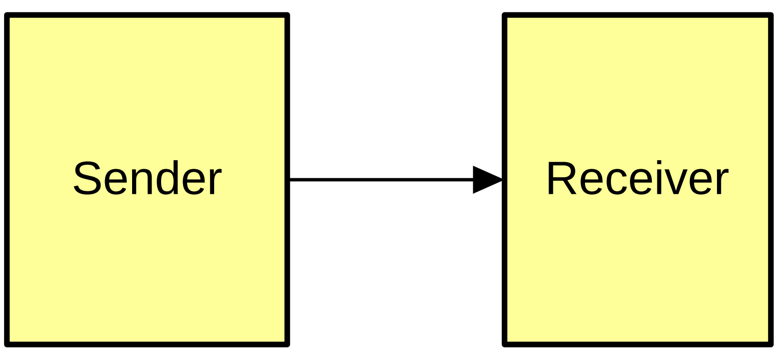
UART

Topics

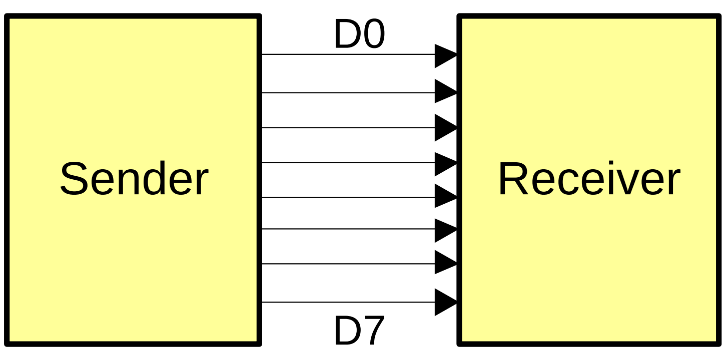
- Communication Theory
- Introduction to USART
- STM32 USART Registers
- STM32 USART Programming

Serial vs. Parallel

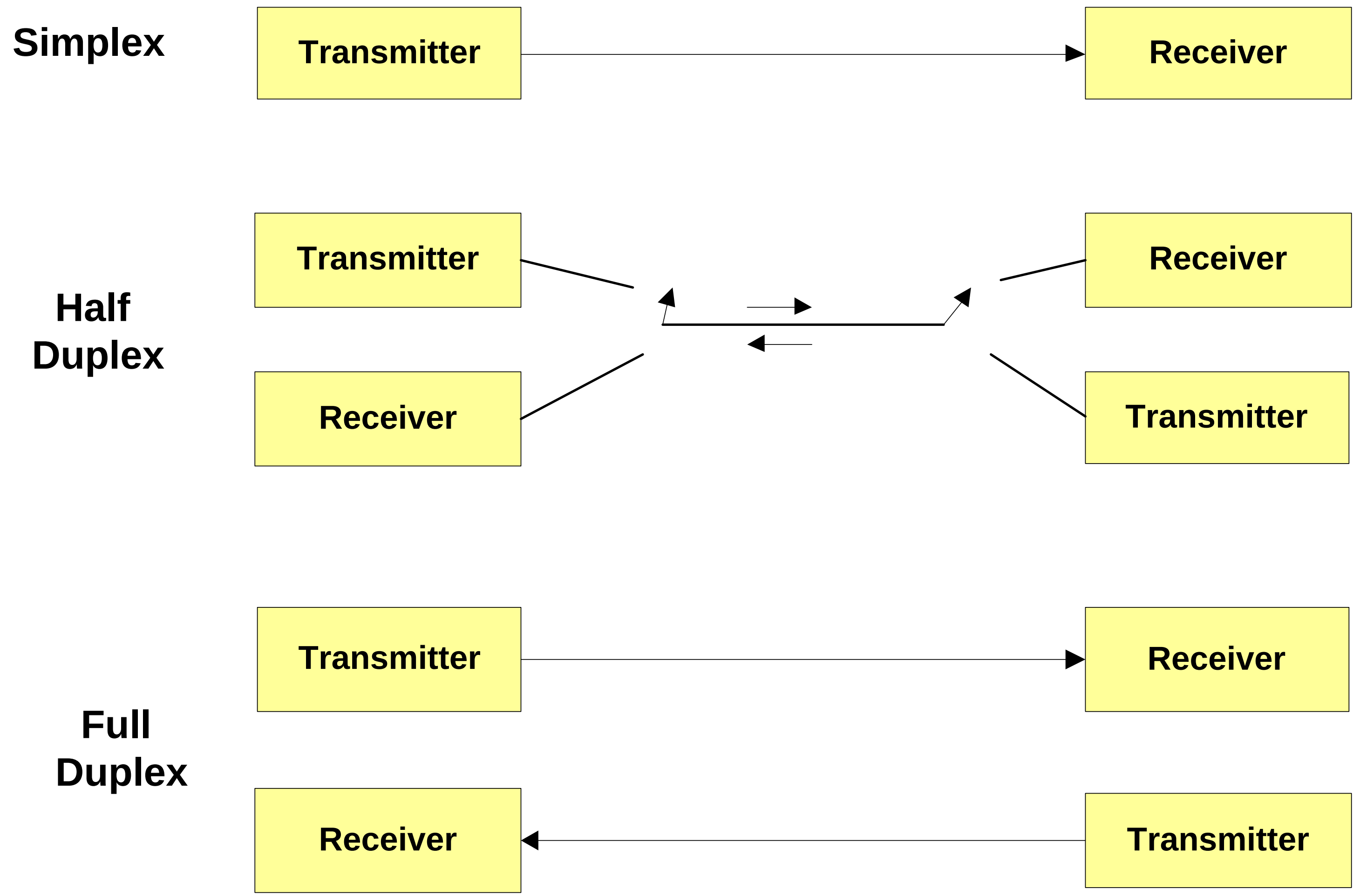
Serial Transfer



Parallel Transfer

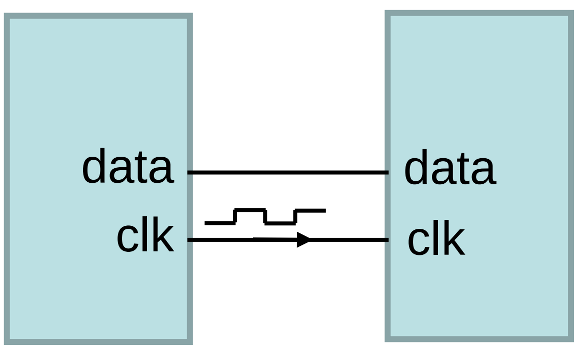


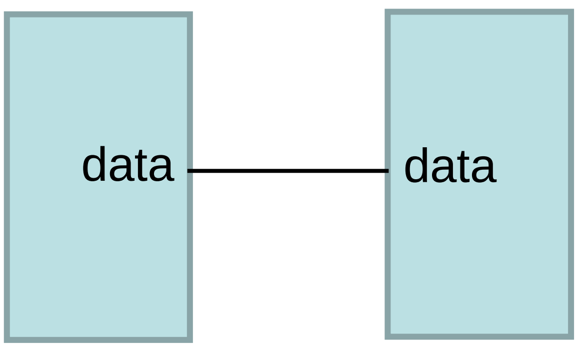
Direction


Simplex
Transmitter
Receiver
Half Duplex
Transmitter
Receiver
Receiver
Transmitter
Full Duplex
Transmitter
Receiver
Receiver
Transmitter

Synchronous vs. Asynchronous

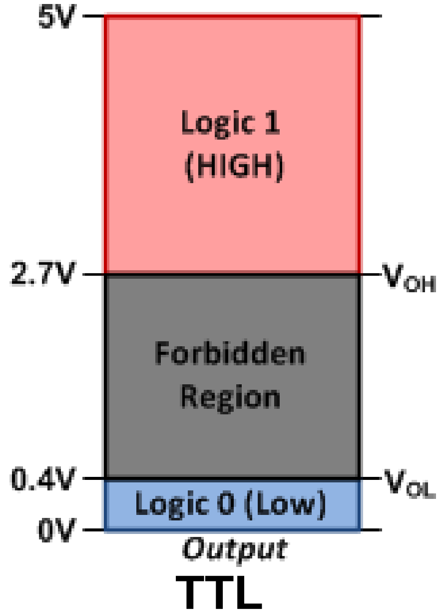
- Synchronous
- Asynchronous
  - No clock

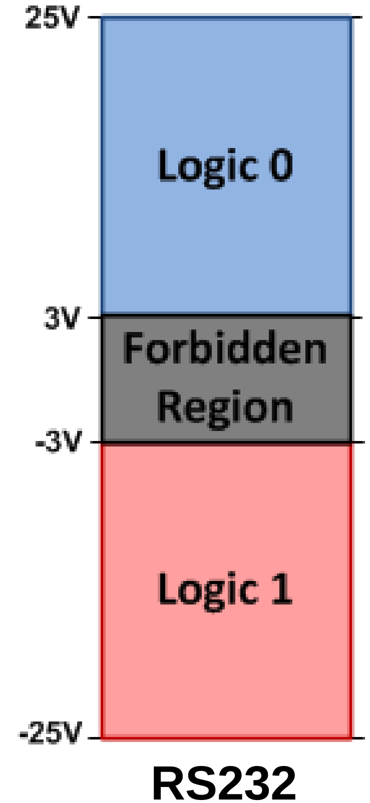




Line coding

- Representing 1 and 0 using signals
  - Level line coding
    - TTL
    - CMOS
    - RS232





Introduction to USART Protocol

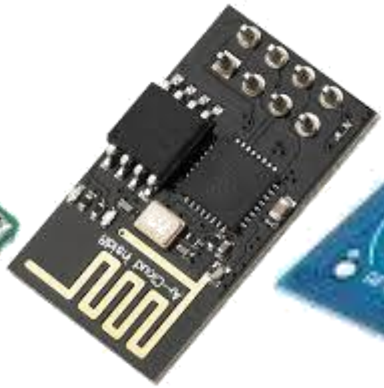
USART vs. UART

- USART: Universal synchronous and asynchronous Receiver-transmitter
  - Support both synchronous and asynchronous communications
- UART: Universal asynchronous Receiver-transmitter

UART

- Now, UART is widely used to communicate different devices with MCUs

SMS/GPRS



Wireless

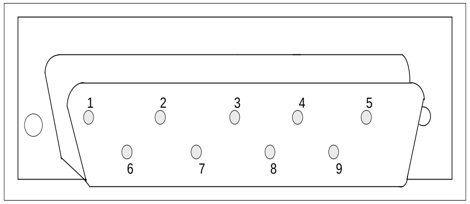
RFID Card Reader

Voice Player

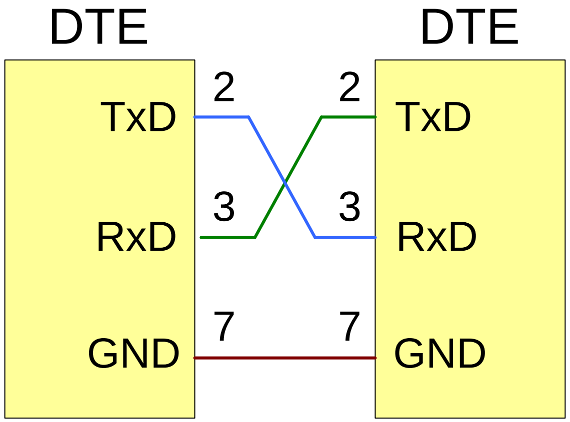
GPS

USART Pins

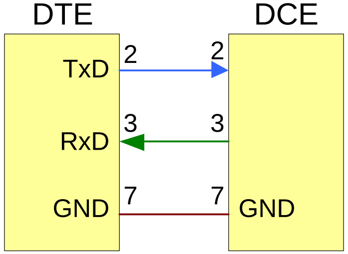
- DCD (Data Carrier Detect)
- RXD (Received Data)
- TXD (Transmitted Data)
- DTR (Data Terminal Ready)
- GND (Signal Ground)
- DSR (Data set ready)
- RTS (Request to send)
- CTS (Clear to send)
- RI (Ring Indicator)



DTE- DTE Connection

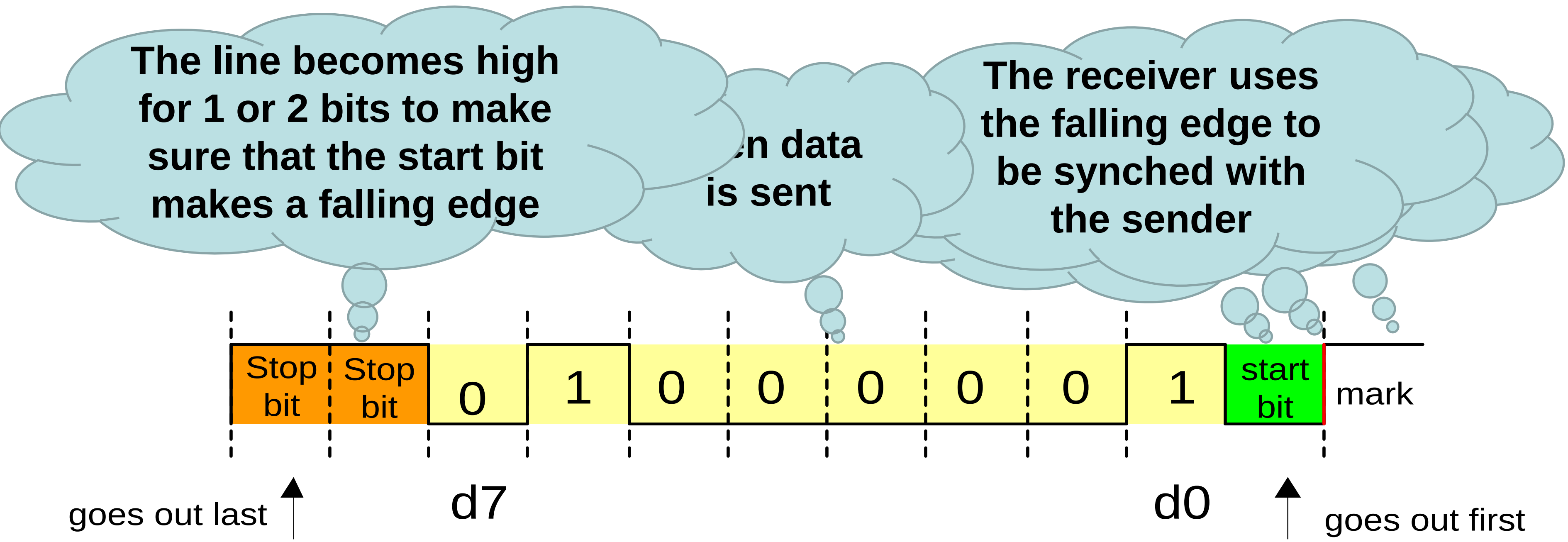


DTE- DCE Connection



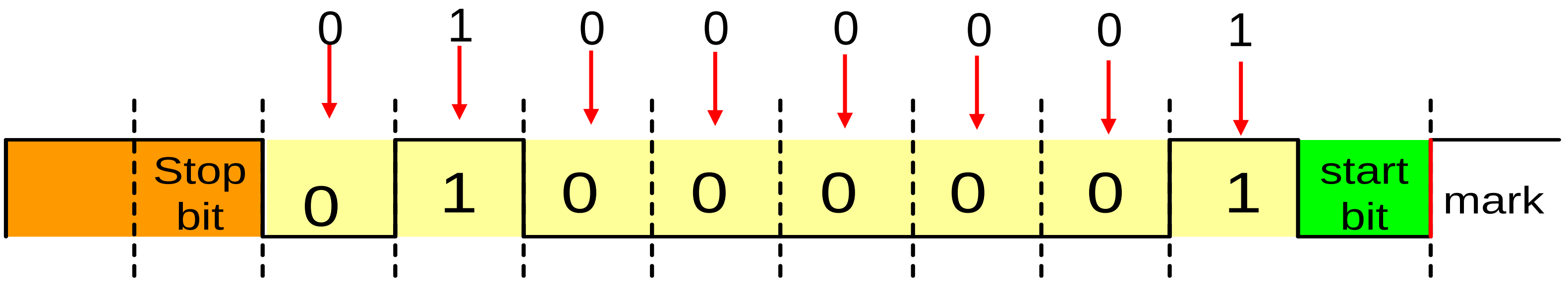
Sending data

- Start bit (0)
- Data
- Parity bit (optional)
- Stop bit (1)

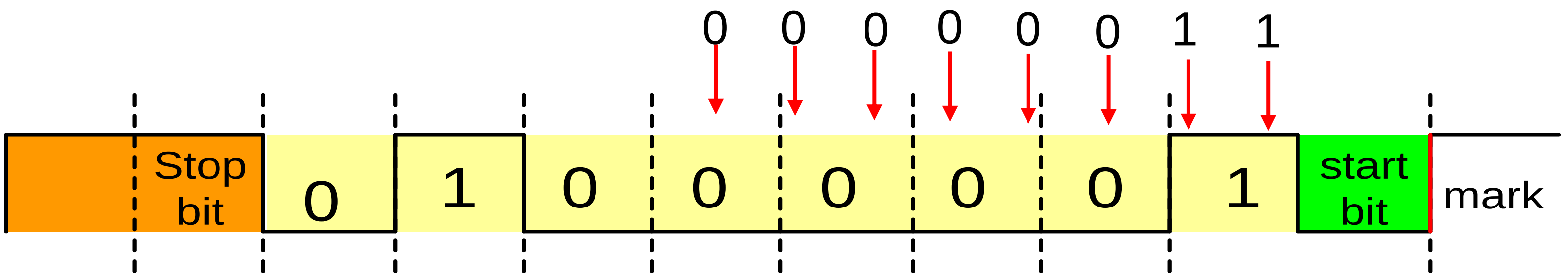


Speed (Baud rate)

- Sample rate = send rate

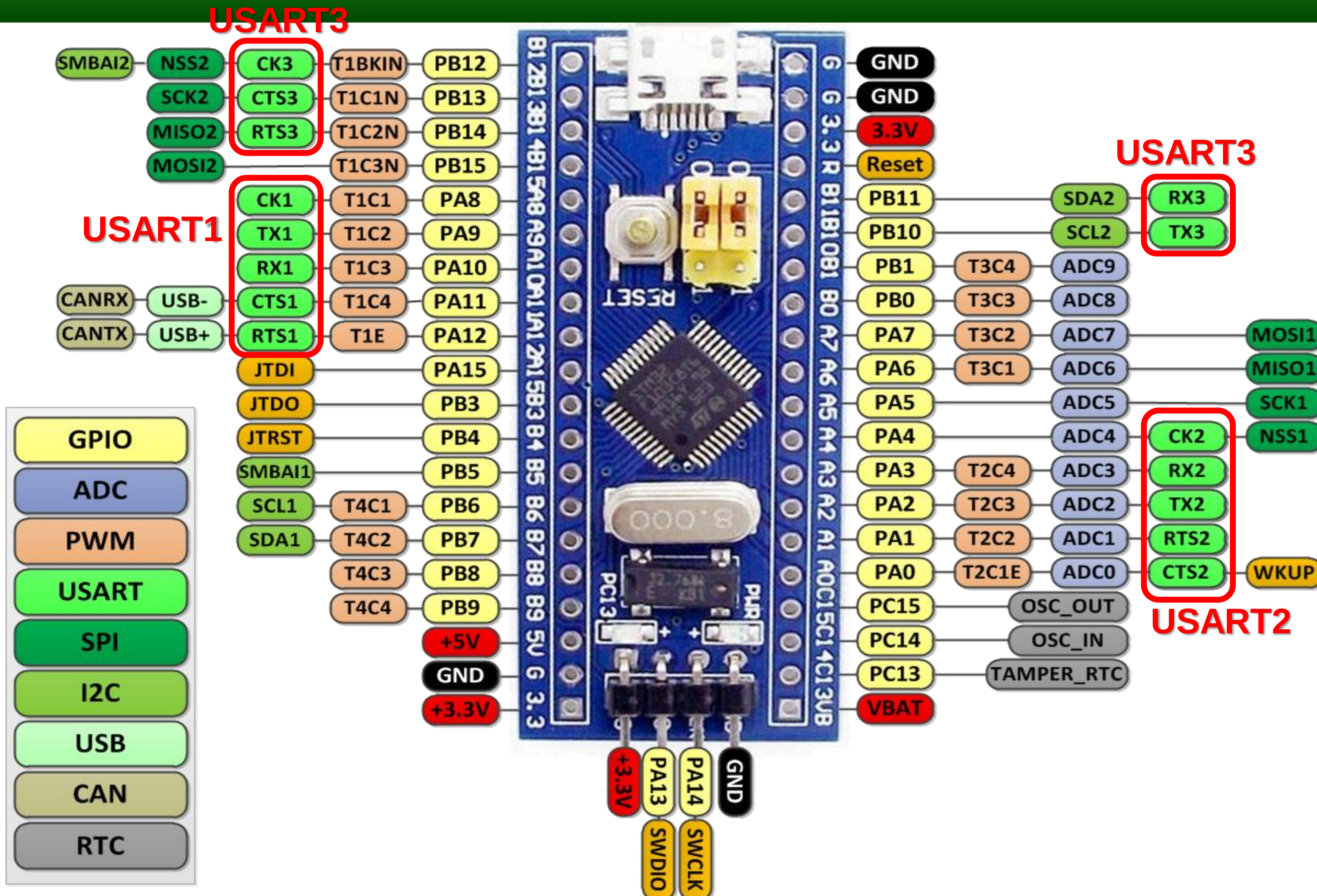


- Sample rate faster than sending

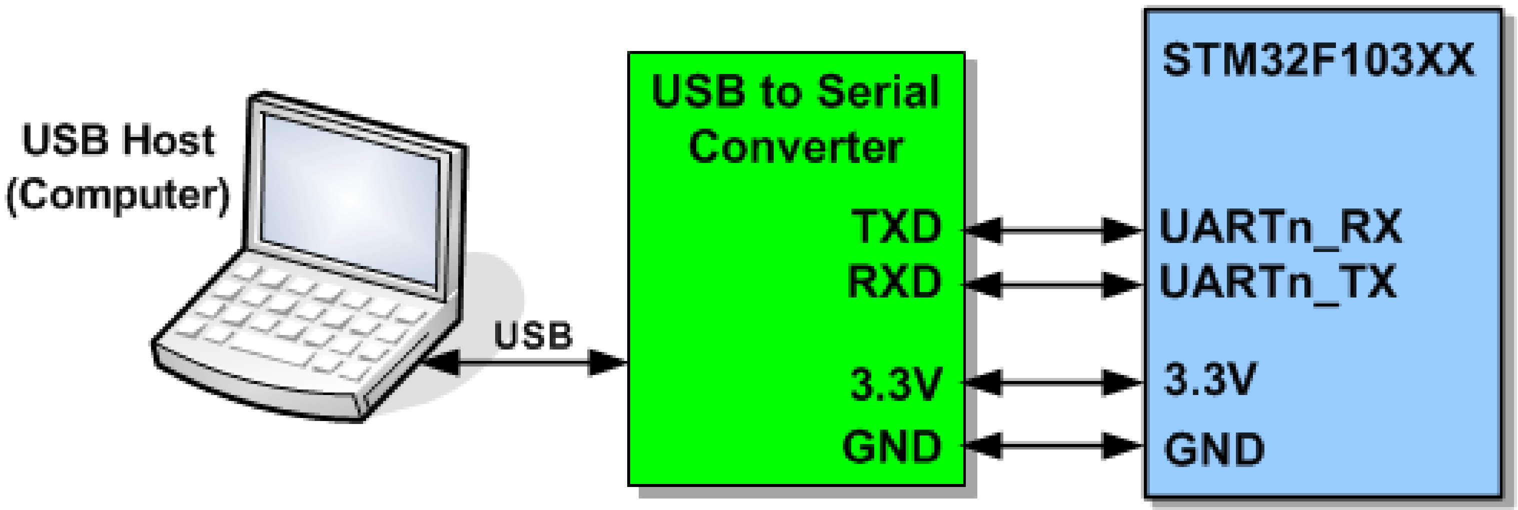


USART in STM32F10x

USART Pins in the Blue Pill

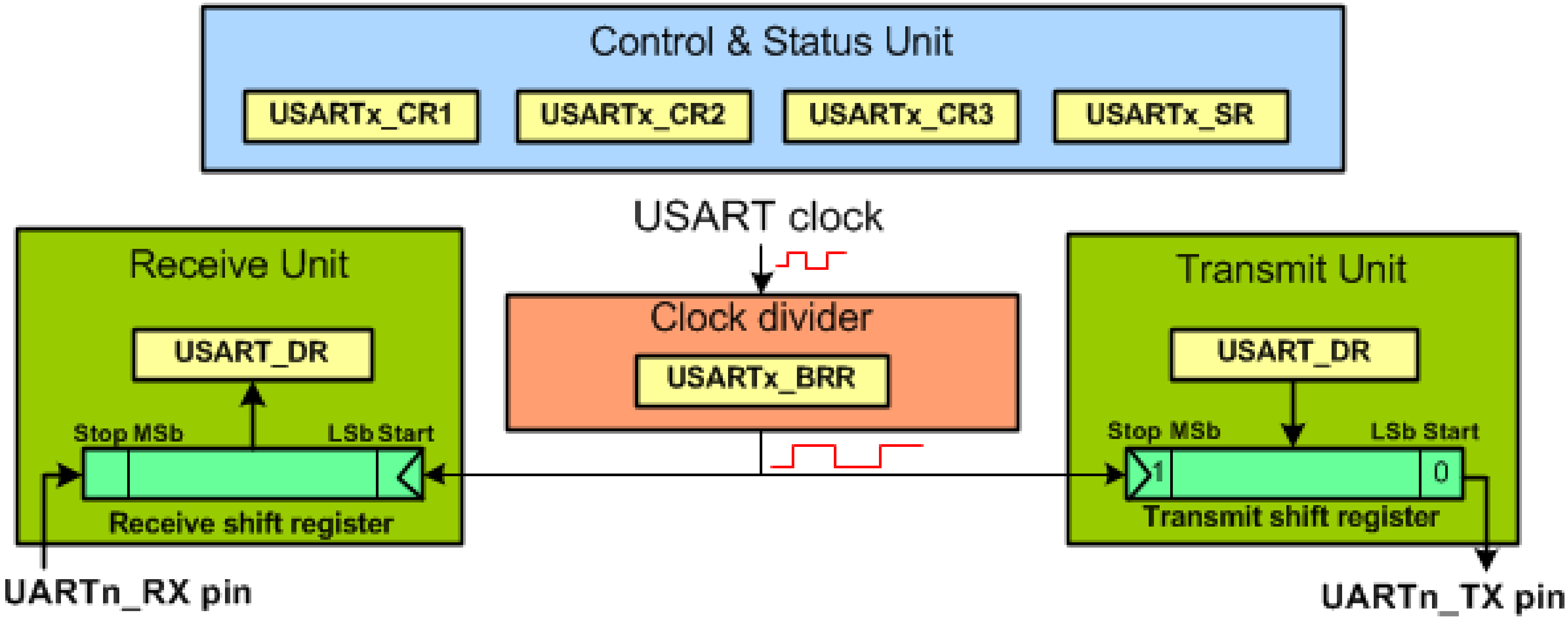


Connecting the Blue Pill to the PC using USART



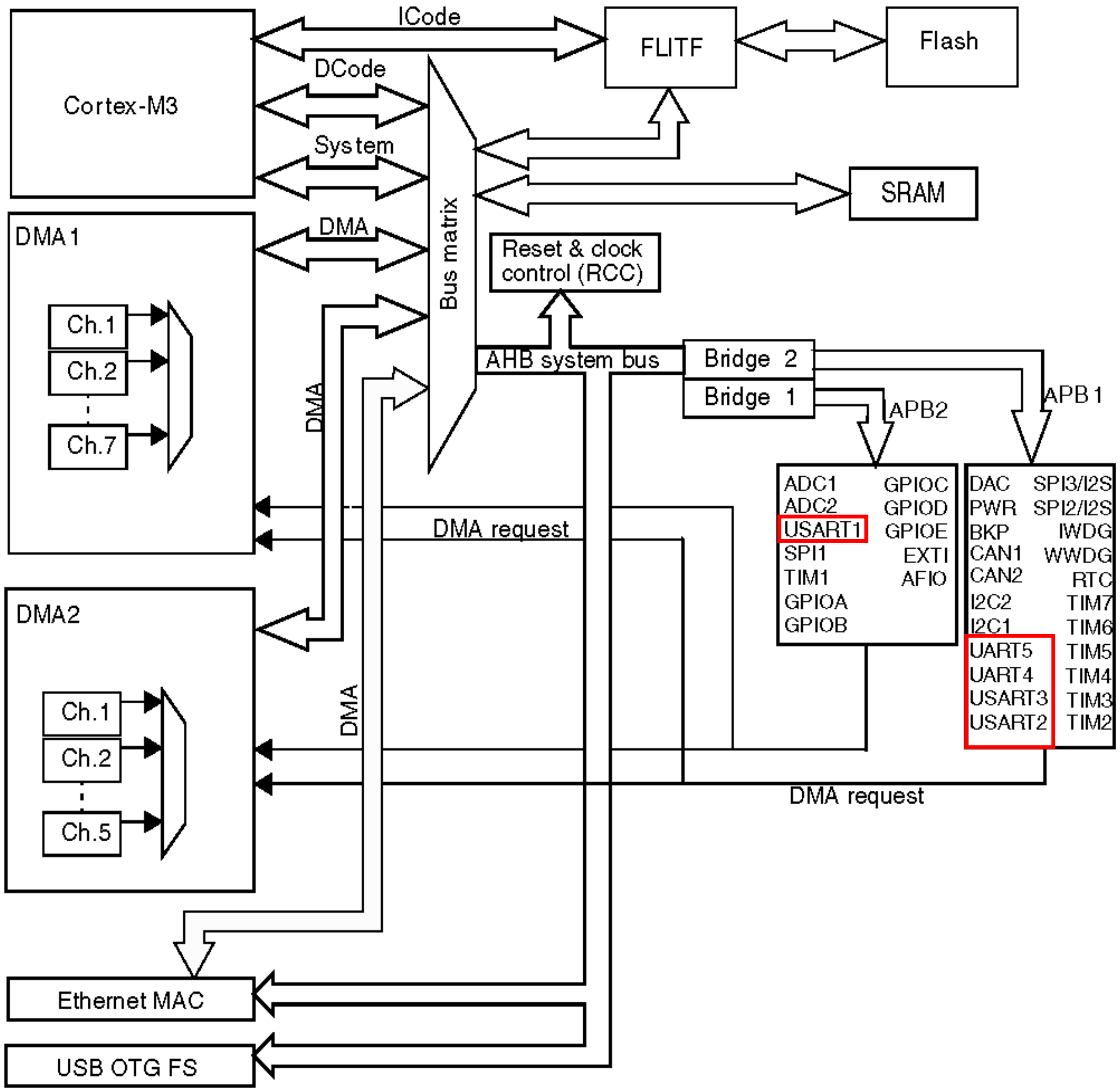
USART Registers

USART Registers



Configuring the USART

APB1, APB2, and AHB



Enabling Clocks

RCC_APB1ENR:
(RCC->APB1ENR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved		DACEN	PWR EN	BKPEN	Res.	CAN EN	Res.	USBEN	I2C2EN	I2C1EN	USART 5EN	USART 4EN	USART 3EN	USART 2EN	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3EN	SPI2EN	reserved		WWDG EN	reserved		TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN

RCC_APB2ENR:
(RCC->APB2ENR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved										TIM11 EN	TIM10 EN	TIM9 EN	Res.		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 EN	USART 1EN	TIM8 EN	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	IOPG EN	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.	AFIO EN

RCC_AHBENR:
(RCC->AHBENR)

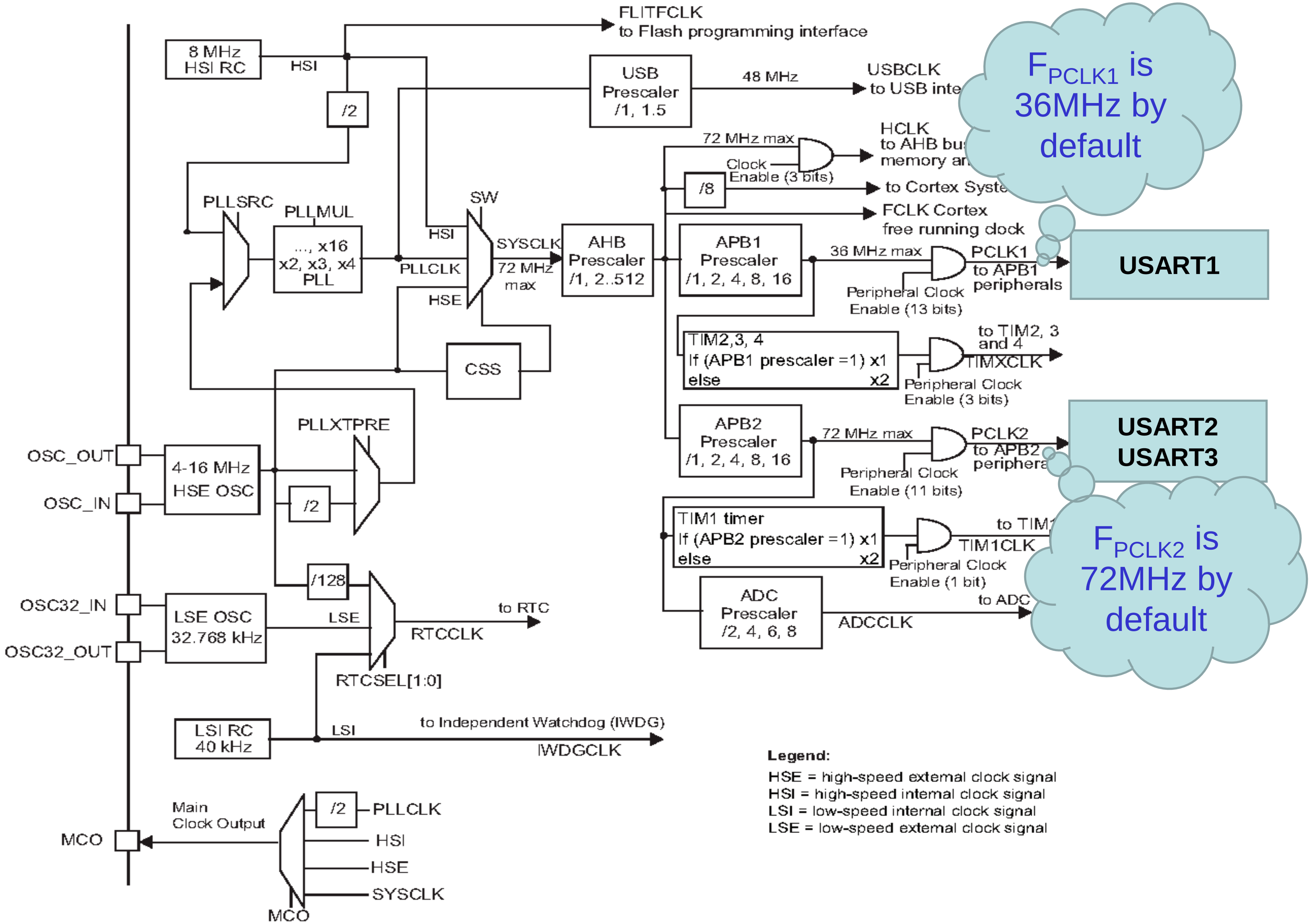
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved		DACEN	PWR EN	BKPEN	Res.	I2C2EN	Res.	USBEN	I2C2EN	I2C1EN	USART 5EN	USART 4EN	USART 3EN	USART 2EN	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved					SDIO EN	Res.	FSMC EN	Res.	CRC EN	Res.	FLITF EN	Res.	SRAM EN	DMA2 EN	DMA1 EN

Label	Description	Label	Description
IOPx	I/O port x clock enable	ADCnEN	ADCn clock enable
USARTnEN	USARTn clock enable	DACnEN	DACn clock enable
USBEN	USB clock enable	TIMnEN	TIMn timer clock enable
CANEN	CAN clock enable	SPInEN	SPI n clock enable
PWREN	Power interface clock enable	BKPEN	Backup interface clock enable
WWDG	Window watchdog clock enable	SDIOEN	SDIO clock enable
DMAnEN	DMAn clock enable	FSMCEN	FSMC clock enable
CRCEN	CRC clock enable	I2CnEN	I2Cn clock enable

Note: (0: clock disabled, 1: clock enabled)

STM32F10X Clock



Baud rate

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR:	DIV_Mantissa												DIV_Fraction			

(USARTx->BRR)

- Baud rate = F_{PCLKx} / BRR

Example: Find BRR for USART1 with baud rate of 57600

Solution:

USART1 is connected to APB1 bus and F_{PLK1} is 36MHz by default.
BRR = F_{PLK1}/Baud rate = 36M/57600 = 625

Control Register 1 (CR1)

USARTx_CR1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK

(USARTx->CR1)

Field	Bit	Description
UE	D13	0 = USART prescaler and outputs disabled. 1 = USART enabled
M	D12	Data format mode bit. We must use this to select 8-bit data frame size 0 = select 8-bit data frame and one start bit 1 = Select 9-bit data frame and one start bit
WAKE	D11	Wake-up condition bit. See the user manual 0 = Idle line wakeup 1 = Address mark wake-up
PCE	D10	Parity Control Enable bit. This will insert a parity bit right after the MSB bit. 0 = no parity bit 1 = parity bit
PS	D9	Parity select (used only if PE is one.) 0 = even parity bit 1 = odd parity bit
PEIE	D8	PE interrupt enable 0 = disabled 1 = A USART interrupt is generated whenever the PE flag of USART_SR is set.
TXEIE	D7	TXE interrupt enable 0 = disabled 1 = A USART interrupt is generated whenever the TXE flag of USART_SR

Control Register 2 (CR2)

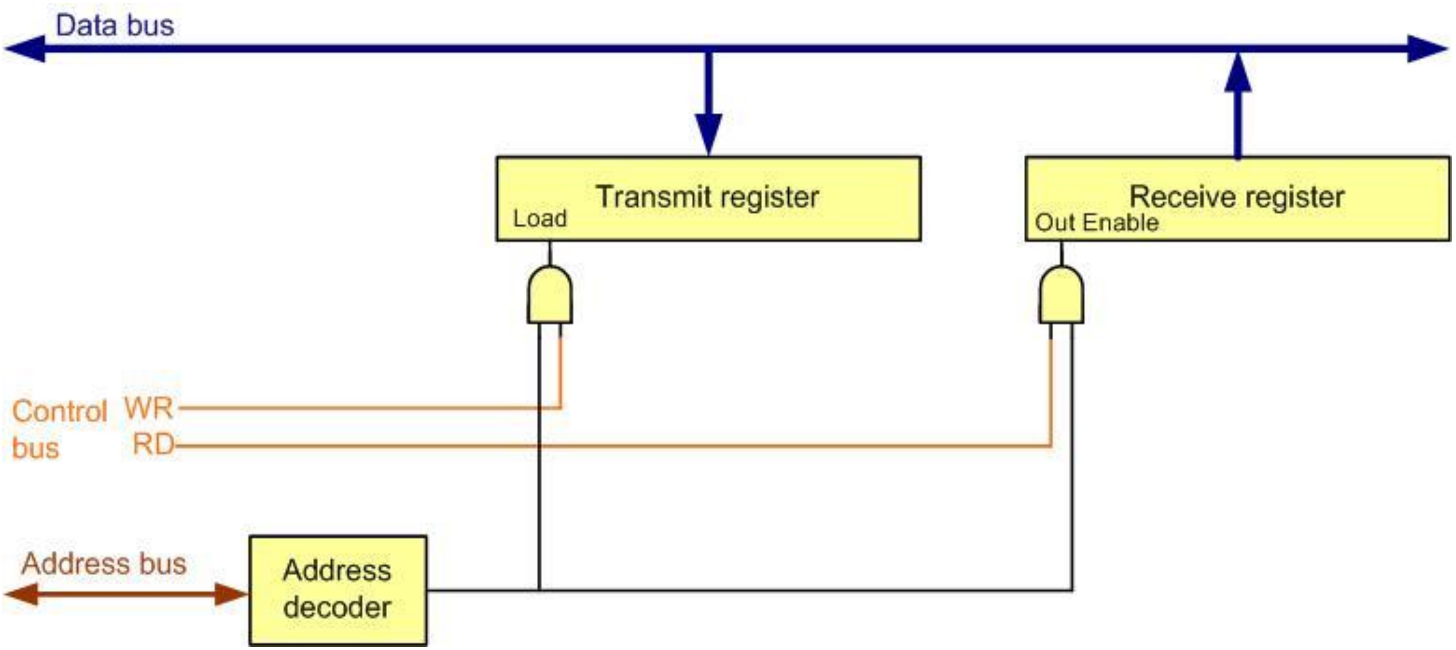
USARTx_CR2:
(USARTx->CR2)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LINEN	STOP		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	Res.	ADD			

Field	Bit	Description
STOP	D12,D13	Number of stop bits 0 = 1 stop bit 1 = 0.5 stop bit 2 = 2 stop bits 3 = 1.5 stop bits

Data Register (DR)

	D7	D6	D5	D4	D3	D2	D1	D0
USARTx_DR:								



Status Register (SR)

	15–10	9	8	7	6	5	4	3	2	1	0
USARTx_SR: (USARTx->SR)	Reserved	CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE

Field	Bit	Description
CTS	D9	When the CTS pin is changed the flag is set by hardware. For more information see the user manual.
LBD	D8	LIB break detection flag (See the user manual)
TXE	D7	Transmit data register Empty 0 = Data is not transferred to the shift register yet. 1 = The Data Register is empty and ready for the next byte.
TC	D6	Transmit Complete flag (The flag is set by hardware. To clear the flag, you can write a zero to it. If you read the USART_SR and then write to the USART_DR register, the flag will be automatically cleared.) 0 = Transmission is in progress (shift register is occupied) 1 = Transmission complete (both shift register and Data Register are empty)

Example: Sending “Hi\n\r” through USART1 with baud rate of 9600bps

```
#include "stm32f10x.h"

void delay_ms(uint16_t t);
void usart1_sendByte(unsigned char c);

int main()
{
  RCC->APB2ENR |= 0xFC | (1<<14); //enable GPIO & USART1 clocks

  //USART1 init.
  GPIOA->ODR |= (1<<10);  //pull-up PA10
  GPIOA->CRH = 0x444448B4; /* RX1=input with pull-up, TX1=alt. func.
output */
  USART1->CR1 = 0x200C;
  USART1->BRR = 7500;        // 72MHz/9600bps = 7500

  while(1)
  {
```

Example: Receiving data through USART2 and sending back with baud rate of 9600bps

```
#include "stm32f10x.h"

void usart2_sendByte(unsigned char c);
uint8_t usart2_recByte(void);

int main()
{
  RCC->APB1ENR |= (1<<17); //enable usart2 clock
  RCC->APB2ENR |= 0xFC; //enable GPIO clocks

  //USART2 init.
  GPIOA->CRL = 0x44448B44; //RX2=in, TX2=alt. func. Output
  GPIOA->ODR |= (1<<3); // pull-up PA3(RX2)
  USART2->CR1 = 0x200C;
  USART2->BRR = 3750; //36MHz/9600 = 3750

  while(1)
  {
```

Example: Receive a byte serially via USART1. If the received data is “H”, make PC13 high; If the received data is “L”, make PC13 low.

```
#include "stm32f10x.h"

uint8_t usart1_recByte(void);

int main()
{
  RCC->APB2ENR |= 0xFC | (1<<14); /* enable GPIO and usart1 clocks */

  /* USART1 init. */
  GPIOA->ODR |= (1<<10);  /* pull-up PA10 */
  GPIOA->CRH = 0x444448B4; /* RX1=input with pull-up, TX1=alt. func output */
  USART1->CR1 = 0x200C;
  USART1->BRR = 7500;             /* 72MHz/9600bps = 7500 */

  GPIOC->CRH = 0x44344444;

  while(1)  {
    uint8_t c = usart1_recByte();
    switch(c)  {
```