USART

USART

- USART stands for Universal Synchronous Asynchronous Receiver Transmitter
- Full-duplex NRZ asynchronous serial data transmission
- Offer wide ranges of baud rate

Serial communication

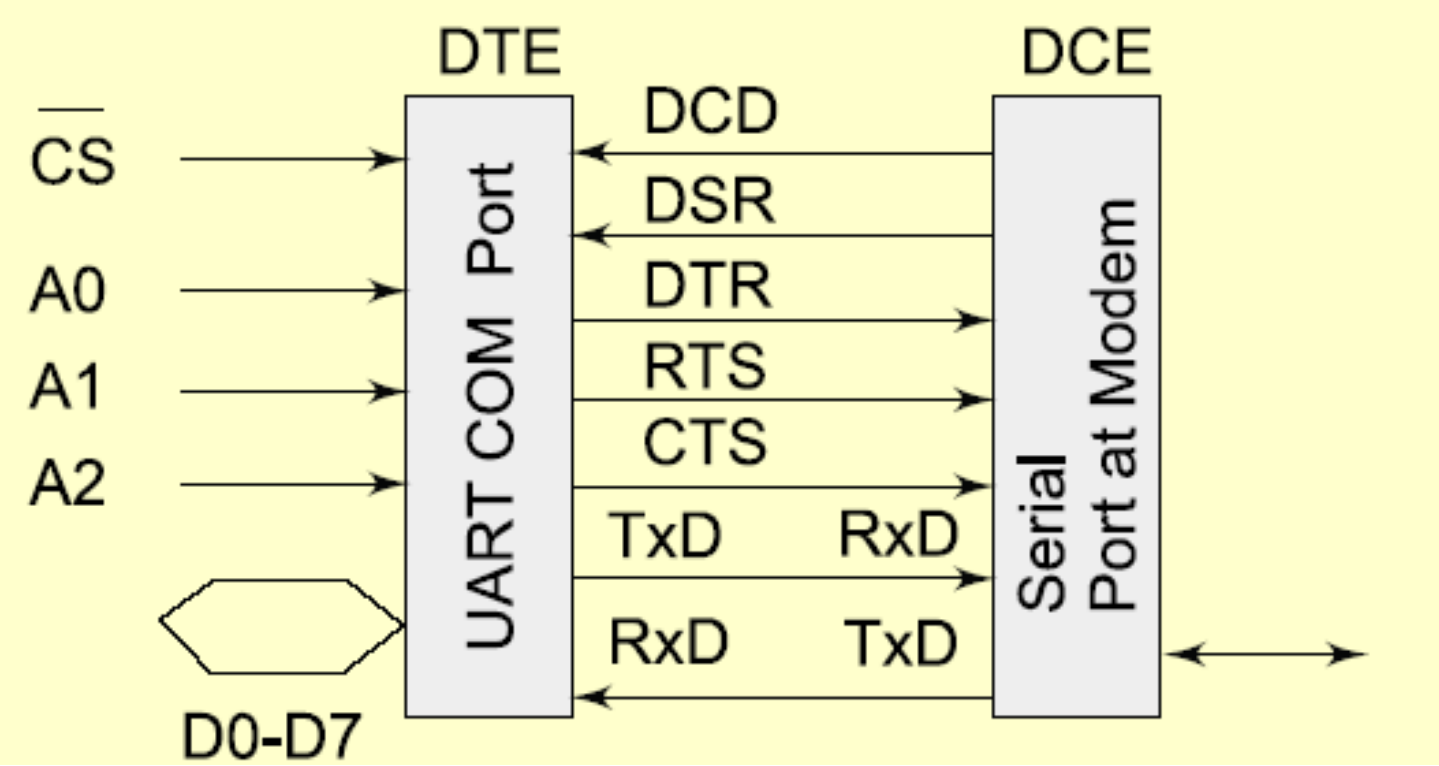
- Can support high speed communication

- Support Synchronous, Asynchronous, and Iso-synchronous

RS232C

- RS232C communication is between Data Terminal Equipment (DTE) e.g. computer and Data Communication Equipment (DCE) e.g. modem
- RS232C (Recommend Standard for Number 232C) specify communication standard such as voltage level, terminating resistances, cable length etc.

RS232C port connection



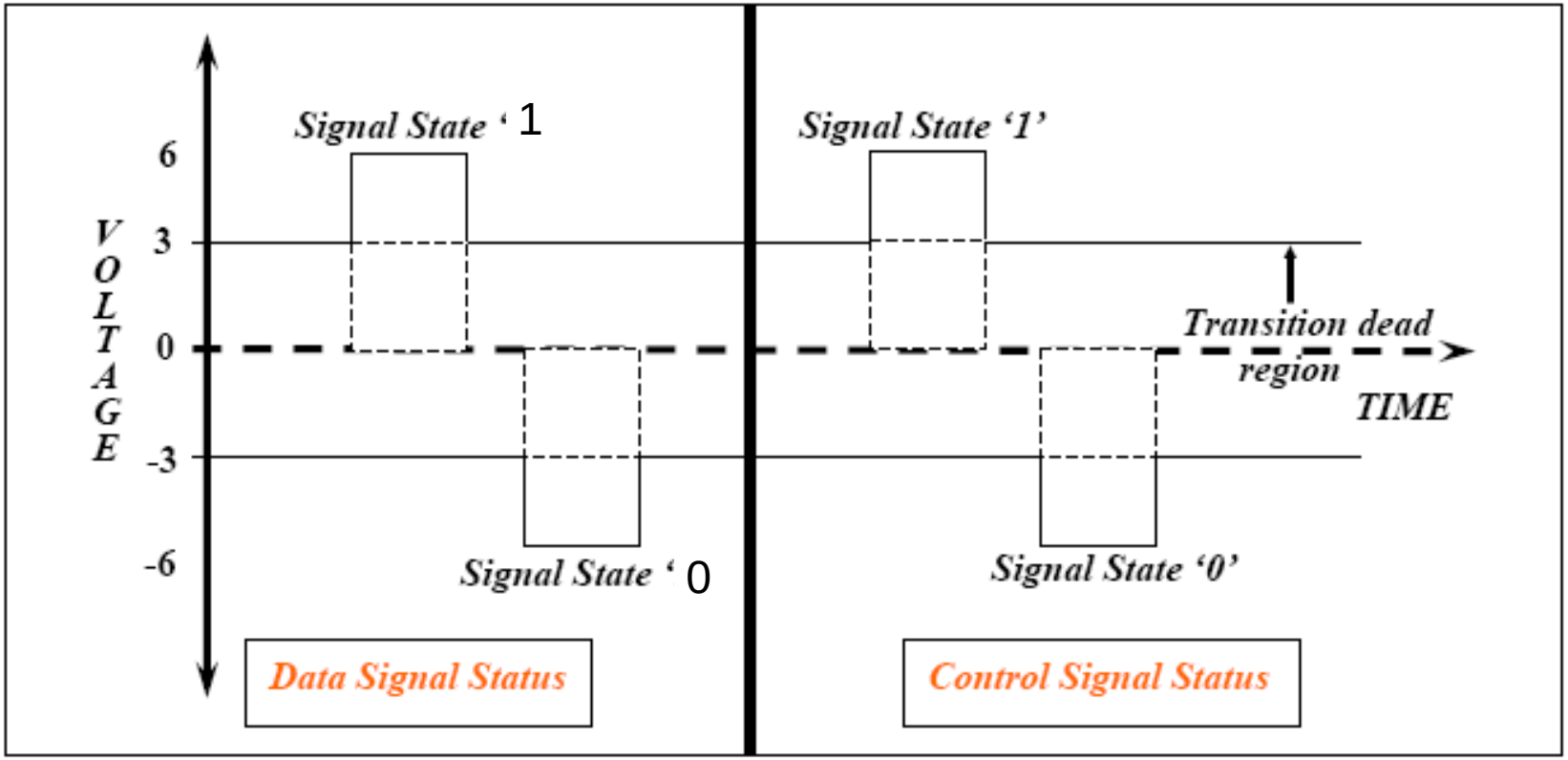
RS-232 Serial Interface

- Transmit and Receive data lines
- No clock signal exchanged – Sender and receiver need to have same baud rate
- Baud rate is the clock rate of the bits
- Normal Bits: Start, 8 Data, Stop
- Voltage levels: a 1 is **>3V** and a 0 is **<-3V**
- Special RS232 voltage level converter chips are typically used in interface

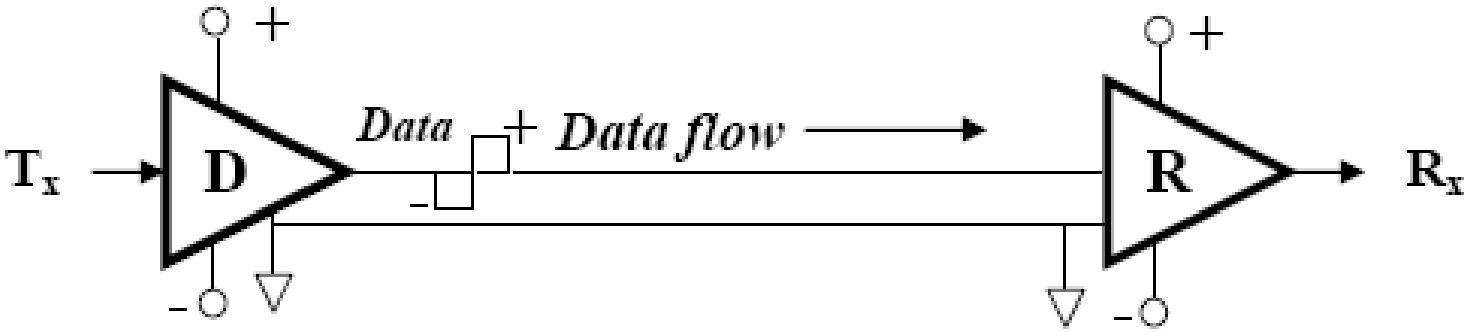
RS-232 standard

- Data rate from 20 kbps to over 1 Mbps
- Range up to 50 feet maximum
- It is robust interface up to 115,200 baud rate (pulse per second)
- Voltage as high/low as ± 15 Volt
- Single-ended means communication is over a single wire reference to ground
- There are 9 pins (DB-9) and 25 pins format (DB-25)

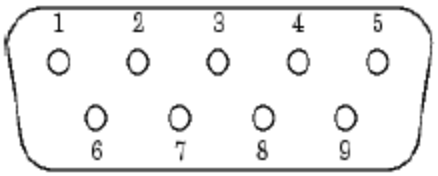
RS-232 signal



RS-232 single ended uni-direction

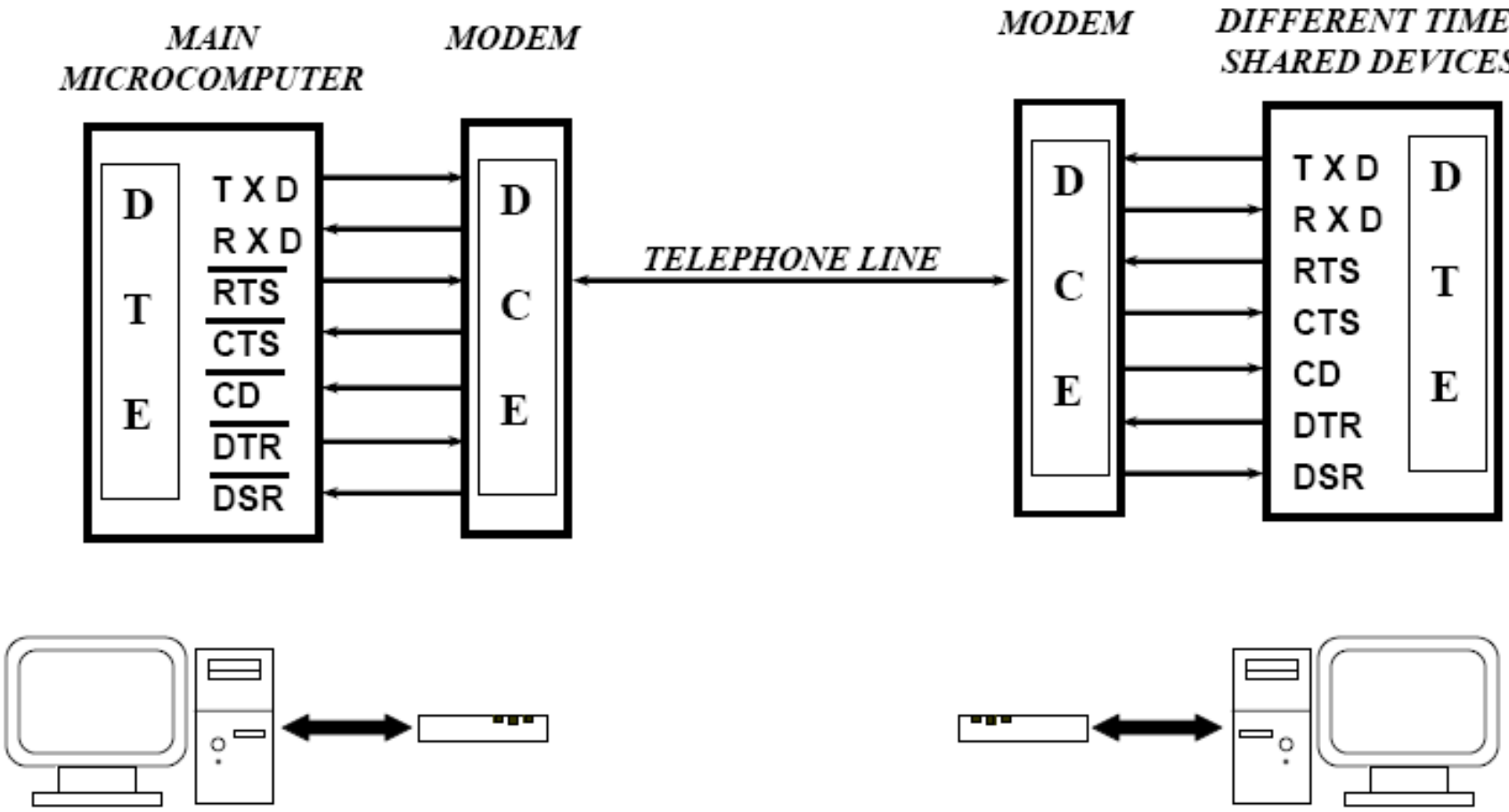


RS-232 (DB9) male connector

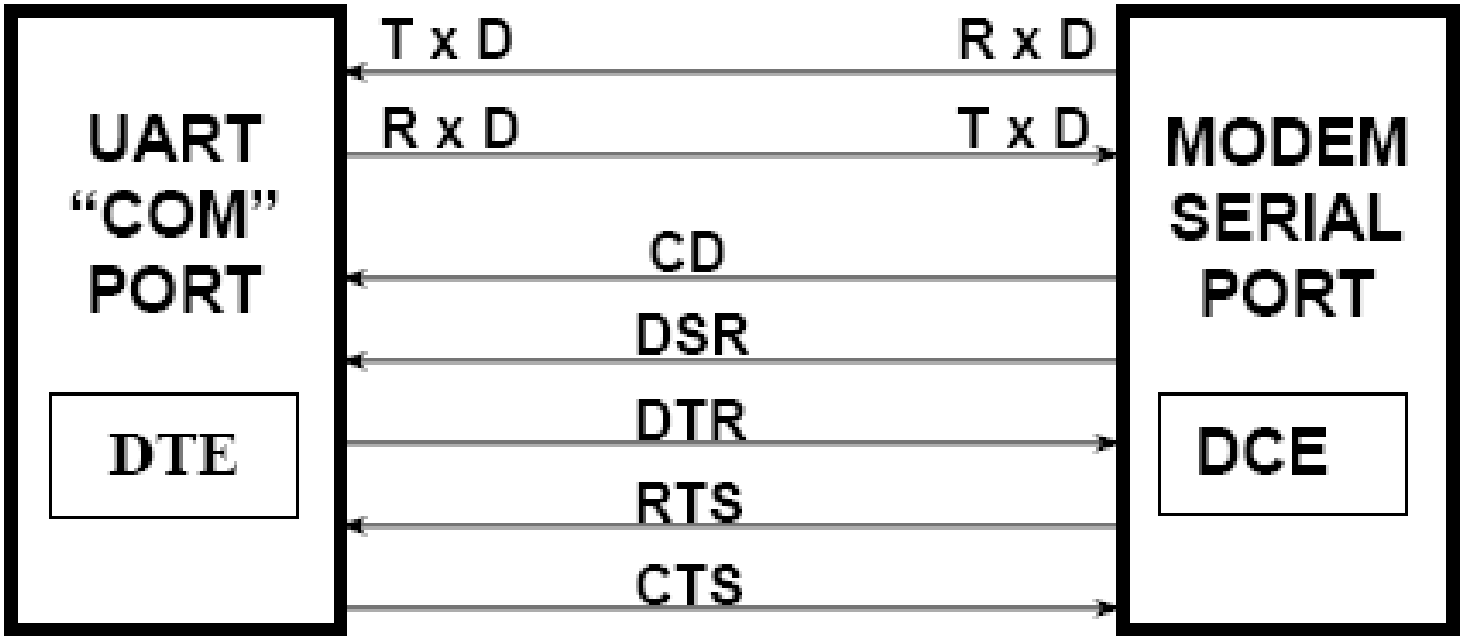


Pin 1: Carrier Detect (CD)
Pin 2: Receive Data (RD)
Pin 3: Transmit Data (TD)
Pin 4: Data Terminal Ready (DTR)
Pin 5: Ground (GND)
Pin 6: Data Set Ready (DSR)
Pin 7: Request to Send (RTS)
Pin 8: Clear to Send (CTS)
Pin 9: Ring Indicator (RI)

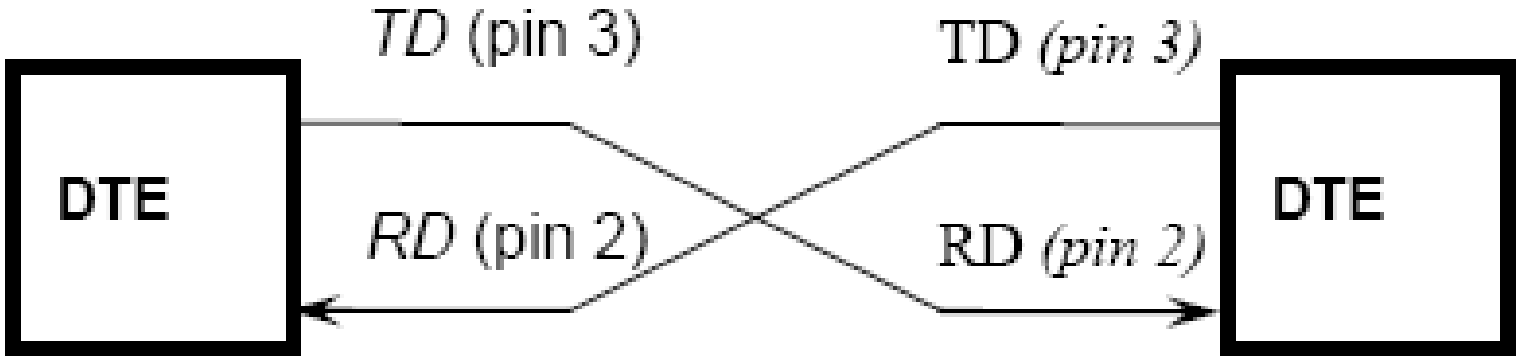
Connect computer-modem



From DTE-DCE


UART
"COM"
PORT
DTE
T x D
R x D
R x D
T x D
CD
DSR
DTR
RTS
CTS
MODEM
SERIAL
PORT
DCE

Connect two PC directly



RS232 Handshaking

Assume modem wants to send data to PC

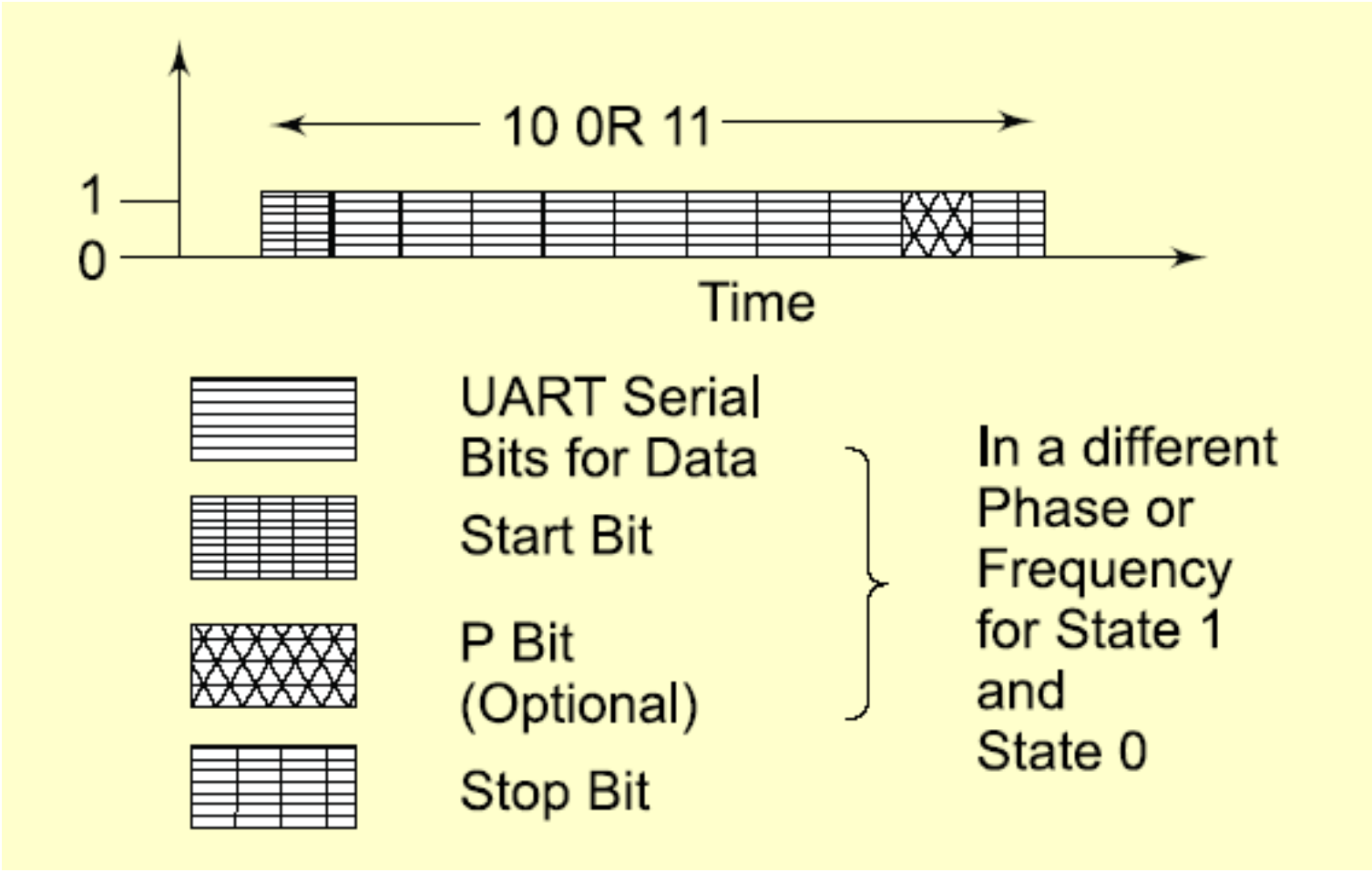
- RI indicate data available
- When modem connects, modem will send DCD signal at time t0
- Modem will send DSR signal at time t1 when it receive data to send
- PC will response with DTR at time t2
- Modem will send RTS at time t3
- PC response with CTS at time t4

RTS and CTS can also be sent again during the transaction

UART

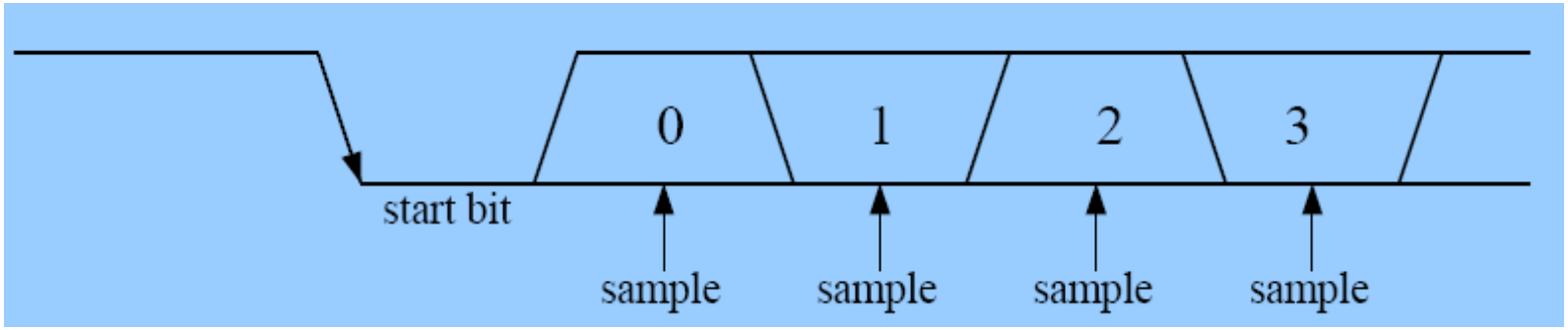
- UART is the name for the hardware used for a RS-232 Serial Interface
- UART – Universal Asynchronous Receiver Transmitter
- Early PCs had a UART chip, but this functionality is now found inside a larger chip that also contains other I/O features

UART transmission



UART initial communication

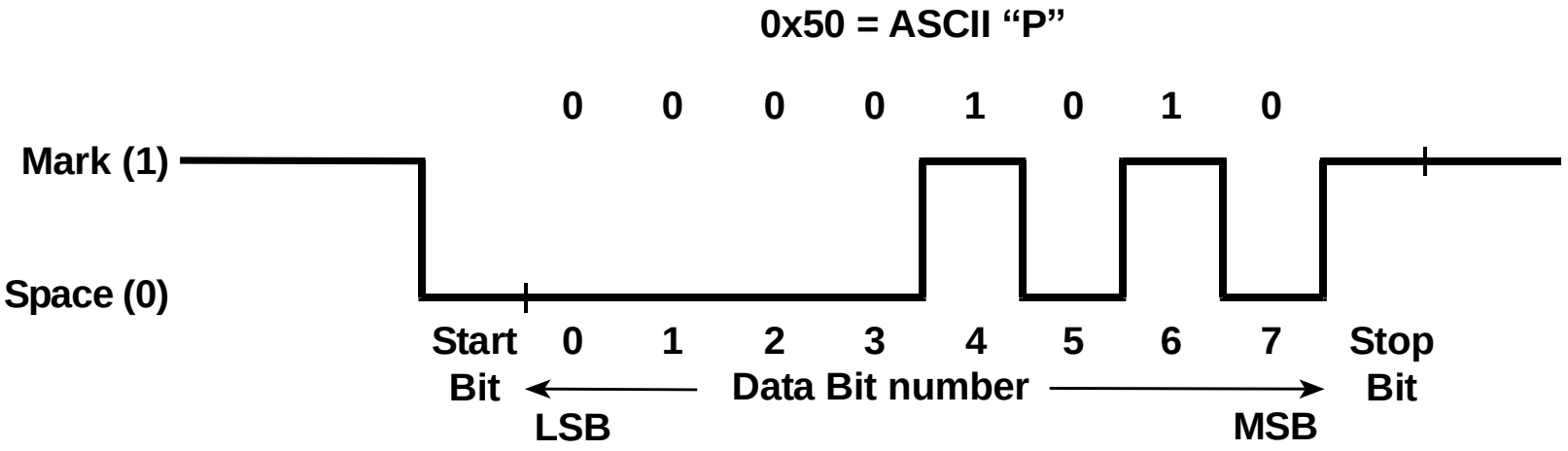
- Need to know how fast the data bits are coming
- Need to know where the starts bit begins
- Then, we know when to sample



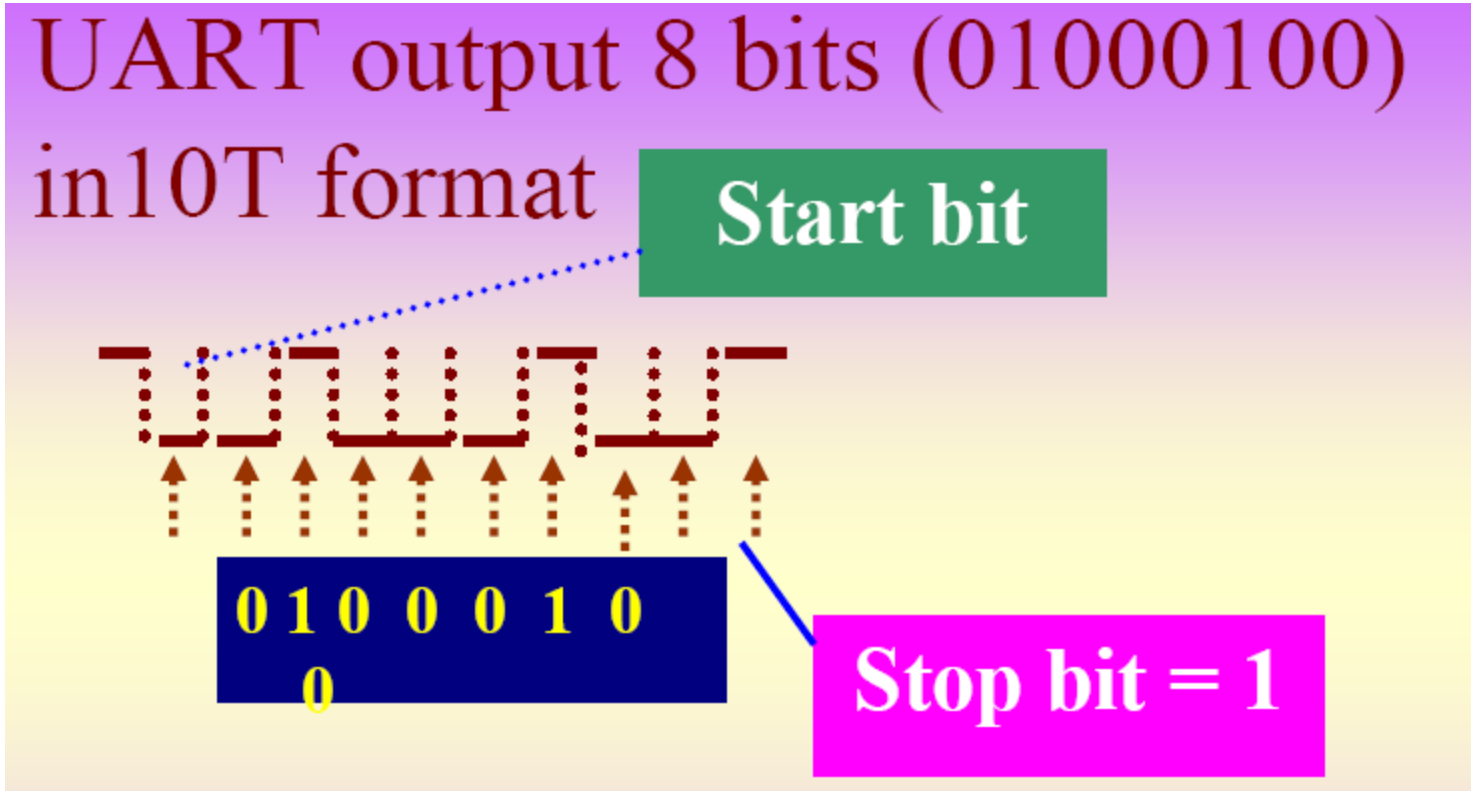
UART communication

- Non-return to zero. In the idle state, the logic state is 1.
- Start bit: transition to 0
- Data bit consists of start bit, 8 bit data, P-bit and stop bit
- Data bits can be changed to 5,6,7, and 8 bits
- The stop bit can be for a minimum of 1.5T, 2T instead of T, when T is normal interval
- P bit can be priority or for other purpose
- Stop bit: transition to 1

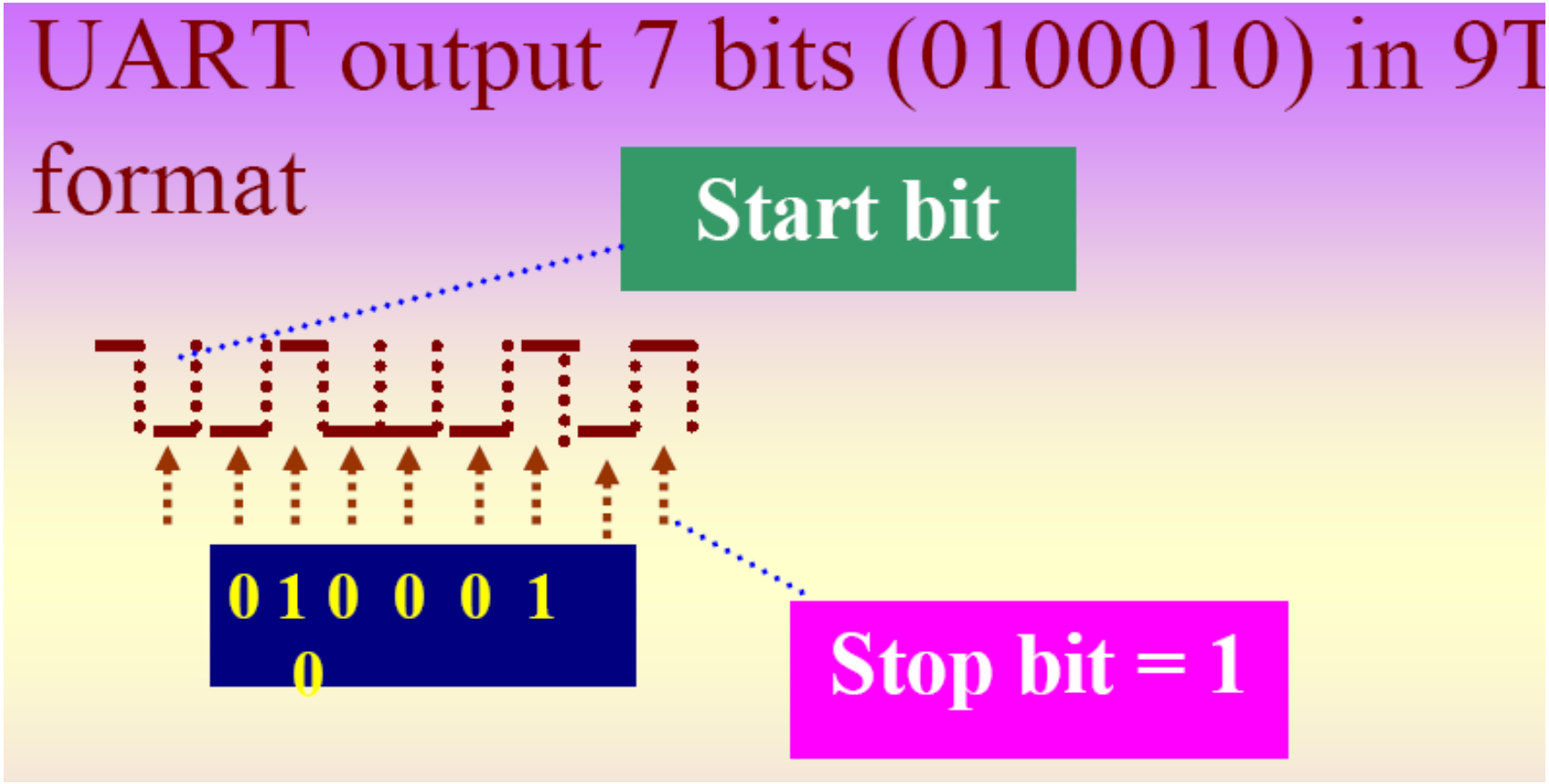
RS-232C Serial interface transmission of an 8-bit data value



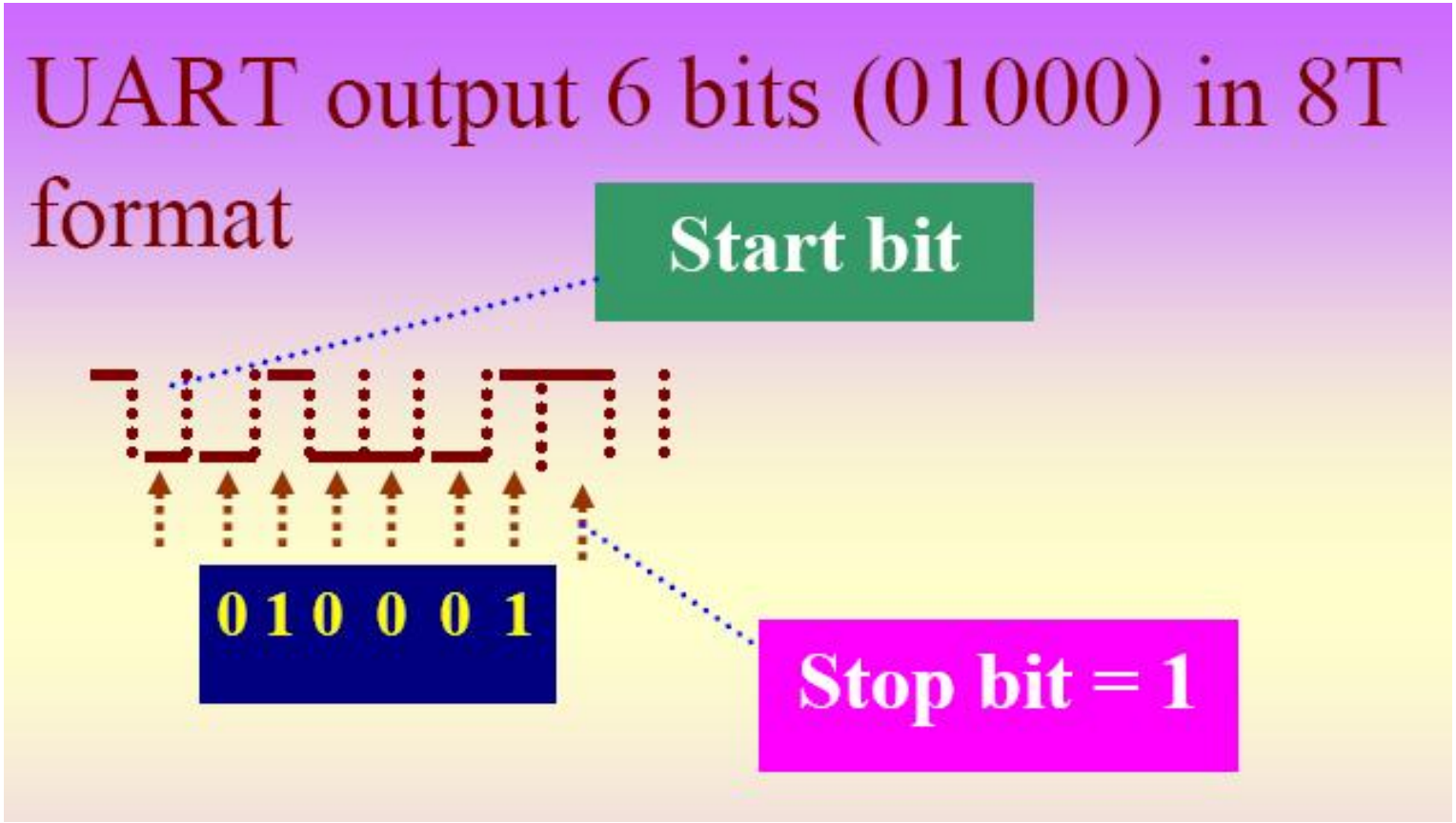
UART output 8 bits in 10T



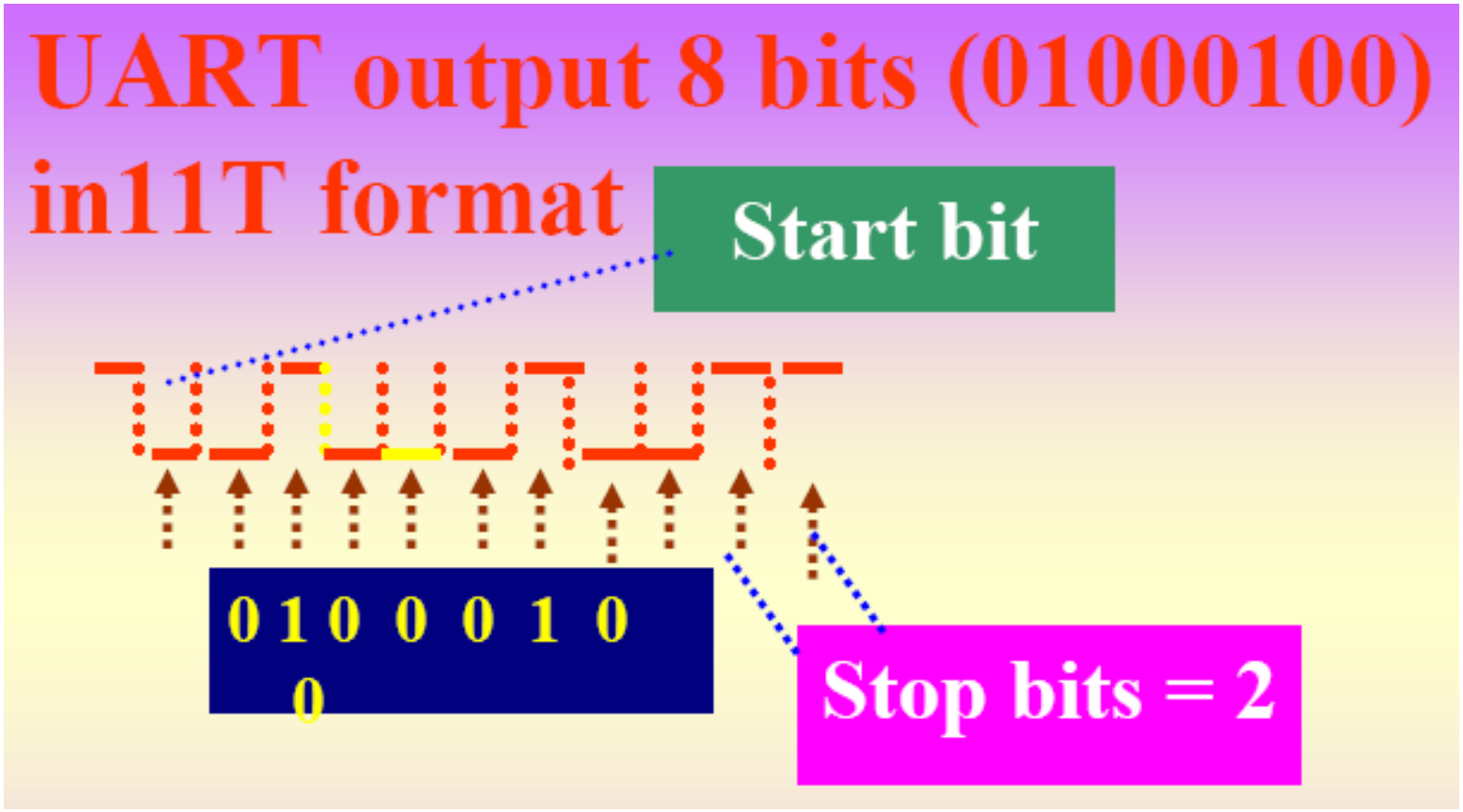
UART output 7 bits in 9T



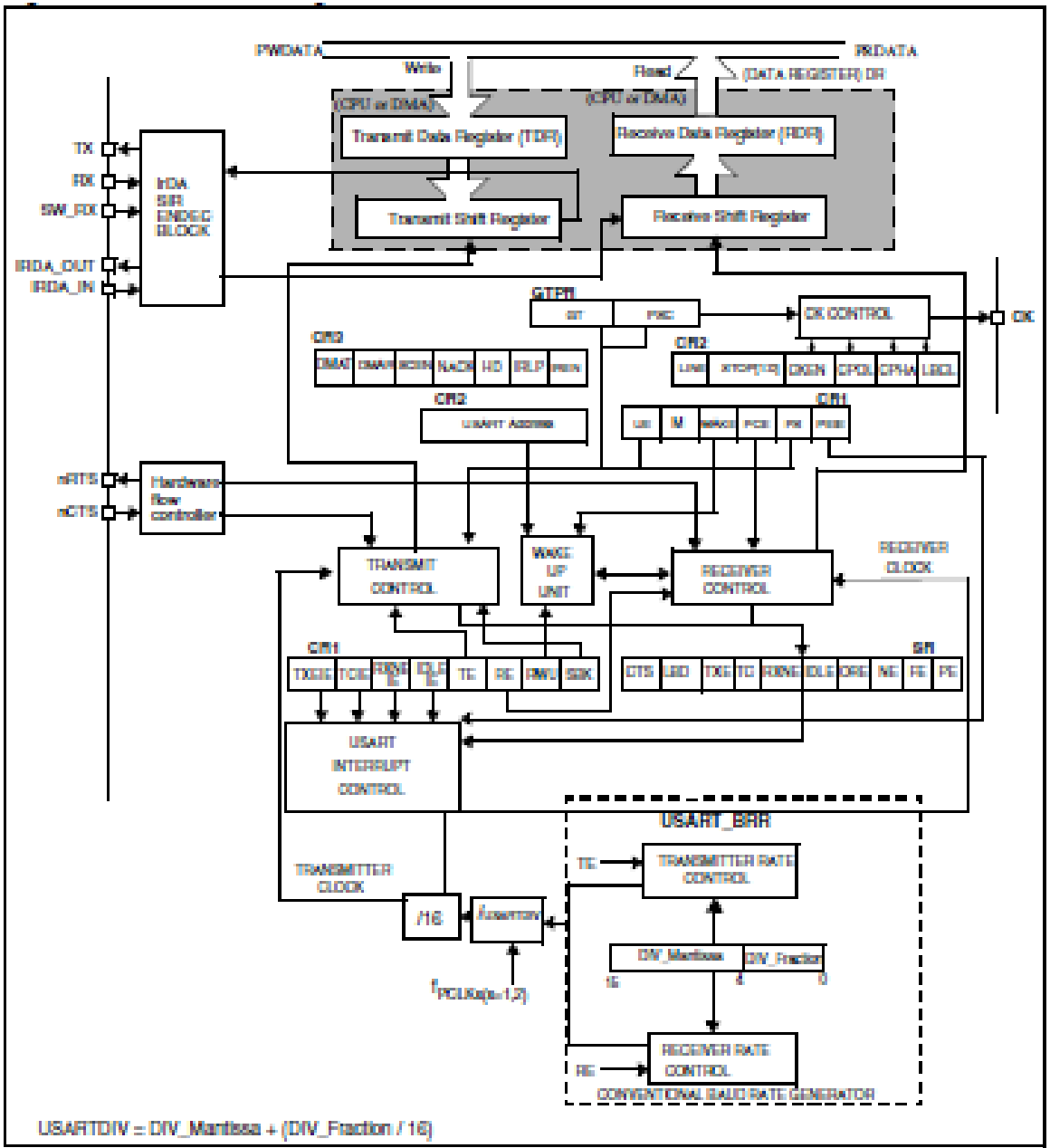
UART output 6 bits in 8T



UART output 8 bit in 11T



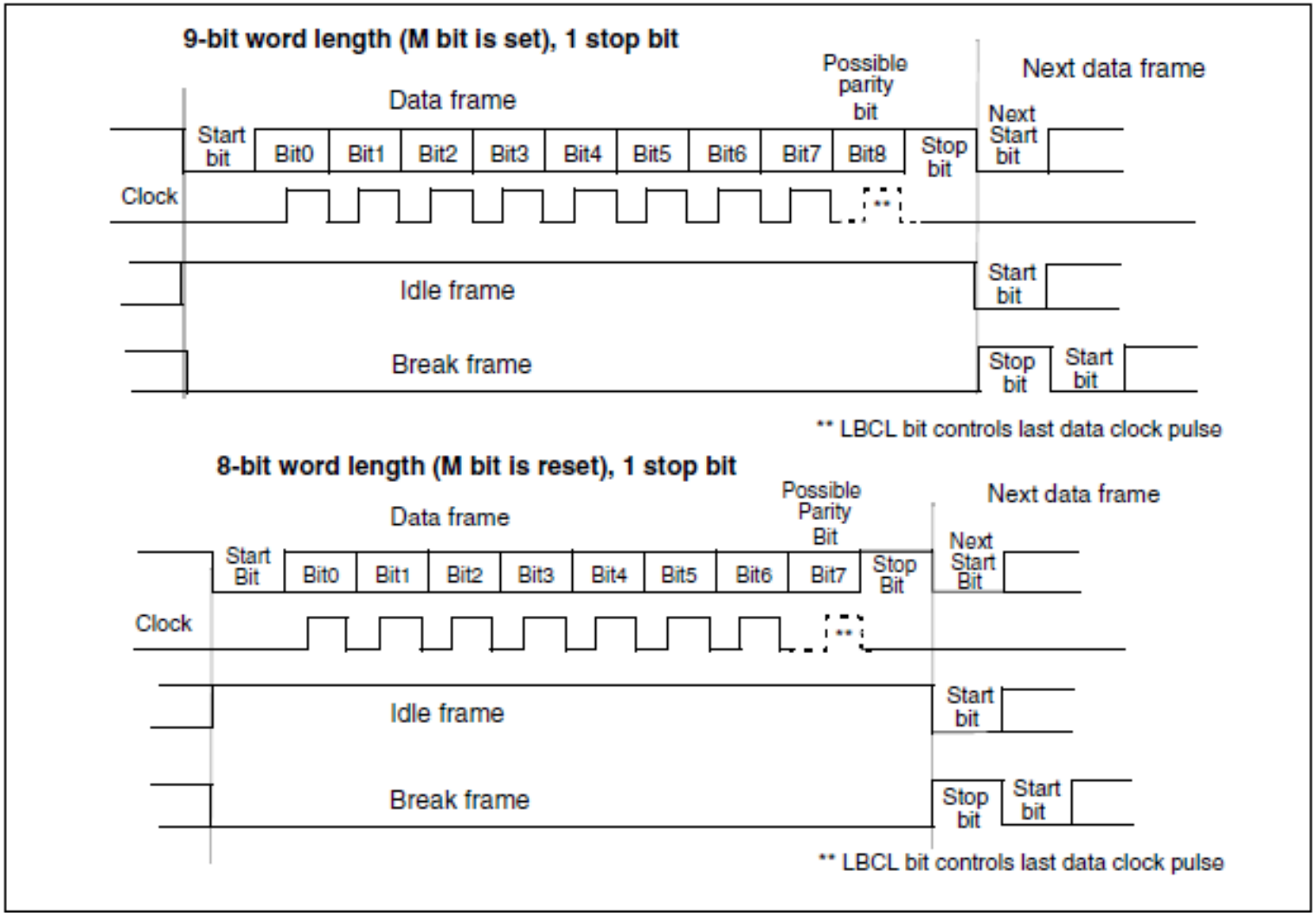
ARM USART block diagram



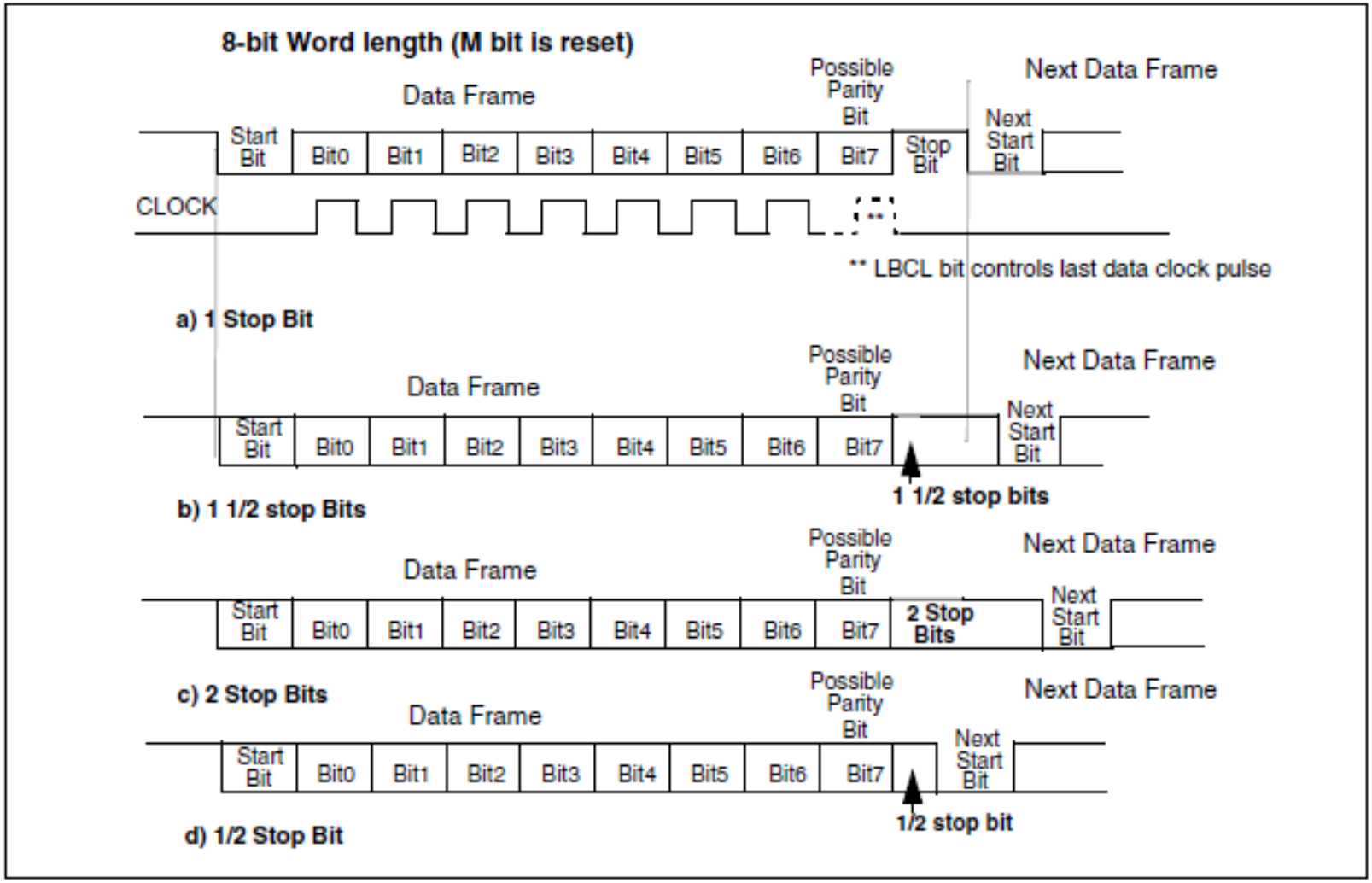
Control register in ARM

- Word length can be set by programming M bit in USART_CR1 register
- Stop bit can be set by programming USART_CR2, bit 12-13
  - 1 stop bit (default)
  - 2 stop bits used in modem
  - 0.5 stop bits used in smart card
  - 1.5 stop bits used in smart card
- Parity bit is set by USART_CR1, PCE bit

Word length setting



Stop bit programming

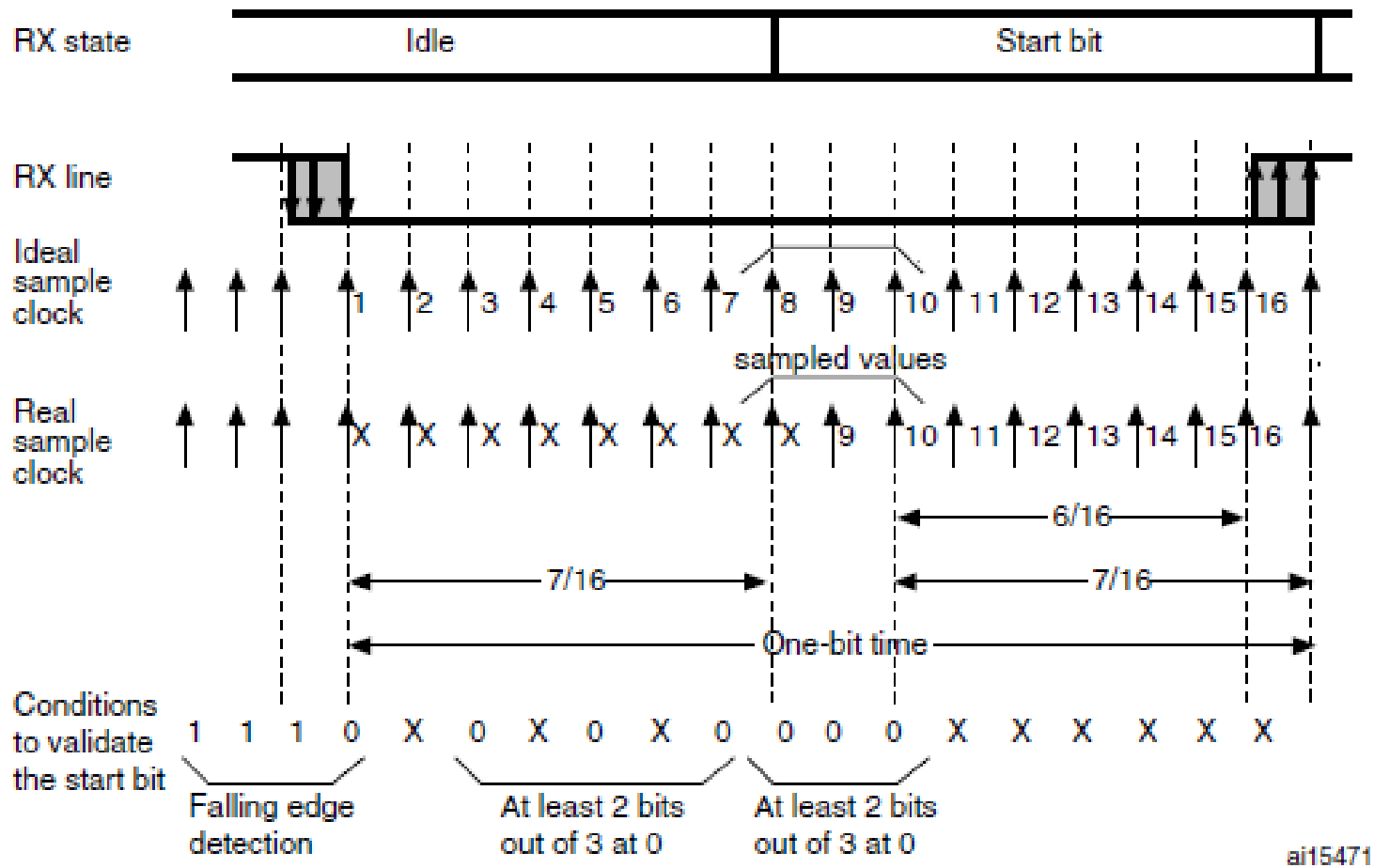


Receiver

- Start bit detection: the sequence is

1 1 1 0 X 0 X 0 X 0 0 0 0

Start bit detection



Data Transmission

1. Enable the USART by writing the UE bit in USART_CR1 to 1
2. Program the M bit in USART_CR1 to define the word length
3. Program the number of stop bit in USART_CR2
4. Select the baud rate using USART_BRR
5. Set the TE bit in USART_DR register to send an idle frame as first transmission
6. Write the data to send in USART_DR register
7. After write the last data, wait for TC bit = 1. This indicates that the transmission is complete

Data receive

1. Enable the USART by writing the UE bit in USART_CR1 to 1
2. Program the M bit in USART_CR1 to define the word length
3. Program the number of stop bit in USART_CR2
4. Select the baud rate using USART_BRR
5. Set the RE bit in USART_CR1 register. This enable the receiver to search for a start bit

When a character is received

- RXNE bit is set
- An interrupt is generated, if RXNEIE bit is set

USART programming

```
int main(void){
    RCC_Configuration(); // System Clocks Configuration

    NVIC_Configuration();       // NVIC Configuration

    USART1GPIOInit(); // Configure the GPIO for USART1

    USART1Init(); // Init USART1

    USART_Communication();
}
```

```
void USART1GPIOInit(void){
    GPIO_InitTypeDef GPIO_InitStructure;

    /* Enable GPIOA and USART1 clock */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA |
                           RCC_APB2Periph_USART1, ENABLE);

    /* Configure USART1 Tx (PA.09) as alternate function push-pull */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* Configure USART1 Rx (PA.10) as input floating */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

```
void USART1Init(void){
    USART_InitTypeDef USART_InitStructure;
    /* USART1 is configured as follow:
       - BaudRate = 115200 baud
       - Word Length = 8 Bits
       - One Stop Bit
       - No parity
       - Hardware flow control disabled (RTS and CTS signals)
       - Receive and transmit enabled*/
    USART_InitStructure.USART_BaudRate = 115200;
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART_InitStructure.USART_StopBits = USART_StopBits_1;
    USART_InitStructure.USART_Parity = USART_Parity_No;
    USART_InitStructure.USART_HardwareFlowControl =
    USART_HardwareFlowControl_None;
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;

    USART_Init(USART1, &USART_InitStructure);          /* Configure USART1 */
    USART_Cmd(USART1, ENABLE);   /* Enable the USART1 */
}
```

```
void USART_Communication(void){
    char ch;
    while(1)
    {
        SendCharUSART1(0x0D);
        SendCharUSART1(0x0A);
        SendCharUSART1('U');
        SendCharUSART1('S');
        SendCharUSART1('A');
        SendCharUSART1('R');
        SendCharUSART1('T');
        SendCharUSART1('1');
        SendCharUSART1('>');
        // Get and echo USART1
        ch = GetCharUSART1();
        while (ch != 0x0D)
        {
            SendCharUSART1(ch);
            ch = GetCharUSART1();
        }
    }
}
```

```
void SendCharUSART1(char ch){
    // Wait until TXE is set
    while(USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET)
    {
    }
    USART_SendData(USART1, ch);
    // Wait until the end of transmit
    while(USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET)
    {
    }
}

char GetCharUSART1(void){
    char ch;
    // Wait until the USART1 Receive Data Register is not empty
    while(USART_GetFlagStatus(USART1, USART_FLAG_RXNE) == RESET)
    {
    }
    ch = (USART_ReceiveData(USART1) & 0xFF);
    return ch;
}
```

Questions?