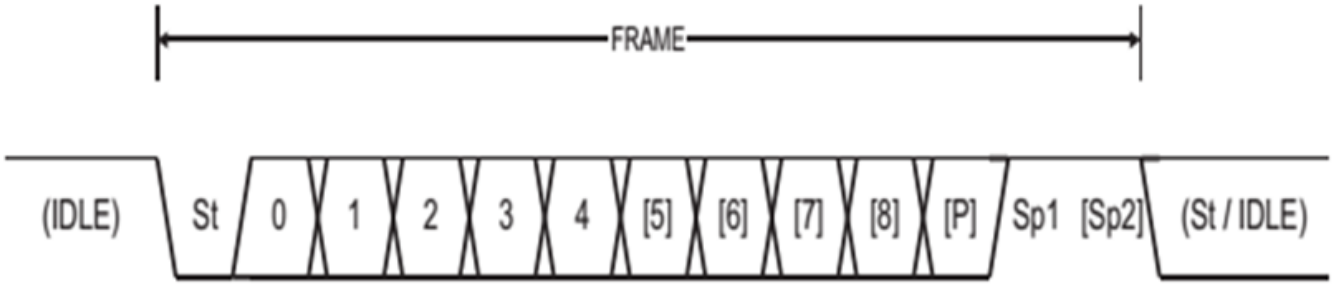
Serial data transmission

- There are two basic methods for data transmission:
  - Parallel (buses, parallel port, etc.)
  - Serial (serial port, SPI, I2C, etc.)
- U(S)ART – Universal Asynchronous (Synchronous) Receive Transmit
- Advantages:
  - Using only few lines (two, TX and RX, in the case of UART) for data transmission
  - Less prone to cross-talk
  - Longer cables (assuming that physical layer is adapted to handle environment noise)

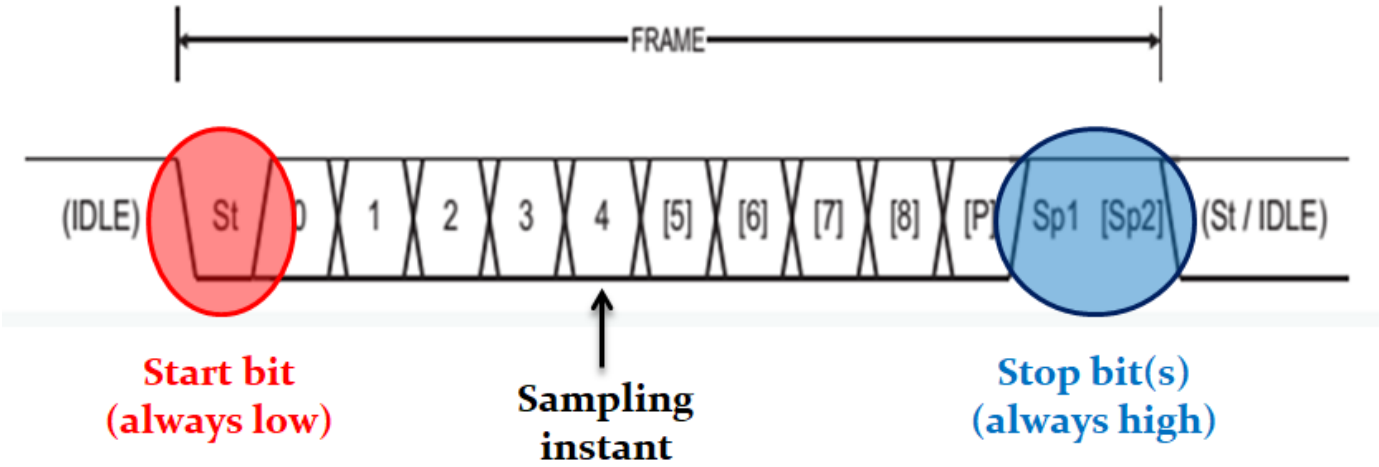
UART internals



UART working principle (1)



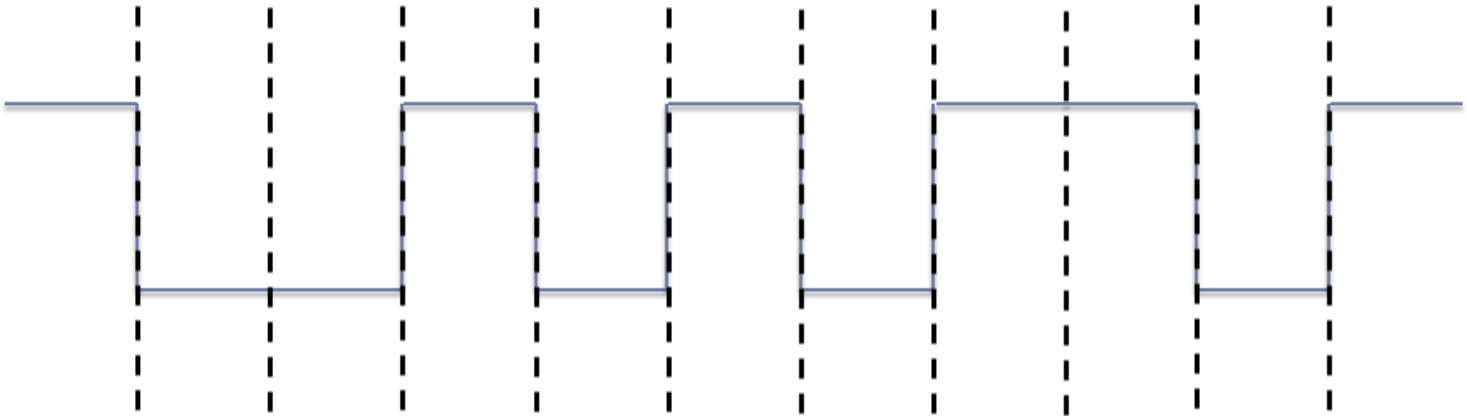
UART working principle (1)



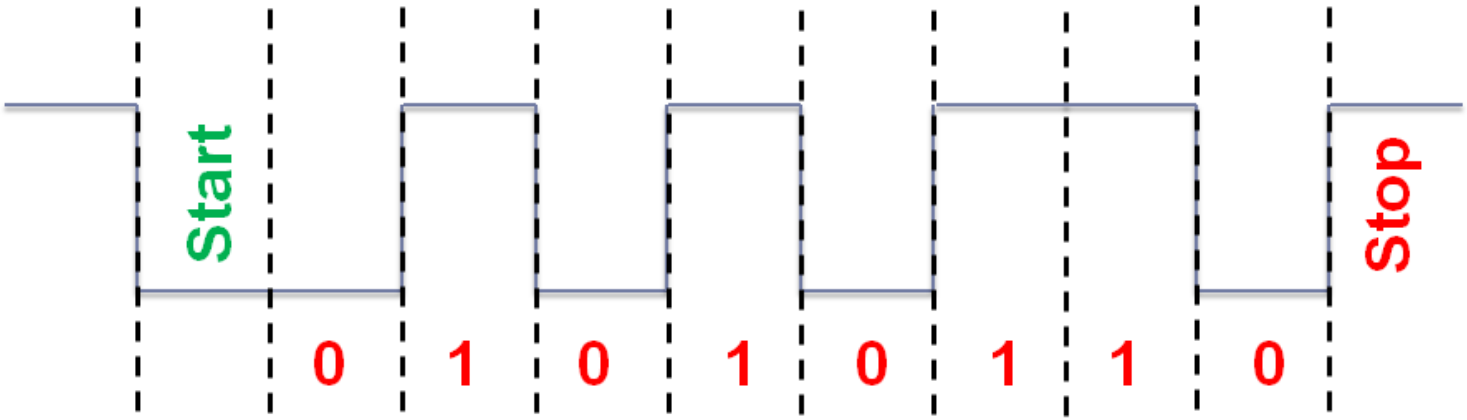
Baud rate = Bits per second

UART working principle (2)

UART working principle (2)



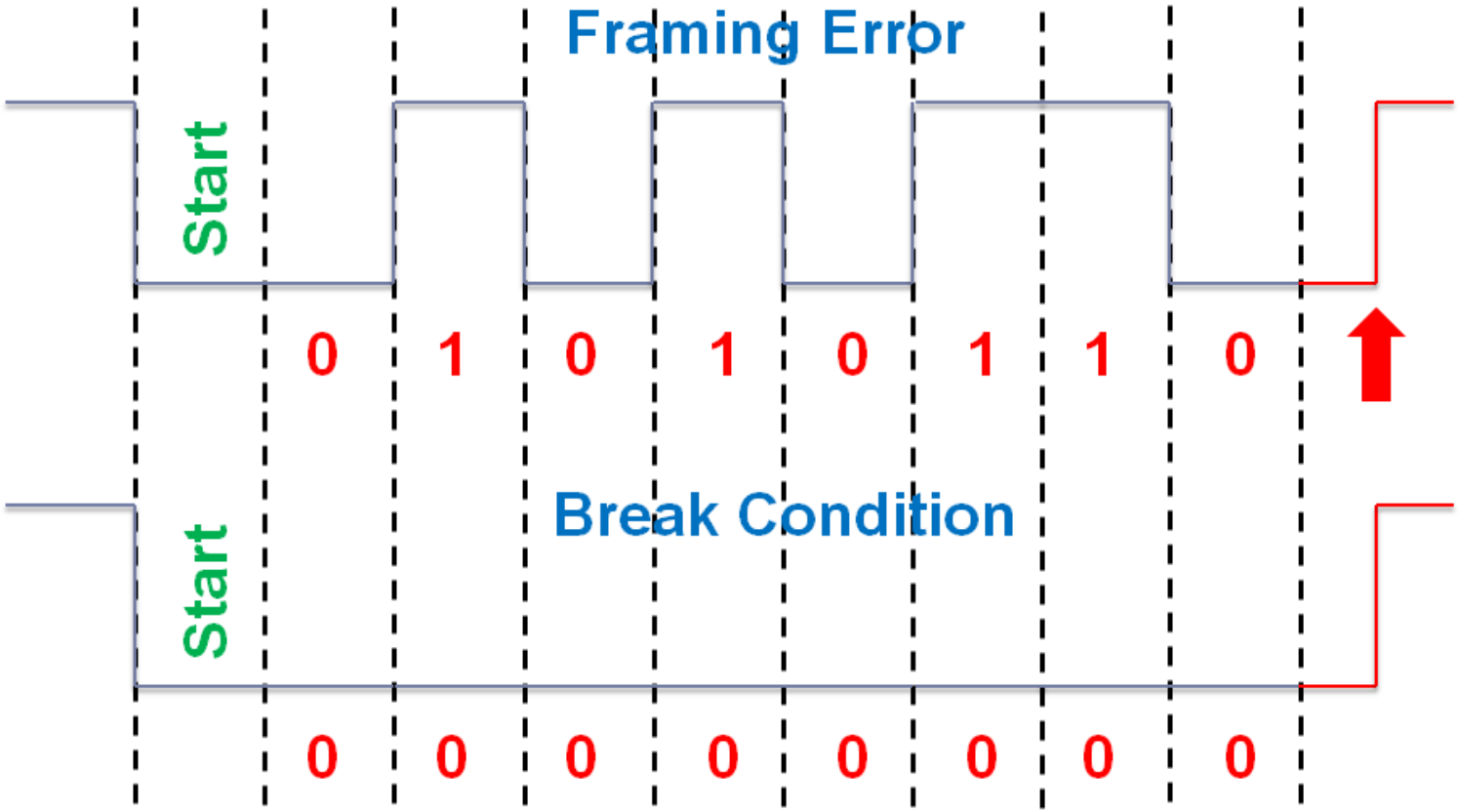
UART working principle (2)



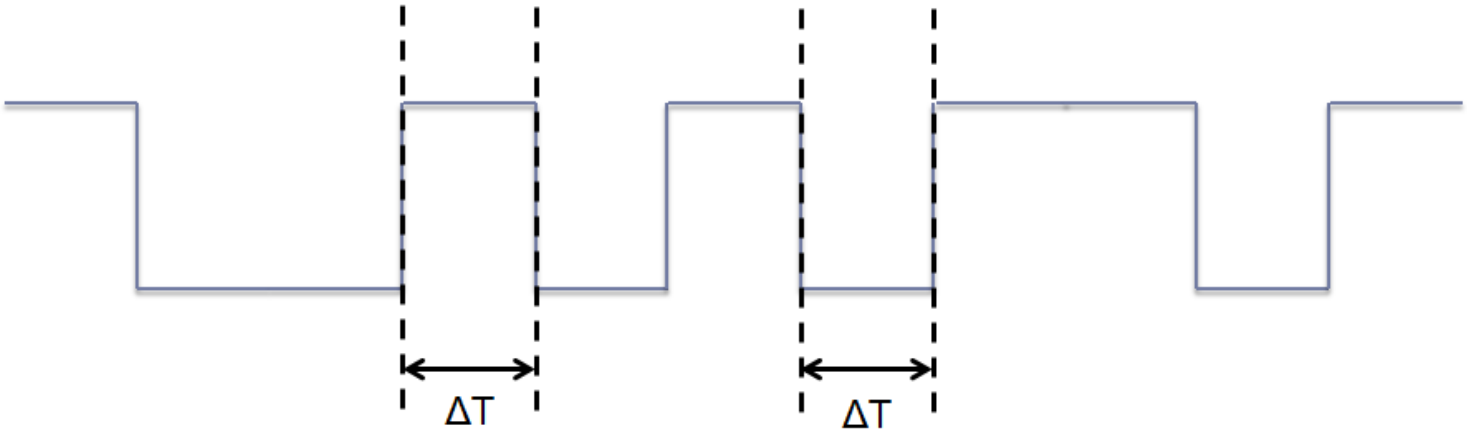
Common sources of errors in UART (1)

- Overrun error – data buffer is full when new character arrives. Can be mitigated using either hardware (RTS/CTS) or software (XON/XOFF) flow control mechanism
- Underrun error – data buffer is empty when last character has been shifted out (usually does not cause much problems)
- Framing error – UART does not see *stop* bit when expected (also applies for *break condition*)
- Parity error – there is a disagreement in one-bits parity rule (only applies when parity mode is enabled)
- Break condition – duration of low state exceeds character time (not necessarily an error): usually interpreted as receiving zero character with framing error

Common sources of errors in UART (2)

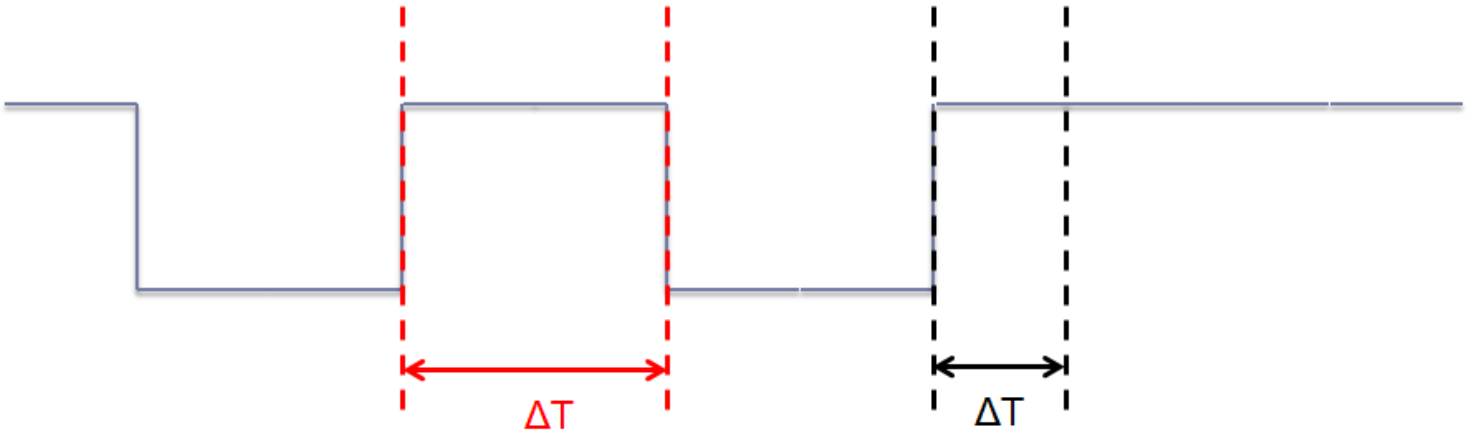


Finding UART baudrate



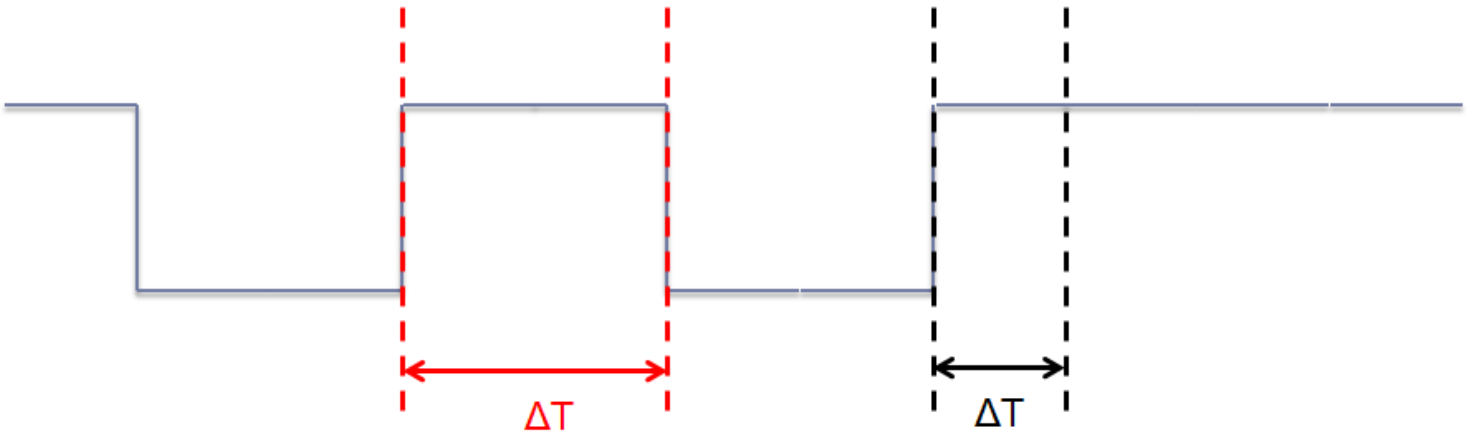
$\Delta T = 1/\text{Baudrate}$

Finding UART baudrate



0xfd, 0xe6

Finding UART baudrate



Typical baudrate values can be helpful in this case.

Manipulating UART device in Linux console

- Raspberry Pi specifics:
  - Checking UART status: `dmseg | grep tty`
  - Disable Linux console and enable UART device (`sudo raspi-config`)
  - For Raspberry Pi 3 devices, you need also to disable bluetooth (edit `/boot/config.txt` file and add `dtoverlay=pi3-disable-bt` line)
  - UART device now can be accessed via `/dev/ttyAMA0`
- Setting UART parameters (e.g, baudrate):
  ```
  stty -F /dev/ttyAMA0 9600
  ```
  For additional information, check man pages of `stty(1)` utility.
- Writing data to UART:
  ```
  echo -n -e '\x12\x34\x56' > /dev/ttyAMA0
  ```
- Reading data from UART:
  ```
  cat /dev/ttyAMA0
  ```