Вбудовані системи

Інтерфейс UART

Інтерфейс UART

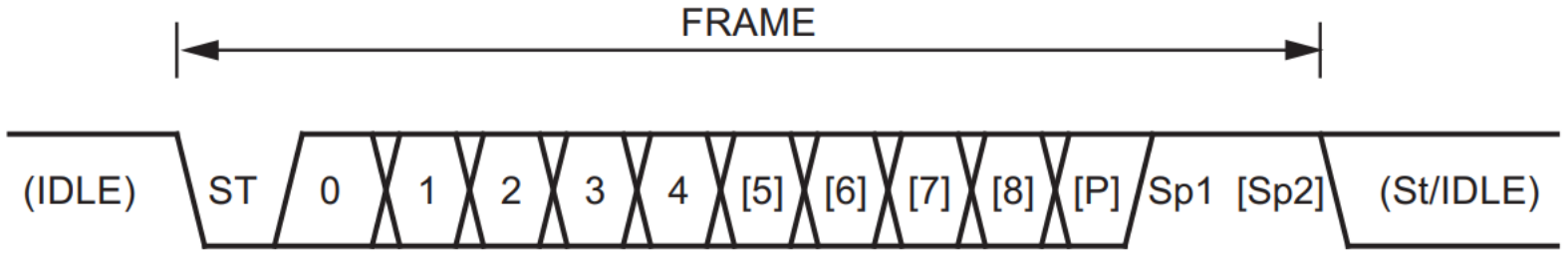
UART (англ. **U**niversal **A**synchronous **R**eceiver-**T**ransmitter) - універсальний асинхронний приймач-передавач, вузол обчислювальних пристроїв, призначений для організації зв'язку з іншими цифровими пристроями. Перетворює передані дані в послідовний вигляд так, щоб було можливо передати їх по одній цифровій лінії іншому пристрою. Метод перетворення добре стандартизований і широко застосовується в комп'ютерній техніці (особливо у вбудованих пристроях і системах на кристалі). Являє собою цифрову логічну схему, з одного боку підключену до шини контролера, а з іншого схема має два або більше виводи для під’єднання зовнішніх пристроїв.

Протокол **UART** - найстаріший і найпоширеніший на сьогоднішній день фізичний протокол передачі даних. Найбільш відомий з сімейства **UART** протокол **RS-232** (***Recommended Standard 232***, широко відомий як послідовний порт персональних комп'ютерів). Історично мав широке поширення в телекомунікаційному обладнанні. В даний час все ще використовується для підключення різного спеціального або застарілого обладнання до комп'ютерів, однак, в основному, вже витіснений інтерфейсом USB.

Інтерфейс UART (продовження)

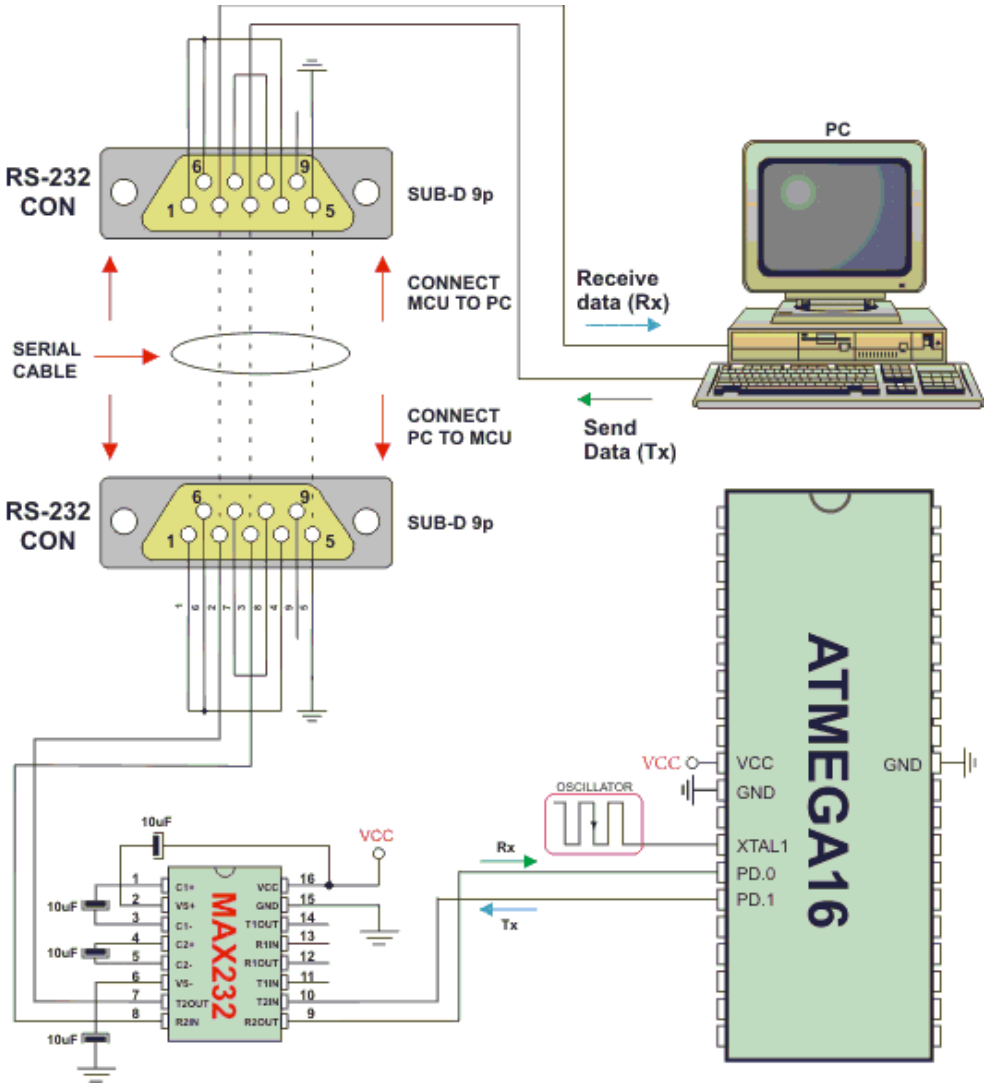
Передача даних в **UART** здійснюється по одному біту за рівні проміжки часу. Цей часовий проміжок визначається заданою швидкістю UART і для конкретного з'єднання вказується в ***бодах*** (що в даному випадку відповідає бітам в секунду). Існує загальноприйнятий ряд стандартних швидкостей: **300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600 бод.** Якщо швидкість передавача і приймача не збігатимуться, то передачі може не відбутися взагалі, або будуть прочитані помилкові дані. Дозволяється розходження швидкостей прийому/передачі до 4%.

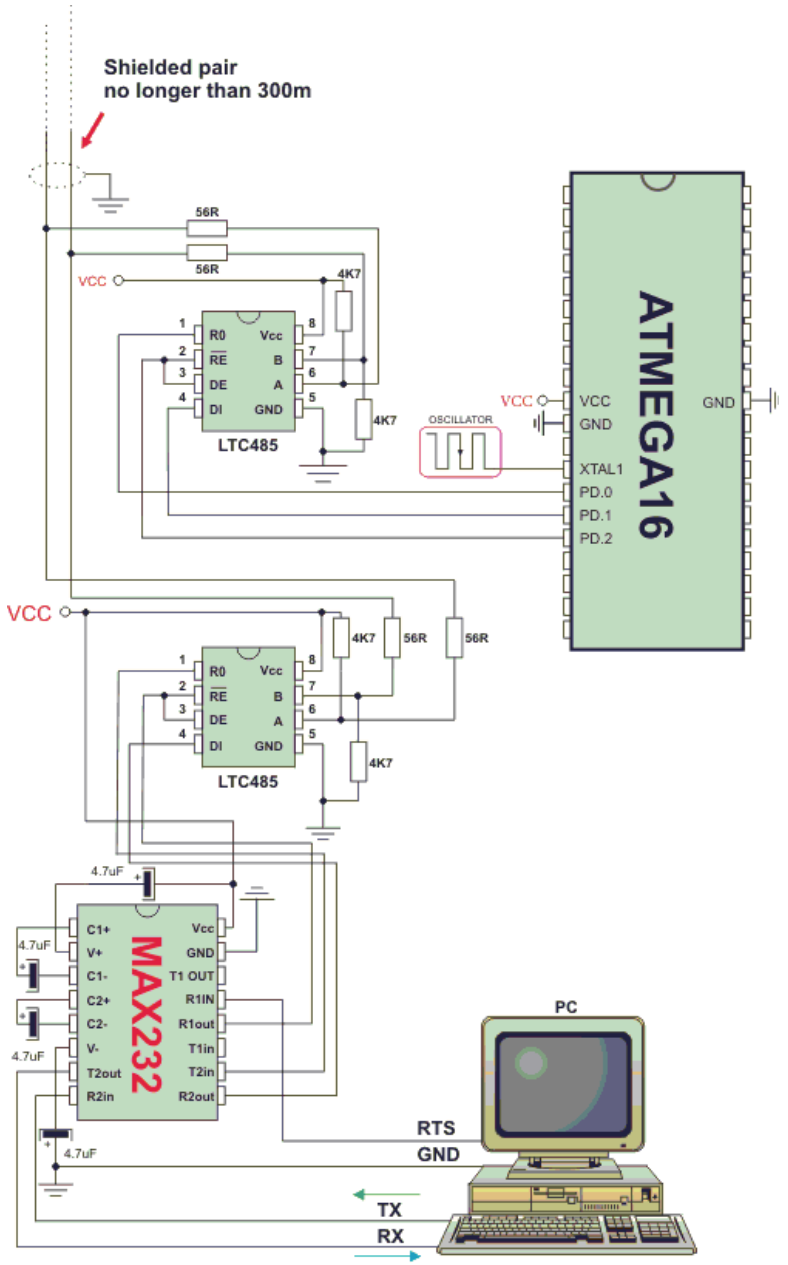
В більшості випадків **UART** є частиною великої інтегральної схеми (наприклад, мікроконтролера), але також може бути окремою мікросхемою (наприклад, Intel I8251, WD1402A)



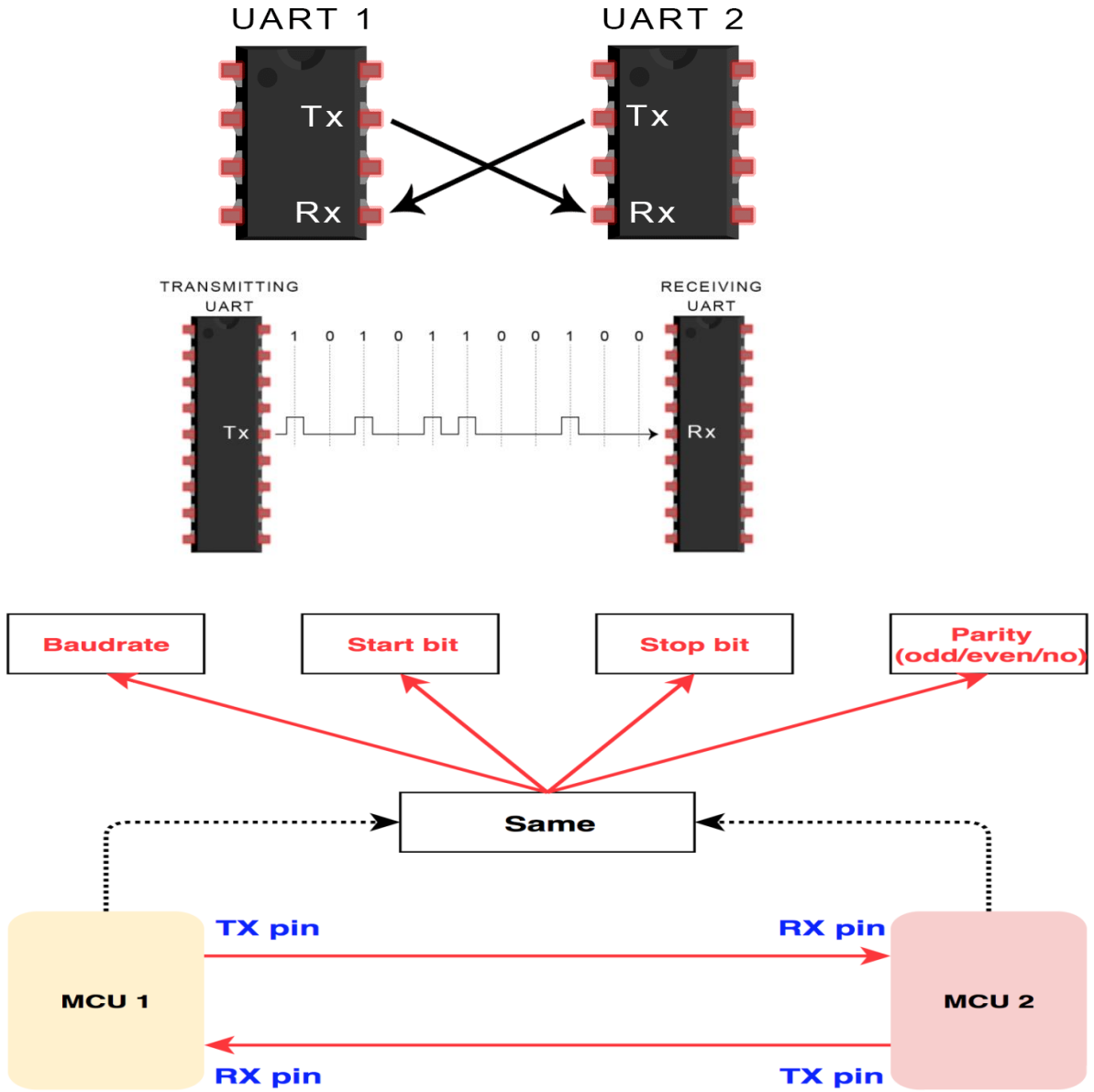
Формат фрейму передачі даних по UART

Приклади використання UART для стандартних інтерфейсів RS232 та RS485





З’єднання пристроїв по UART. Опис процесу передачі даних.



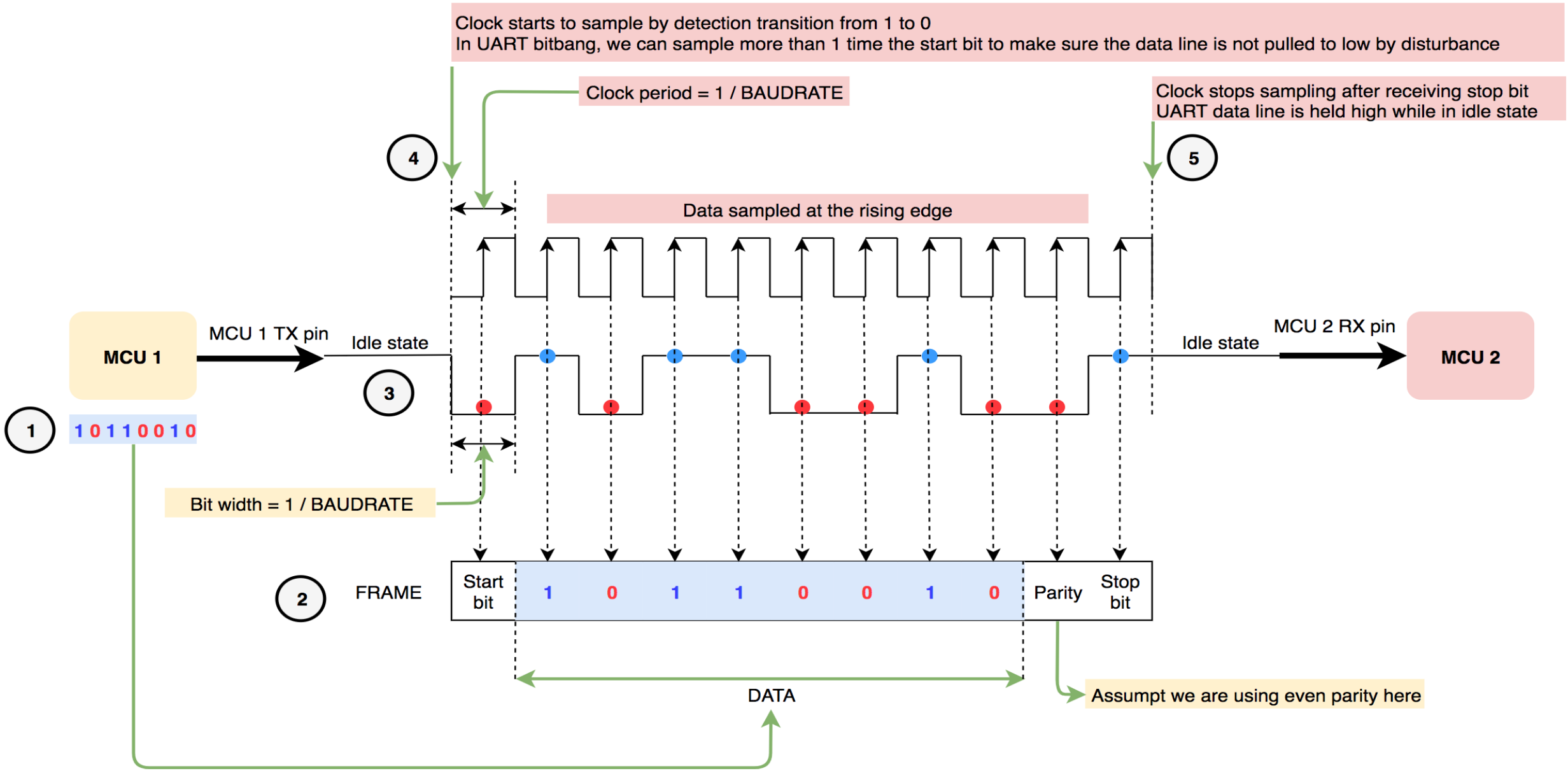
Крок 1. Обидві сторони повинні бути налаштовані з однаковою швидкістю передачі даних, парністю і кількістю стоп-бітів.
Процес:
Крок 2: MCU1 створює повний кадр даних із бітами Start - Data - Parity - Stop bit.

Крок 3: MCU1 починає генерувати сигнал на своєму виводі-TX відповідно до кадру даних. Бітова ширина 1/ BAUDRATE.

Крок 4: Прямо в той момент, коли MCU2 отримує падаючий фронт сигналу на лінії, він запускає тактовий сигнал для початку вибірки даних. Тактовий період на MCU2 = бітова ширина на MCU1 = 1/BAUDRATE. Ось чому ми повинні мати однакову швидкість передачі даних з обох боків, або MCU2 не зможе здійснювати дискретизацію на правильній частоті, що призведе до збою в зчитуванні даних. Існує кілька способів налаштування вибірки: вибірка по висхідному фронту тактового сигналу, вибірка по нижньому фронту або вибірка на середині імпульсу.

Крок 5: MCU2 зупиняє тактовий генератор, який використовується для вибірки даних після прийому стоп-біта, Після цього кроку дані будуть передані на обробку.

Процес передачі даних по UART

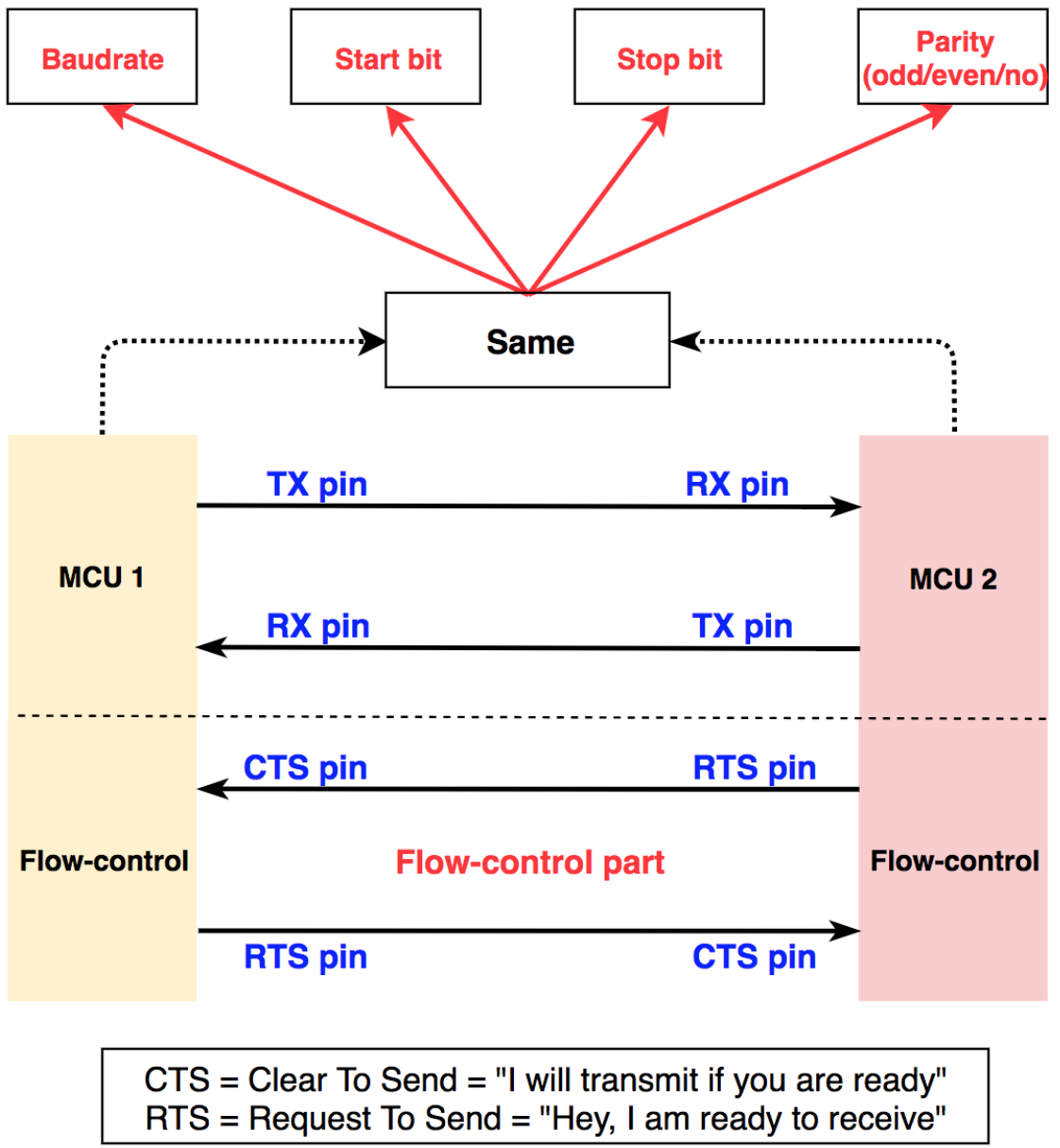


Контроль потоку передачі даних

Розглянемо випадок: *MCU1* передає дані в *MCU2*. Однак час обробки *MCU2* повільніший, ніж *MCU1*. У якийсь момент *MCU2* більше не може встигати за цим. Він або обробляє дані, або спорожнює буфер прийому, перш ніж продовжувати отримувати дані. Якщо ні, це може закінчитися втратою даних, поки *MCU1* продовжує надсилати повідомлення, поки *MCU2* зайнятий старими даними.

Використання регулятора потоку для вирішення цієї проблеми.

Контроль потоку - це метод обробки зв'язку по **UART** між швидким і повільним пристроєм без ризику втрати даних. *MCU2* використовуватиме управління потоком, щоб створити сигнал назад до *MCU1* для зупинки/паузи або відновлення передачі. Існує кілька способів реалізації контролю потоку в протоколі **UART**, використовуючи апаратне управління потоком або програмне управління потоком.

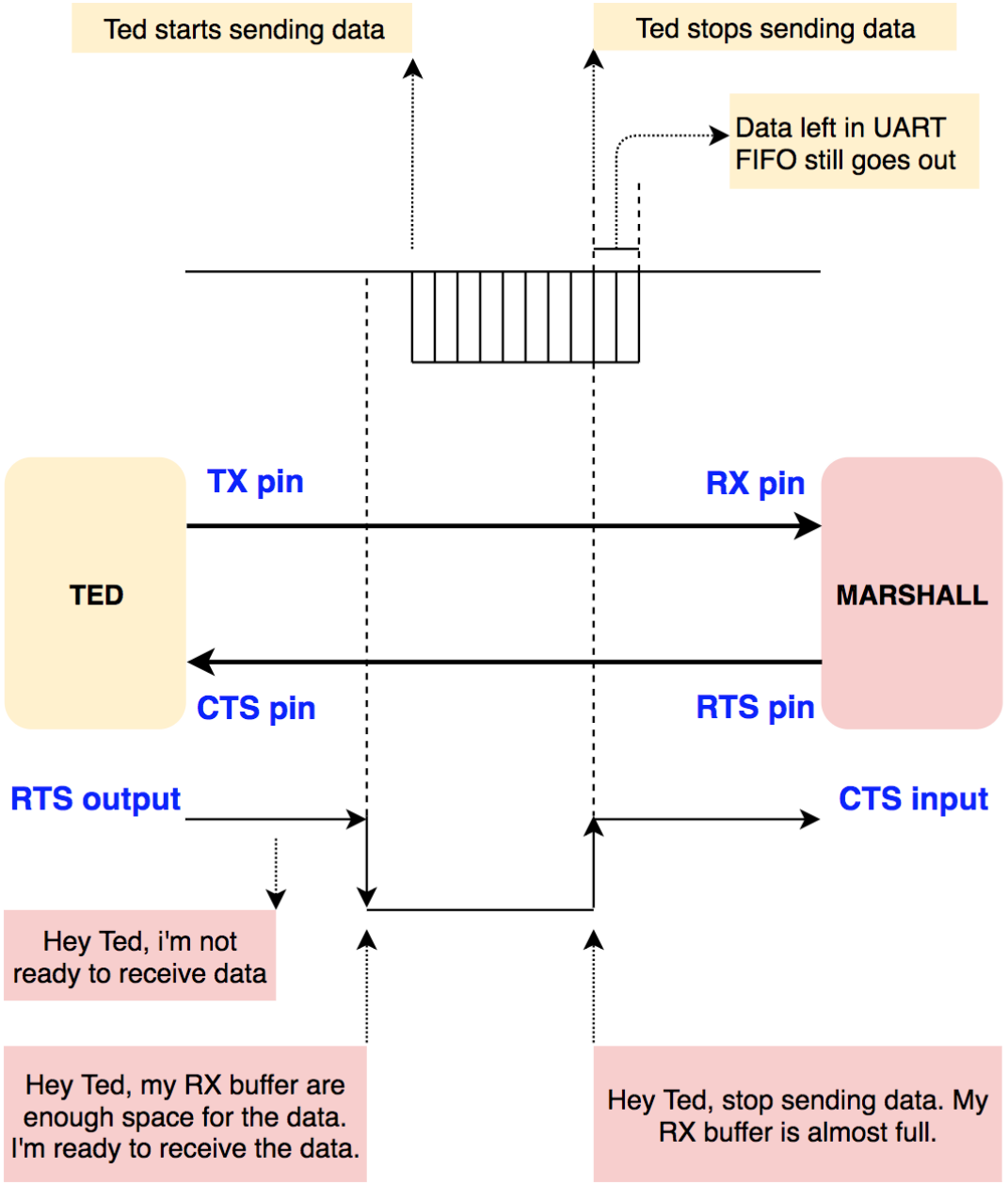


CTS = Clear To Send = "I will transmit if you are ready"
RTS = Request To Send = "Hey, I am ready to receive"

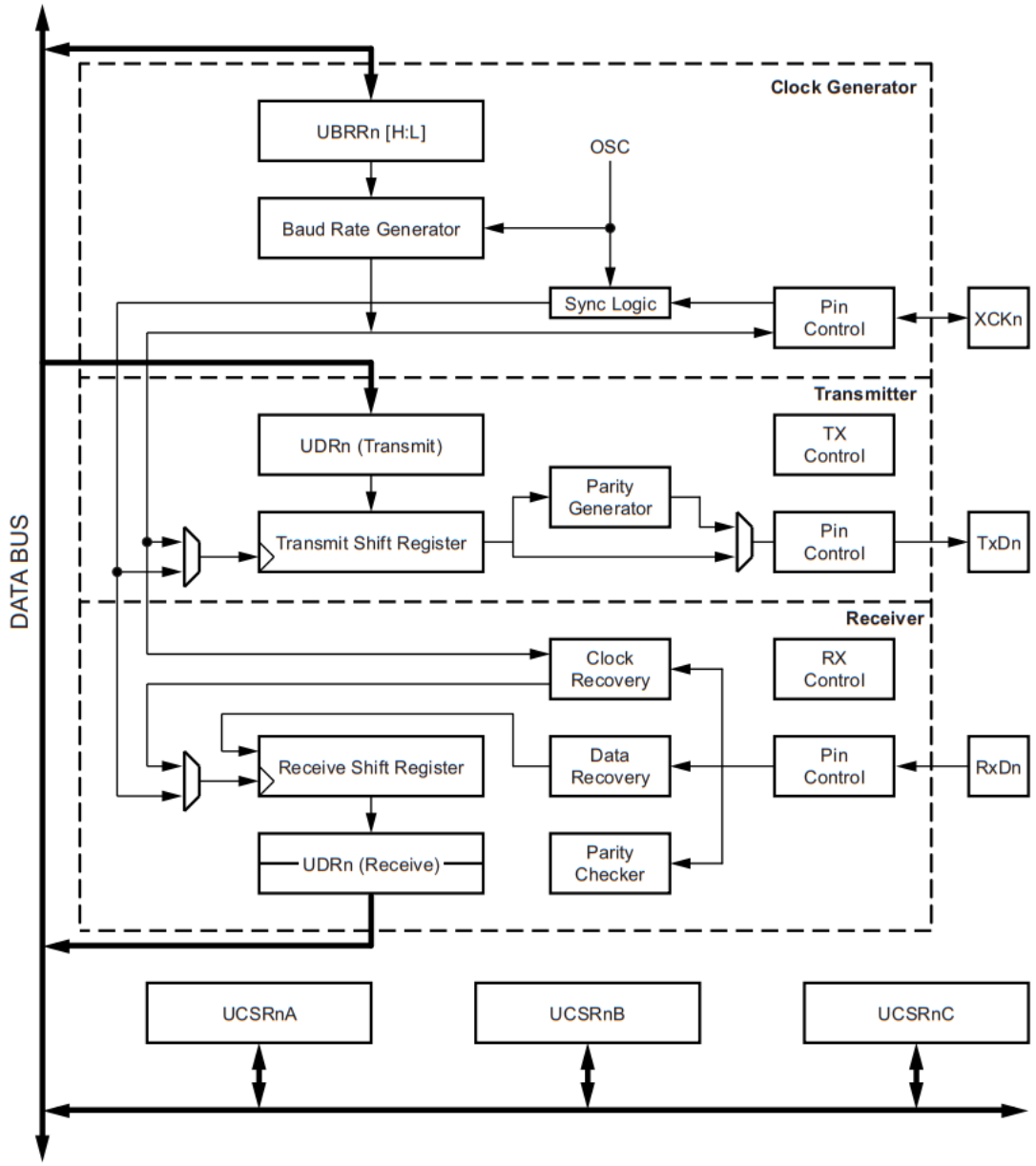
Контроль потоку передачі даних (продовження)

A (Ted) хоче би передати дані ***B (Marshall)***, перед тим, як відправити дані, ***A*** перевіряє A_CTS (він же B_RTS). Якщо на лінії низький рівень то ***B*** підтверджує що він готовий отримати більше даних, відповідно ***A*** продовжує їх надсилати. Кожного разу, коли ***B*** хоче припинити отримання даних для обробки чогось іншого, він скасовує B_RTS (A_CTS) → ***A*** перевіряє цю лінію і припиняє надсилання даних.

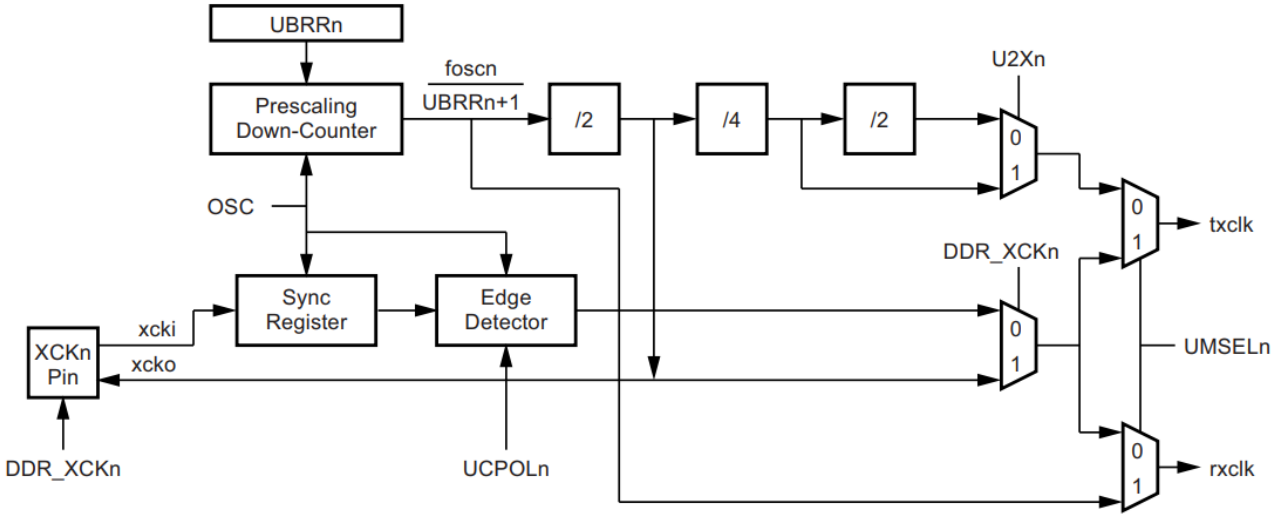
Замість того, щоб використовувати додаткові виводи для управління потоком, часто використовують програмне управління потоком. Зупинка або запуск потоку відбувається, надсиланням спеціального символа управління потоком на лінії ***TX/RX***. Спеціальними символами управління потоком зазвичай є ASCII-код **XON** та **XOFF** (0x11 та 0x13). Коли ***A*** надсилає **XOFF** у ***B***, ***B*** зупиняє передачу до отримання **XON** від ***A***.



Структура модуля USART мікроконтролера ATMega328



Блок-схема генератора швидкості прийому/передачі (Clock Generator)



Регістри керування модулем USART мікроконтролера ATMega328

Регістри контролю та статусу модуля USART ATMega328: **UCSRnA, UCSRnB, UCSRnC**

Bit	7	6	5	4	3	2	1	0	
	RXCn	**TXCn**	**UDREn**	**FEn**	**DORn**	**UPEn**	**U2Xn**	**MPCMn**	**UCSRnA**
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	**TXCIEn**	**UDRIEn**	**RXENn**	**TXENn**	**UCSZn2**	**RXB8n**	**TXB8n**	**UCSRnB**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	**UMSELn0**	**UPMn1**	**UPMn0**	**USBSn**	**UCSZn1**	**UCSZn0**	**UCPOLn**	**UCSRnC**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

UDRn – регістр даних (читання/запис)

UBRRnH, UBRRnL – регістри задання швидкості передачі.

n – номер модуля UART

Опис бітів регістра **UCSRnA**

RXCn –Завершення прийому. Встановлюється в 1 за наявності непрочитаних даних в буфері приймача (регістр даних ***UDR***). Скидається апаратно після читання буфера.

TXCn – Завершення передачі. Встановлюється в 1 після передачі всіх бітів з регістра зсуву передавача, за умови, що в регістр даних ***UDRn*** не було завантажено нового значення. Скидається апаратно при виконанні процедури обробки переривання або програмно, записом в нього логічної 1.

UDREn – Спустошення регістра даних. Встановлюється в 1 при порожньому буфері передавача (після пересилки байта з регістра даних ***UDRn*** у регістр зсуву передавача). Встановлений біт означає, що в регістр даних можна завантажувати нове значення. Якщо розряд ***UDRIEn*** регістра ***UCSRnB*** установлено, генерується запит на переривання «регістр даних – порожній». Скидається апаратно, при записі в регістр даних.

FEn – Помилка кадру. Якщо перший стоп-біт прийнятої посилки рівний 0, даний біт встановлюється в 1. Скидається апаратно при прийомі стоп-біта рівного 1.

DORn – Біт переповнення. Встановлюється в 1, якщо в момент виявлення нового старт-біта у регістрі зсуву приймача знаходиться останній прийнятий байт і буфер приймача ще незчитано.

UPEn – Помилка контролю парності. Встановлюється в 1, якщо в даних, що знаходяться в буфері приймача, виявлена помилка контролю парності. При відключеному контролі парності цей розряд постійно читається як 0.

U2Xn – Подвоєння швидкості обміну. Використовується тільки при асинхронному режимі роботи USART. У синхронному режимі він повинен бути скинутий.

MPCMn – Режим мультипроцесорного обміну. Якщо він установлений в 1 то, ведений мікроконтролер очікує на прийом кадру, що містить адресу. Кадри які не містять адреси пристрою, ігноруються.

Опис регістрів **UCSRnB** і **UCSRnC**

Біти регістра **UCSRnB:**
RXCIEn – Дозвіл переривання по завершенню прийому. Якщо в даний розряд записано 1, то при встановленні прапора ***RXCn*** регістра ***UCSRnA*** генерується переривання.
TXCIEn – Дозвіл переривання по завершенню передачі. Якщо в даний розряд записано 1, то при встановленні прапора ***TXCn*** регістра *UCSRnA* генерується відповідне переривання.
UDRIEn – Дозвіл переривання при очищенні регістра даних UART. Якщо в даний розряд записано 1, то при встановленні прапора ***UDREn*** в регістрі ***UCSRnA*** генерується переривання.
RXENn – Дозвіл прийому даних.
TXENn – Дозвіл передачі даних.
UCSZn2 – Формат посилки. У модулях USART він використовується спільно з розрядами ***UCSZn1*** і ***UCSZn0*** регістра ***UCSRnC***.
RXB8n – 8-й розряд отриманих даних, при використанні 9-розрядних посилок. Вміст цього розряду має бути зчитаний до читання регістра даних ***UDRn***.
TXB8n – 8-й розряд даних для передачі, при використанні 9-розрядних посилок. Необхідне значення має бути занесено в цей розряд до завантаження байта даних в регістр ***UDRn***.

Біти регістра **UCSRnC:**
UMSELn1, UMSELn0 – Вибір режиму роботи USART. (00 - модуль USART працює в асинхронному режимі).
UPMn1, UPMn0 – Функціонування схем контролю та формування парності (00 - відключено перевірку парності)
USBSn – Кількість стоп-бітів. (***USBSn*** =0 – 1 стоп-біт, ***USBSn*** =1 – 2 стоп-біти)
UCSZn1, UCSZn0 – Формат посилки (011 – 8-біт)
UCPOLn – Задає полярність синхронізації ***XCKn*** (використовується тільки в синхронному режимі).

Розрахунок швидкості передачі даних по UART

Формула розрахунку «швидкості передачі»

```
// Для U2X = 0
// UBRR = F_CPU / (16 * baudrate) - 1;
UBRRnH = (F_CPU / (16 * baudrate) - 1) >> 8;
UBRRnL =  F_CPU / (16 * baudrate) - 1;

// Для U2X = 1
// UBRR = F_CPU / (8 * baudrate) - 1;
UBRRnH = (F_CPU / (8 * baudrate) - 1) >> 8;
UBRRnL =  F_CPU / (8 * baudrate) - 1;
```

Розрахунок помилки визначення швидкості передачі даних

$$\text{Error}[\%] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1\right) \times 100\%$$

Таблиця розрахунку бод-рейту (швидкості передачі/прийому) для різних тактових частот

Baud Rate (bps)	f_{osc} = 8.0000MHz				f_{osc} = 11.0592MHz				f_{osc} = 14.7456MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	207	0.2%	416	–0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	–0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	–0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	–3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	–7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	–3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	–7.8%	5	–7.8%	3	–7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	–7.8%	1	–7.8%	3	–7.8%
1M	–	–	0	0.0%	–	–	–	–	0	–7.8%	1	–7.8%

Робота з модулем USART мікроконтролера ATMega328

Ініціалізація модуля USART0

```
void USART_Init(uint32_t baud)
{
  //Розрахунок швидкості(U2X = 0)
  UBRR0H = (uint8_t)((F_CPU/(16*baud)-1)>>8);
  UBRR0L = (uint8_t)(F_CPU/(16*baud)-1);

  UCSR0A = 0;
  //Включити передавач та приймач
  UCSR0B = (1 << RXEN0)|(1<<TXEN0);
  //Налаштування фрейму передачі/прийому даних
  //8-біт даних, 1-стоп біт, без перевірки парності
  UCSR0C = (1<<UCSZ01)|(1<<UCSZ00);
}
```

Функція передачі даних по UART

```
void USART_PutChar(uint8_t data)
{
    // Чекаємо завершення передачі даних
    while (!(UCSR0A & (1<<UDRE0))){ ; }
    // Почати передачу даних
    UDR0 = data;
}
```

Функція прийому даних по UART

```
uint8_t USART_Receive(void)
{
    // Чекаємо завершення прийому даних
    while (!(UCSR0A & (1<<RXC0))){ ; }
    // Повертаємо прийняті дані
  return UDR0;
}
```

Тестова програма «exo»

```
#include <avr/io.h>

void USART_Init(uint16_t baud);   //...
void USART_PutChar(uint8_t data); //...
uint8_t USART_Receive(void);      //...

void print_string(const char *str)
{
    while(*str){
        USART_PutChar(*str++);
    }
}
//=====================================
int main(void)
{
  USART_Init(9600);
  print_string("Hello!\r\n");
    // main loop ============
    for(;;)
    {
      USART_PutChar(USART_Receive());
    }
  return 0;
}
```

Використання стандартних потоків вводу/виводу для роботи з UART

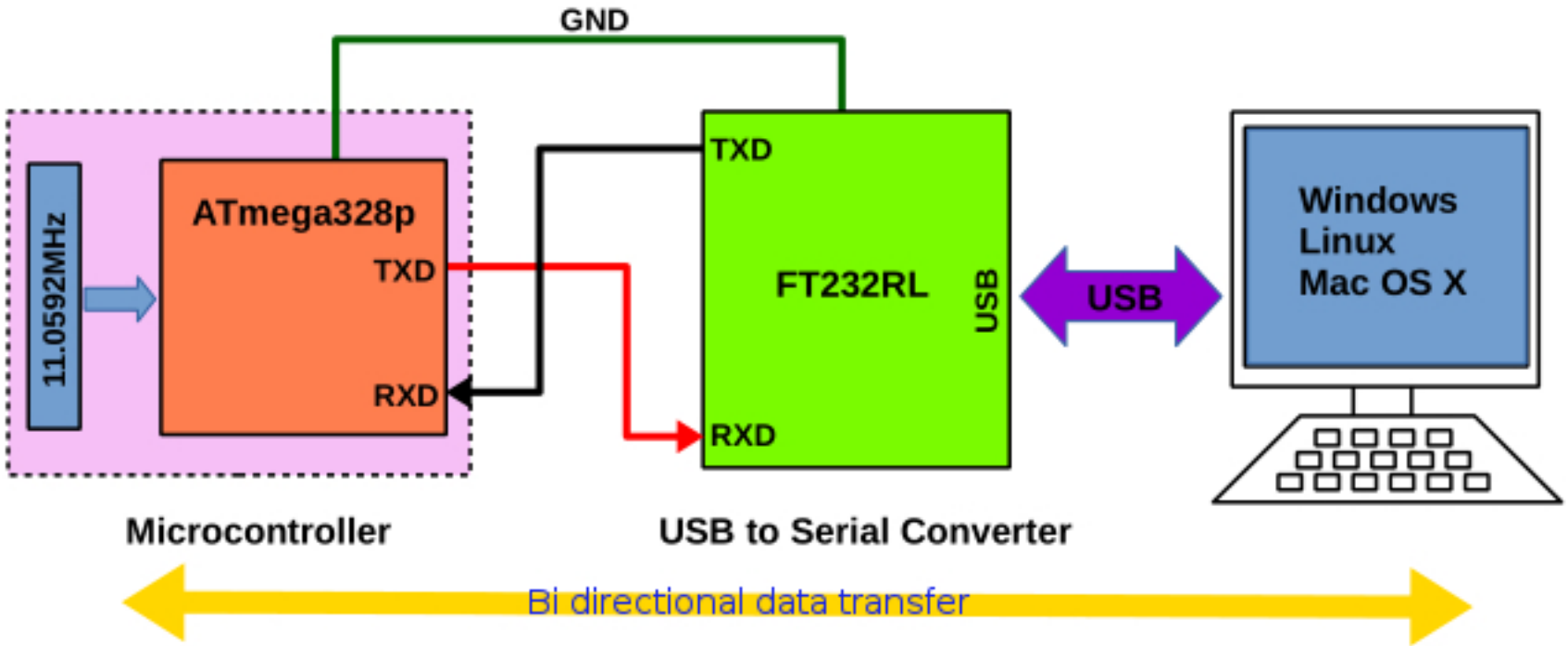
```
#include <avr/io.h>
#include <stdio.h>

void USART_Init(uint16_t baud);   //...
void USART_PutChar(uint8_t data); //...
uint8_t USART_Receive(void);      //...

int uart_putch(char ch, FILE *stream)
{
   if (ch == '\n')
      USART_PutChar('\r');
   USART_PutChar(ch);
 return 0;
}

int uart_getch(FILE *stream)
{
  uint8_t ch;
    ch = USART_Receive();
    // Echo the Output Back to terminal
    USART_PutChar(ch);
  return ch;
}
```

```
//продовження...

// Assign I/O stream to UART
FILE uart_stream = FDEV_SETUP_STREAM(uart_putch,
                                     uart_getch, _FDEV_SETUP_RW);

int main(void)
{
    USART_Init(9600);

    // Define Output/Input Stream
    stdout = stdin = &uart_stream;

    printf("Hello World\n");

    // Loop forever
    for(;;)
    {
     //...
    }
  return 0;
}
```

Використання UART для виводу відладочної інформації

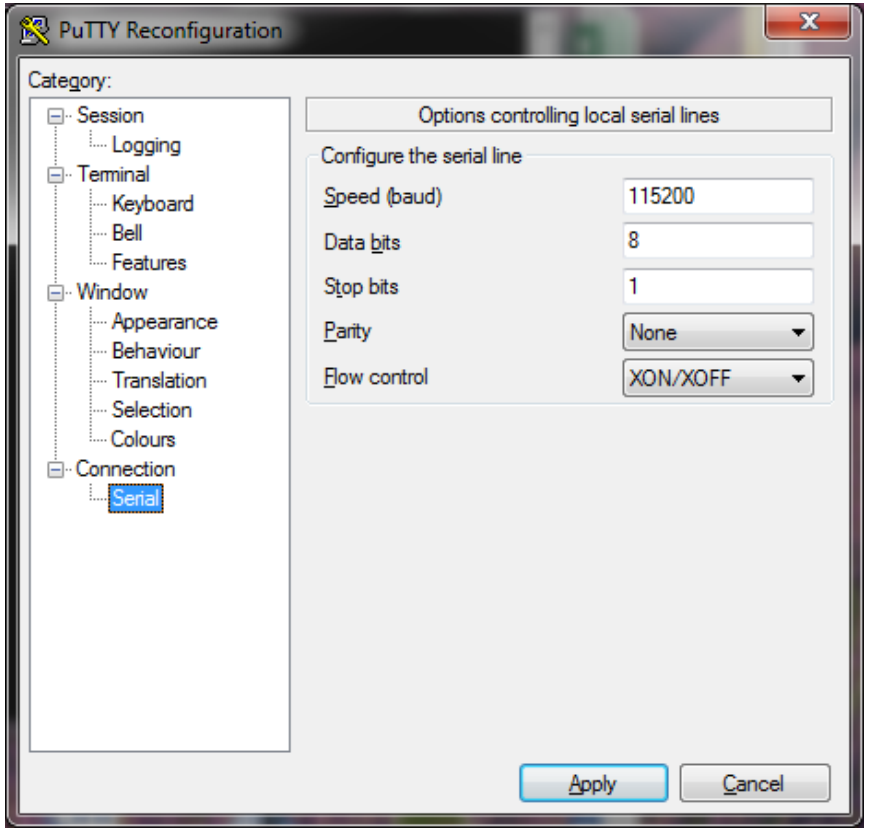
```
#include <stdio.h>
#include <stdlib.h>

#define DEBUG

//source
void printlog(const char *fmt, ...)
{
  va_list args;
    va_start(args, fmt);
    vfprintf(stdout, fmt, args);
}

//header
#if defined(DEBUG)
void printlog(const char *fmt, ...);
#define LOG_DEBUG(tg, fmt,...) \
    printlog("D [%s] (%s:%d) " fmt "\n", tg,  __FUNCTION__, __LINE__, ##__VA_ARGS__)
#else
#define LOG_DEBUG(tg, fmt,...)
#endif
```

Підключення мікроконтролера до комп’ютера з використанням USB-UART перетворювача



Програми для комунікації по UART на стороні персонального комп’ютера

MobaXterm, Terminal v1.9b by Bray, PuTTY



Вікно налаштувань послідовного порта програми MobaXterm