

# Universal Asynchronous Receiver Transmitter (UART)

Microcontroller Application and Development  
Sorayut Glomglome

$\pi$

# Outline

1. Universal Asynchronous Receiver/Transmitter
2. Waveform
3. Transmitting & Receiving Mechanism
4. UART Block Diagram
5. RS-232
6. STM32F767 UART
7. Coding
8. STM32F429 UART

# Learning Outcomes

1. Understanding asynchronous serial transmission
2. Using UART to transmitting and receiving data

# Introduction

- UART – Stands for Universal Asynchronous Receiver Transmitter
- USART – Stands for Universal Synchronous Asynchronous Receiver Transmitter
- UART is about a frame format.
- UART needs to work with physical layer e.g. TTL or RS-232 to define signal characteristics.

## When to use UART

- High speed is not required
- An inexpensive communication link between **two** devices
  - Single wire for each direction (plus ground wire)
  - Asynchronous because no clock signal is transmitted
  - Relatively simple hardware

# A Basic of Serial Communication

## Bit Rate

- Number of bits sent every second (bit/sec, bps)

## Buad Rate

- Number of symbols sent every second, where every symbol can represent more than one bit.

## Overhead

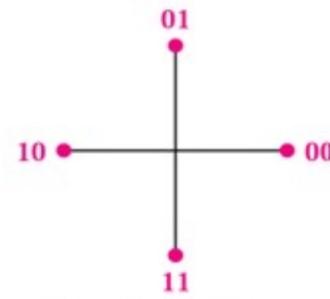
- Additional bits that are needed to sent along with data bits

$\pi$

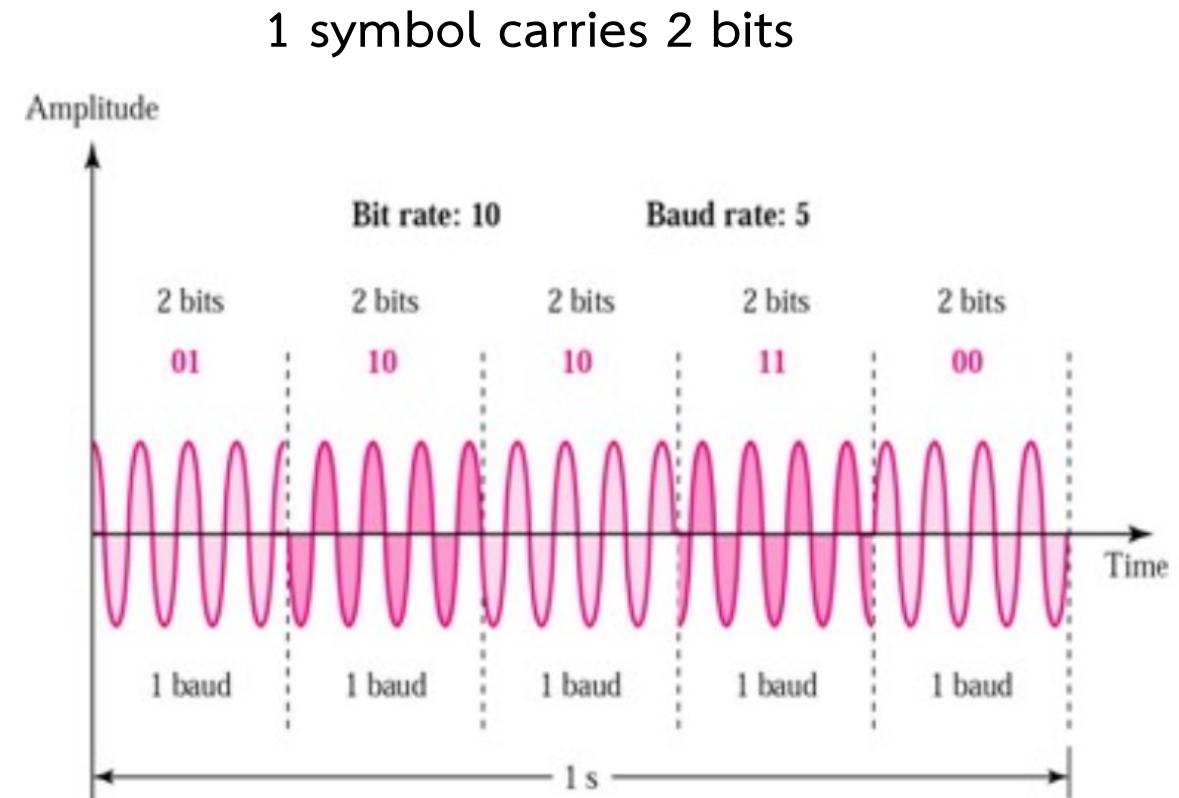
# Quadrature phase-shift keying (QPSK/4-PSK)

Dibit	Phase
00	0
01	90
10	180
11	270

Dibit  
(2 bits)



Constellation diagram

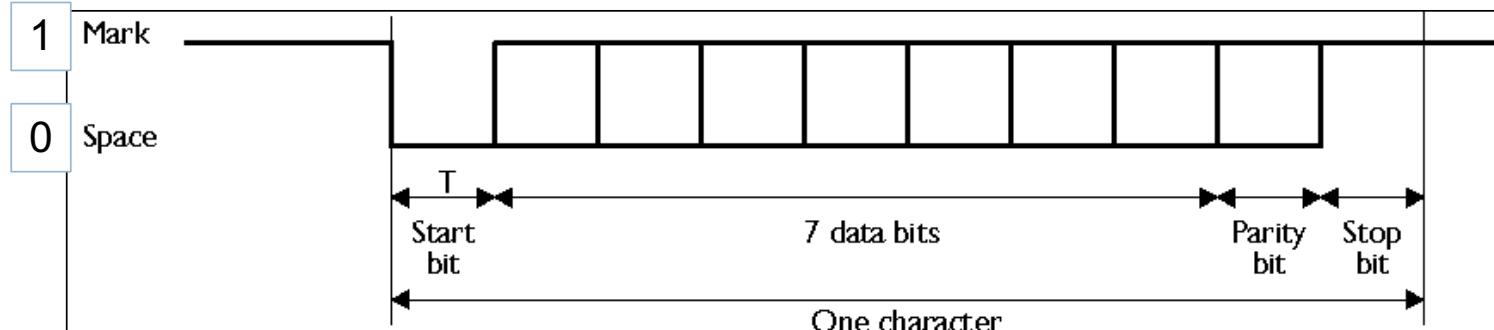


<https://slideplayer.com/slide/4468742/>

# UART (Universal Asynchronous Receiver/Transmitter)

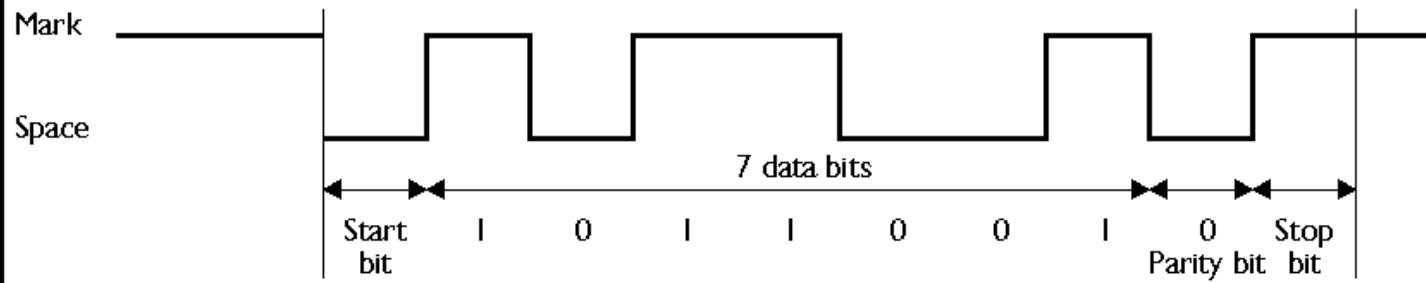
- Point to Point Communication
- Most UARTS are full duplex that allows serial output and serial input to take place simultaneously.
- Based on shift registers and a clock signal.
- UART clock determines baud rate (2400 – 115,200 bits/sec)
- UART frames the data bits with
  - a start bit to provide synchronisation to the receiver
  - one or more (usually one) stop bits to signal end of data
- Most UARTs can also optionally generate PARITY bit to provide error detection.
- UARTs often have receive and transmit buffers (FIFO's) as well as the serial shift registers

# Asynchronous Serial Transmission Waveform



Serial transmission is **little endian** (least significant bit first)

Example: Letter M = 1001101 (even parity)



π

# ASCII Table

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	-	127	7F	[DEL]

# UART Configuration

- Baud rate (bits per second)
  - 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 128000, 256000, ...
- Data bits
  - 5, 6, 7 or 8 bits
- Stop Bits
  - 1, 1.5 or 2 stop bits
- Error Checking
  - Even, odd or no parity bit



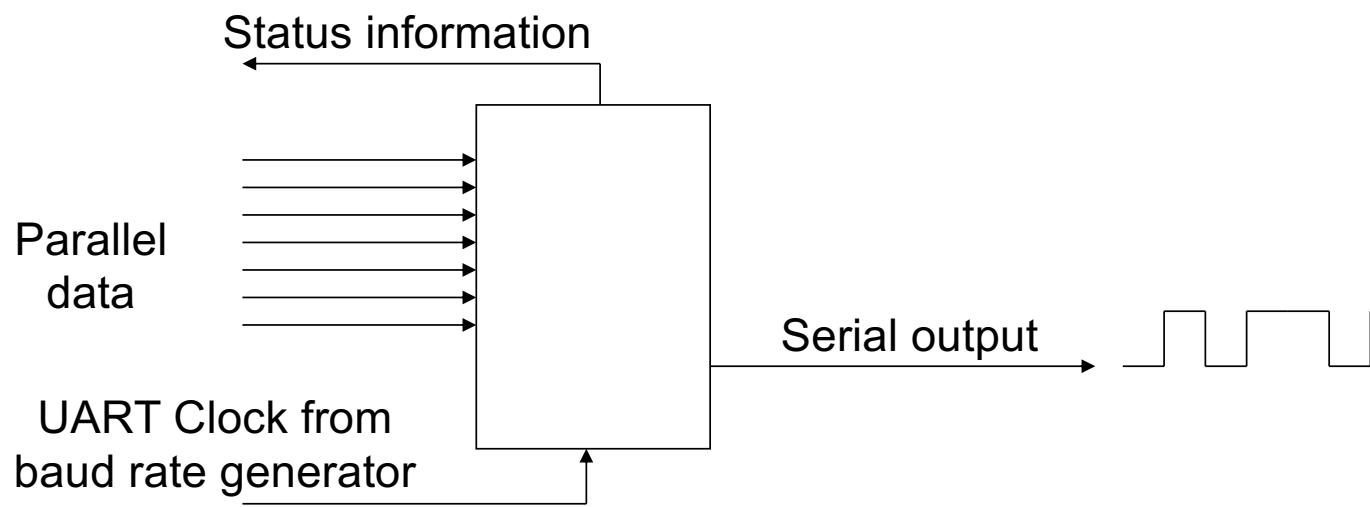
## 9600 8N1 – Most Default Configuration

- 9600 baud, 8 data bits, no parity, and 1 stop bit
- Little endian
- What are the transmitted characters?
- What is the transmission overhead?
- Try different configurations!



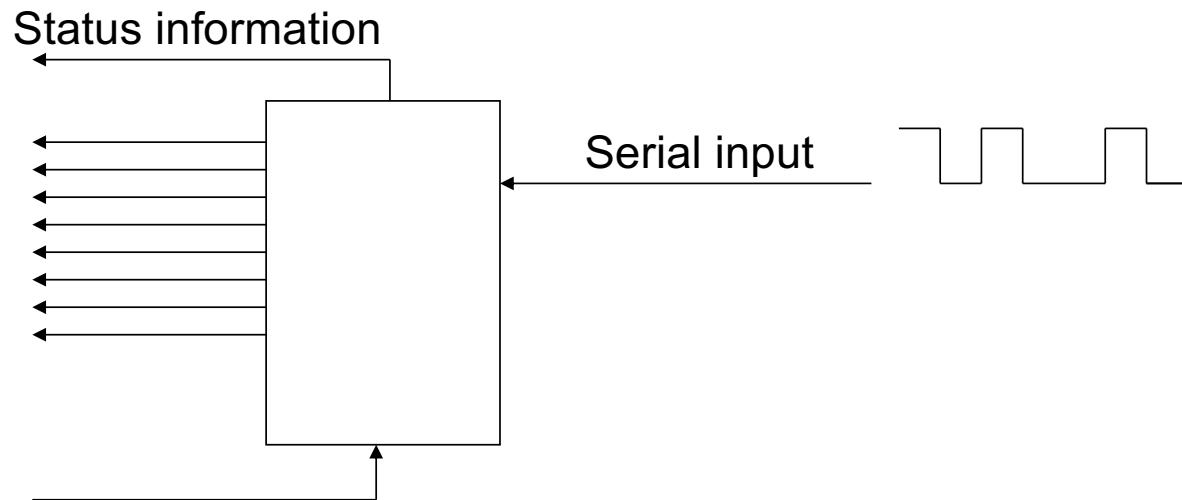
# UART - Transmitter

- Transmitter (Tx) - converts data from parallel to serial format
  - inserts start and stop bits
  - calculates and inserts parity bit if required
  - output bit rate is determined by the UART clock

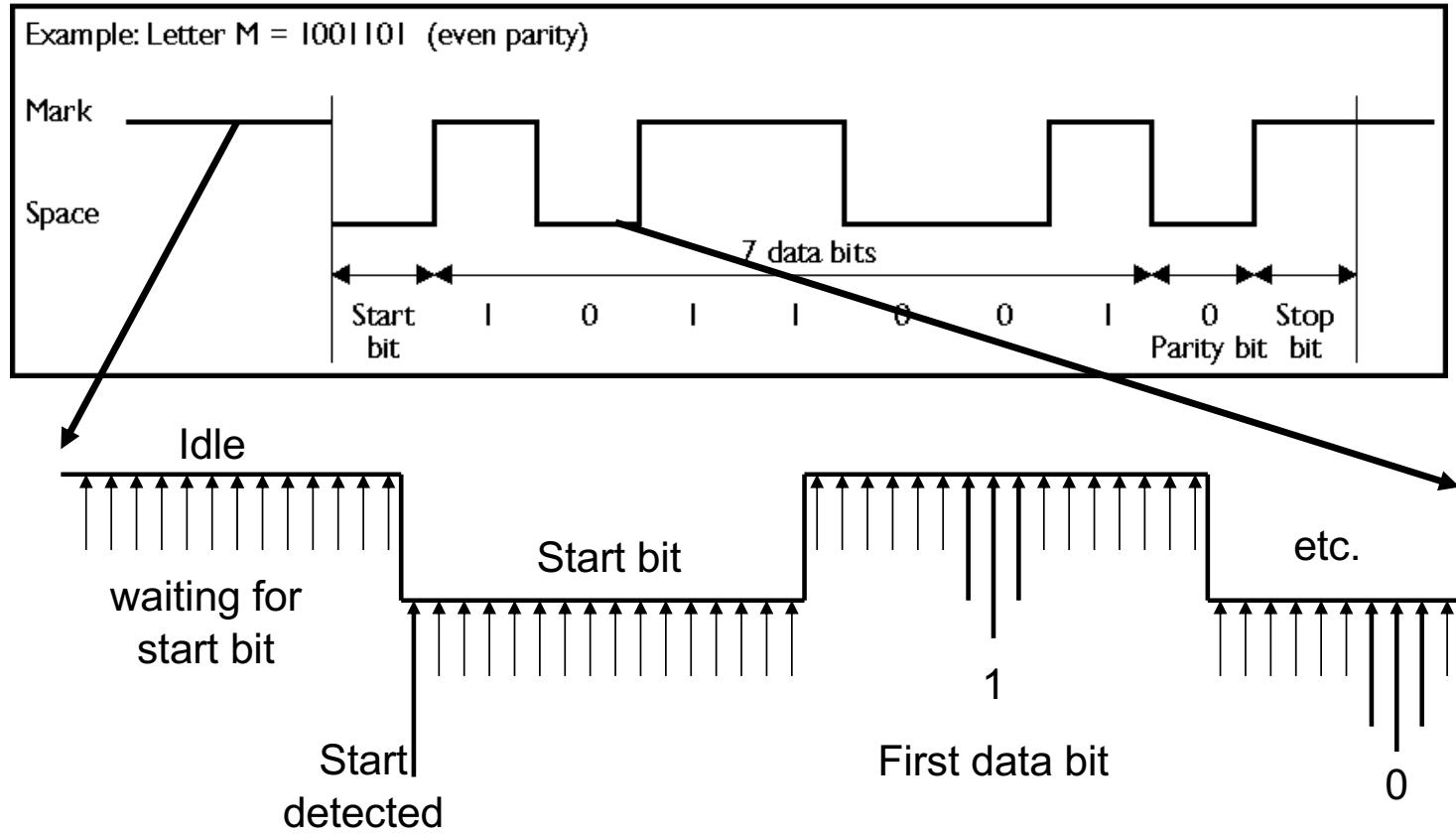


# UART - The Receiver

- Synchronises with transmitter using the falling edge of the start bit.
- Samples the input data line at a clock rate that is normally a multiple of baud rate, typically 16 times the baud rate.
- Removes the start and stop bits, optional calculates and checks the parity bit.
- Presents the received data value in parallel form.

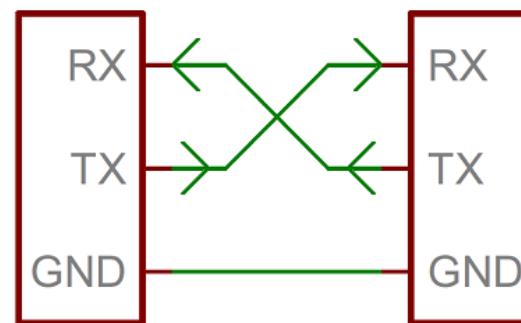
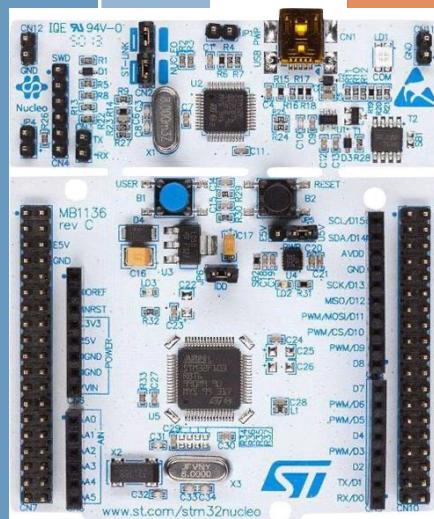


# Asynchronous serial reception



$\pi$

# UART Application 1



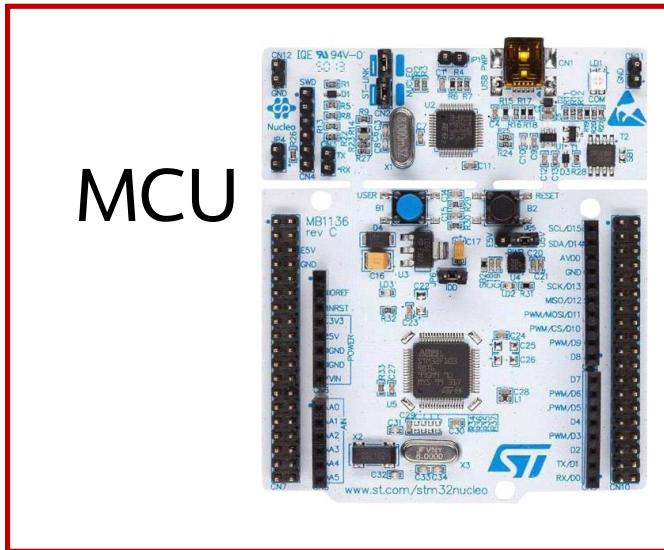
MCU 2



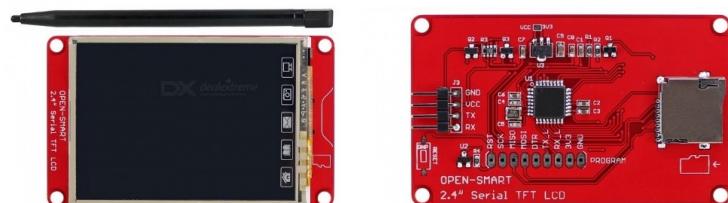
- Signal Level?
- Work between different MCU Models.



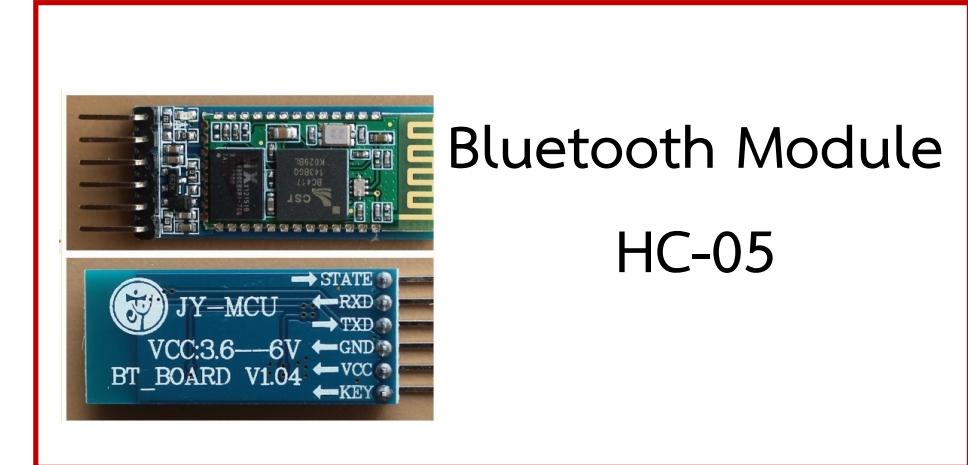
# UART Application 2



# MCU

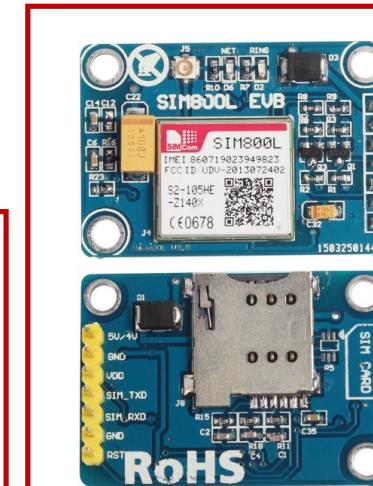


2.4" UART TFT LCD with Touch Sensor



# Bluetooth Module

## HC-05



# GPRS Module

## SIM800L

## UART Application 3

- Interfacing with PC
- Debugging by sending variable value
- Both need additional software to display the received data

# UART over RS232

$\pi$

19

# The Open Systems Interconnection (OSI) Reference Model

## 7 Layers of the OSI Model

### Application

- End User layer
- HTTP, FTP, IRC, SSH, DNS

### Presentation

- Syntax layer
- SSL, SSH, IMAP, FTP, MPEG, JPEG

### Session

- Synch & send to port
- API's, Sockets, WinSock

### Transport

- End-to-end connections
- TCP, UDP

### Network

- Packets
- IP, ICMP, IPSec, IGMP

### Data Link

- Frames
- Ethernet, PPP, Switch, Bridge

### Physical

- Physical structure
- Coax, Fiber, Wireless, Hubs, Repeaters

L2: UART

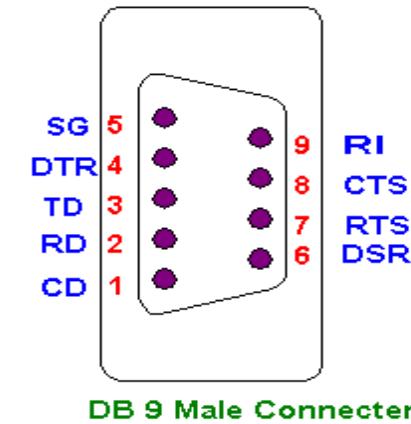
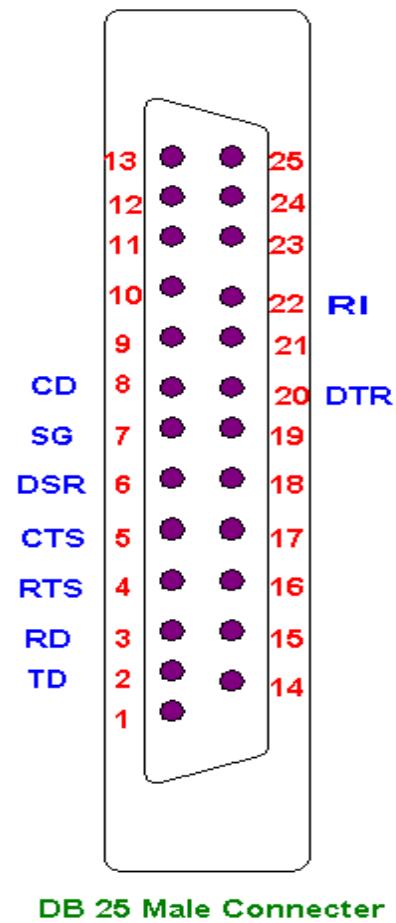
L1: TTL / RS232

# EIA RS232C Serial Interface Standard

- Electronic Industries Association
- A “Space” (logic 0) will be between 3 and 25 volts.
- A “Mark” (logic 1) will be between -3 and -25 volts.
- The region between 3 & -3 volts is undefined.
- Maximum data rates may be up to 20 kbps.
- Maximum serial cable length may be 15 meters.
- The reason to study RS-232C is that the serial part (Com port) found in PC'S uses this standard.

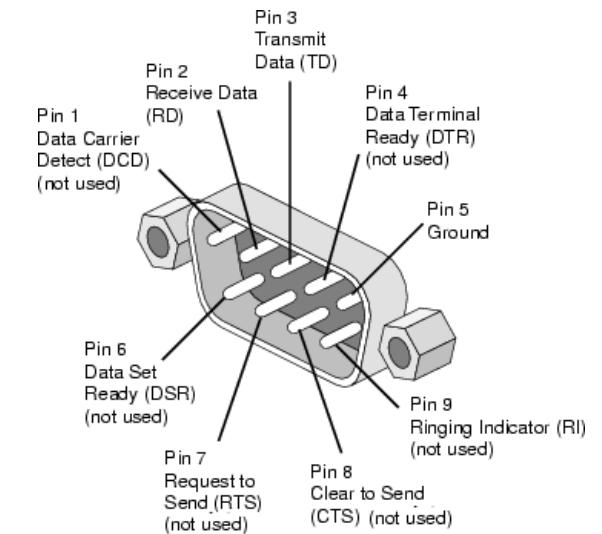
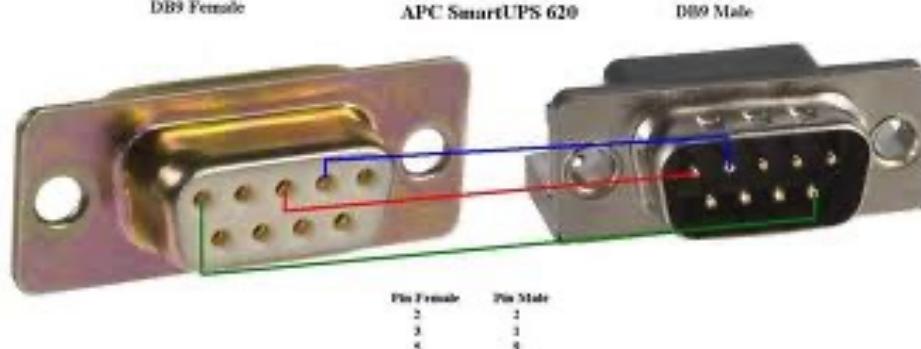
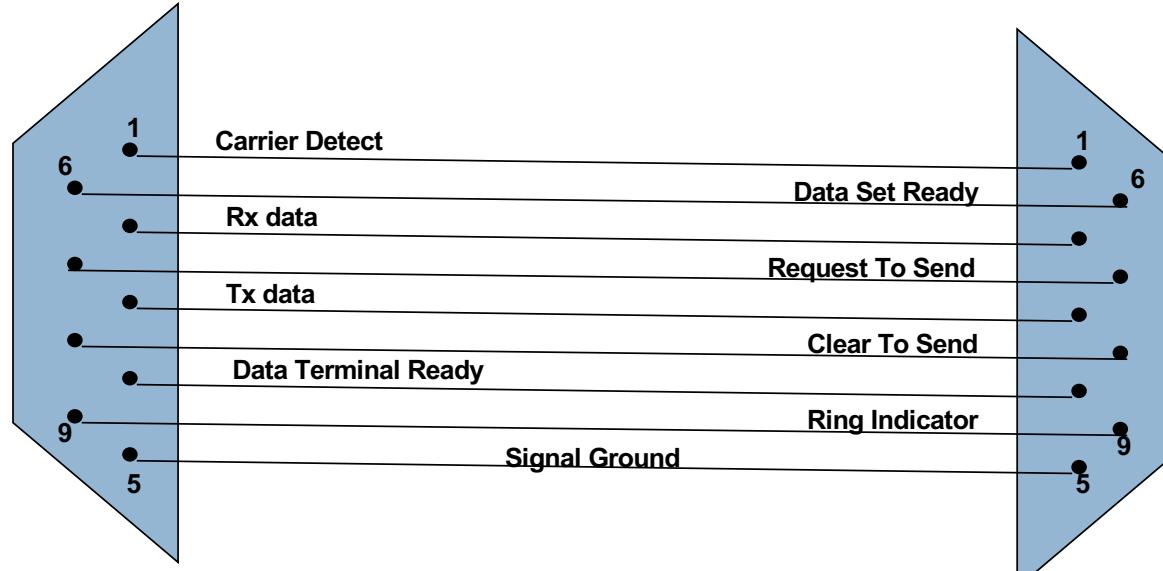
$\pi$

# RS-232 Connectors



$\pi$

# DB-9 Pin Assignment

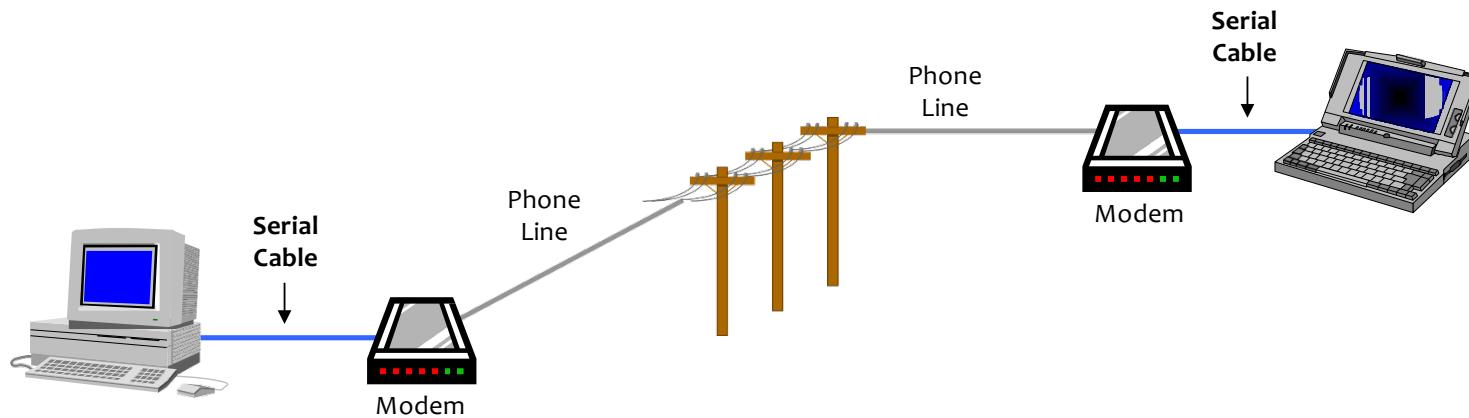


# Pin Functions

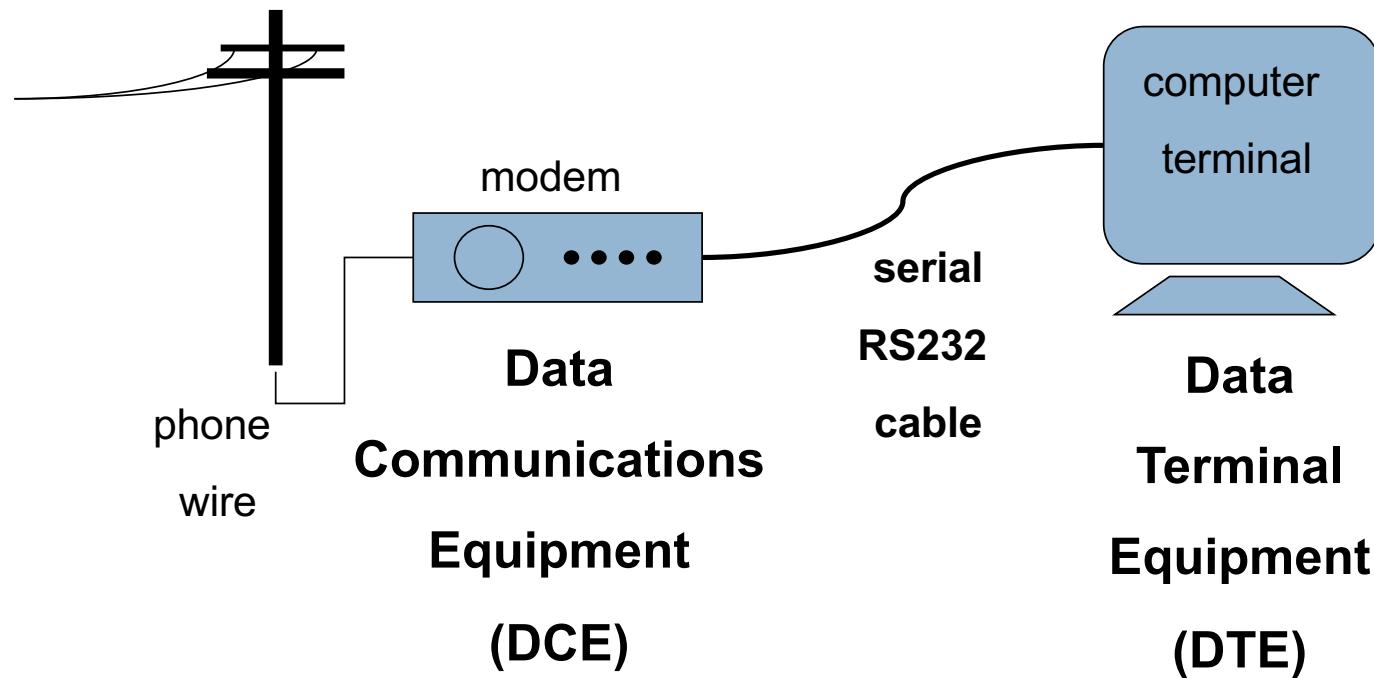
Pin No.	Signal	Name	Functions
1	CD	Carrier Detect	It is used by Modem to inform PC that it has detected Carrier on Phone Line.
2	RD	Received Data	Serial data is received on this line by PC.
3	TD	Transmit Data	Serial Data is transmitted on this pin by PC.
4	DTR	Data Terminal Ready	When terminal (computer) powers up it asserts DTR high.
5	SG	Signal Ground	It is signal ground with reference to which voltages are interpreted as high or low.
6	DSR	Data Set Ready	When modem powers up it asserts DSR high.
7	RTS	Request to Send	Request to send is sent from (DTE) terminal (PC) to modem (DCE) to inform it that PC wants to send some data to modem.
8	CTS	Clear To Send	Upon received RTS from DTE (PC), the modem (DCE) asserts CTS high whenever it is ready to receive data.
9	RI	Ring Indicator	It is set by modem to indicate the PC that a ringing signal has been detected on line.

# RS232 Application 1

- A PC with a modem

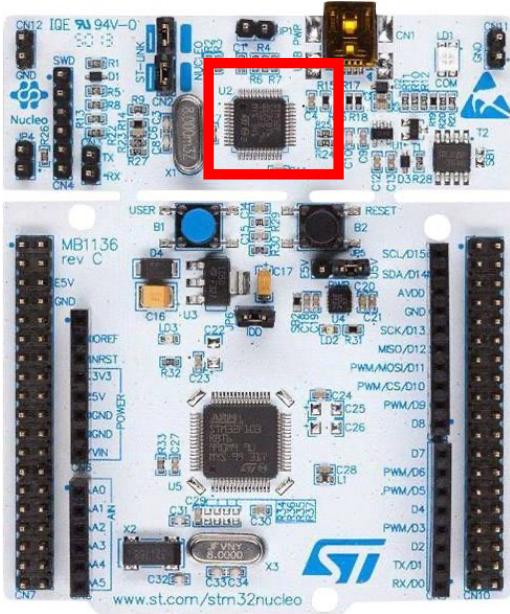


## DCE/DTE



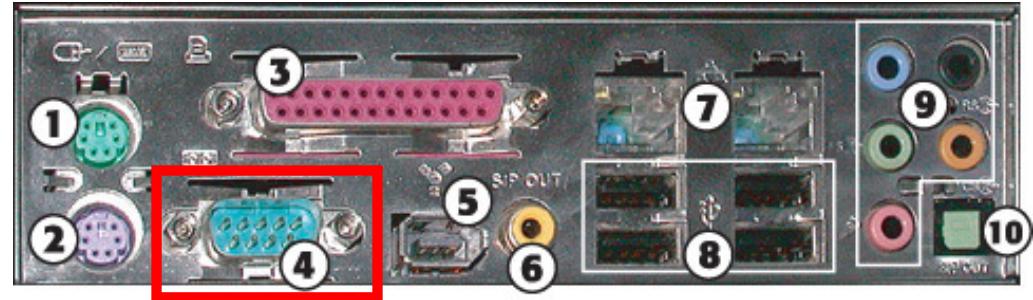
π

# COM Port Legacy



## ST-Link

- In circuit debugging and programming
- Virtual COM Port
- Mass Storage



1. PS/2 mouse port
2. PS/2 keyboard port
3. Parallel (LPT) port
4. Serial (COM) port
5. FireWire 400 (IEEE-1394a) port

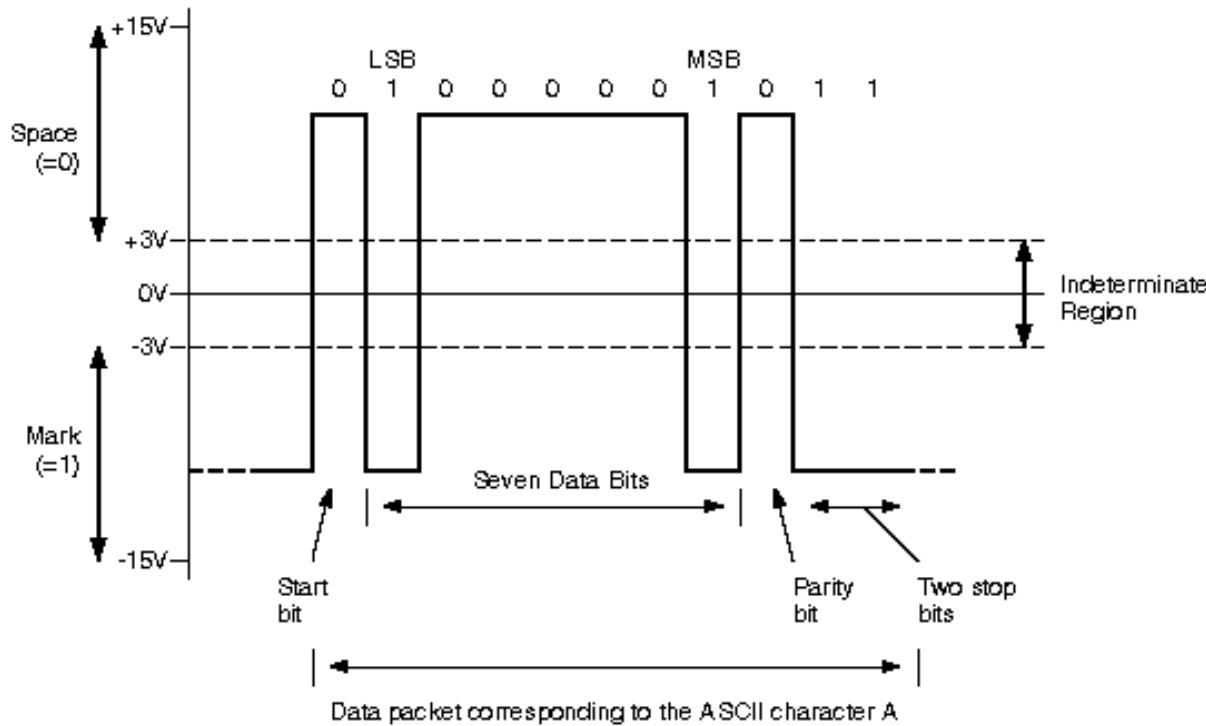
6. Coaxial SPDIF (digital audio) port
7. RJ-45 Ethernet port
8. Hi-Speed USB port
9. 5.1 surround audio ports
10. Fiber Optic SPDIF port



## USB to Serial

# RS232 Signal Voltage Levels

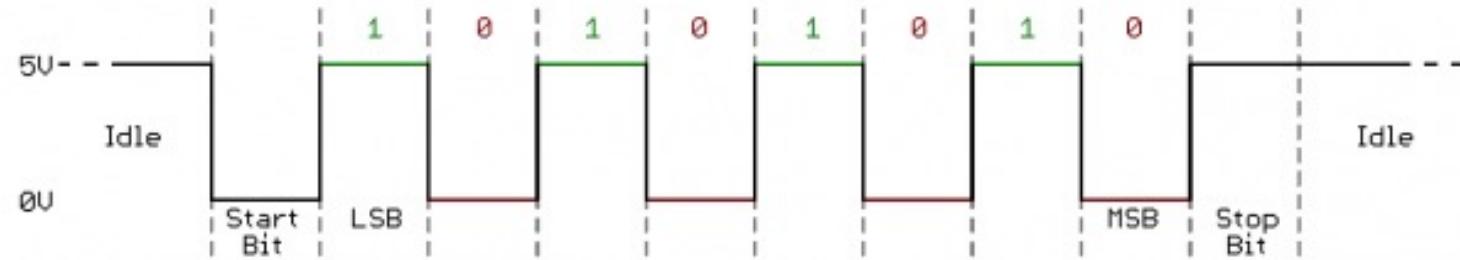
## (7 data bits, Even Parity, 2 stop bits)



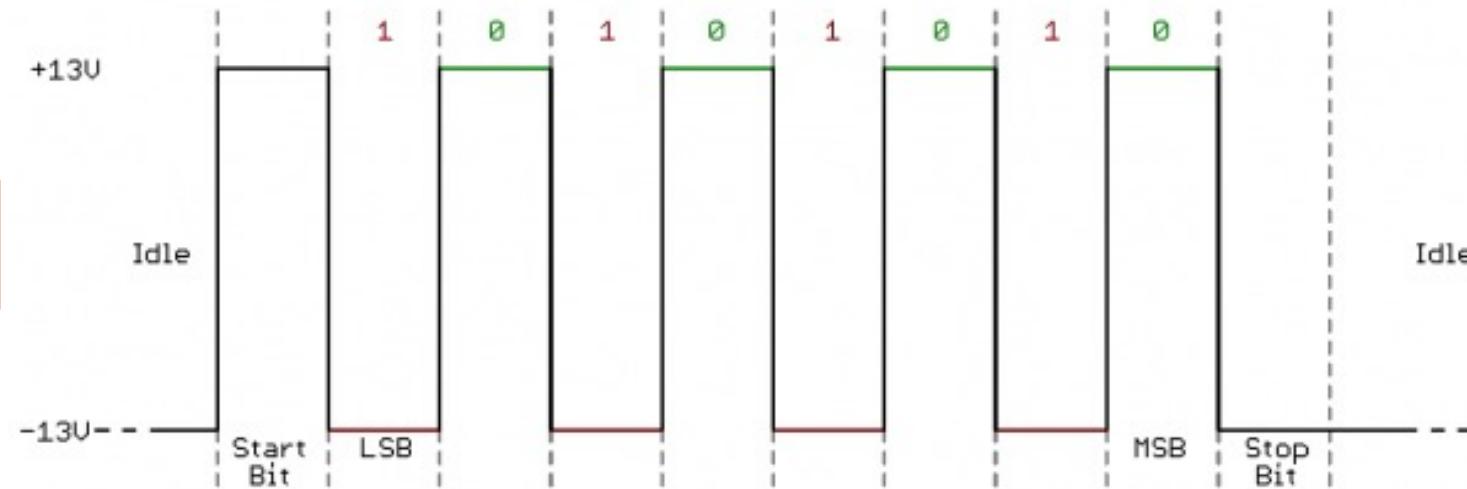
$\pi$

# UART TTL&RS232 Signal Voltage Level

UART TTL

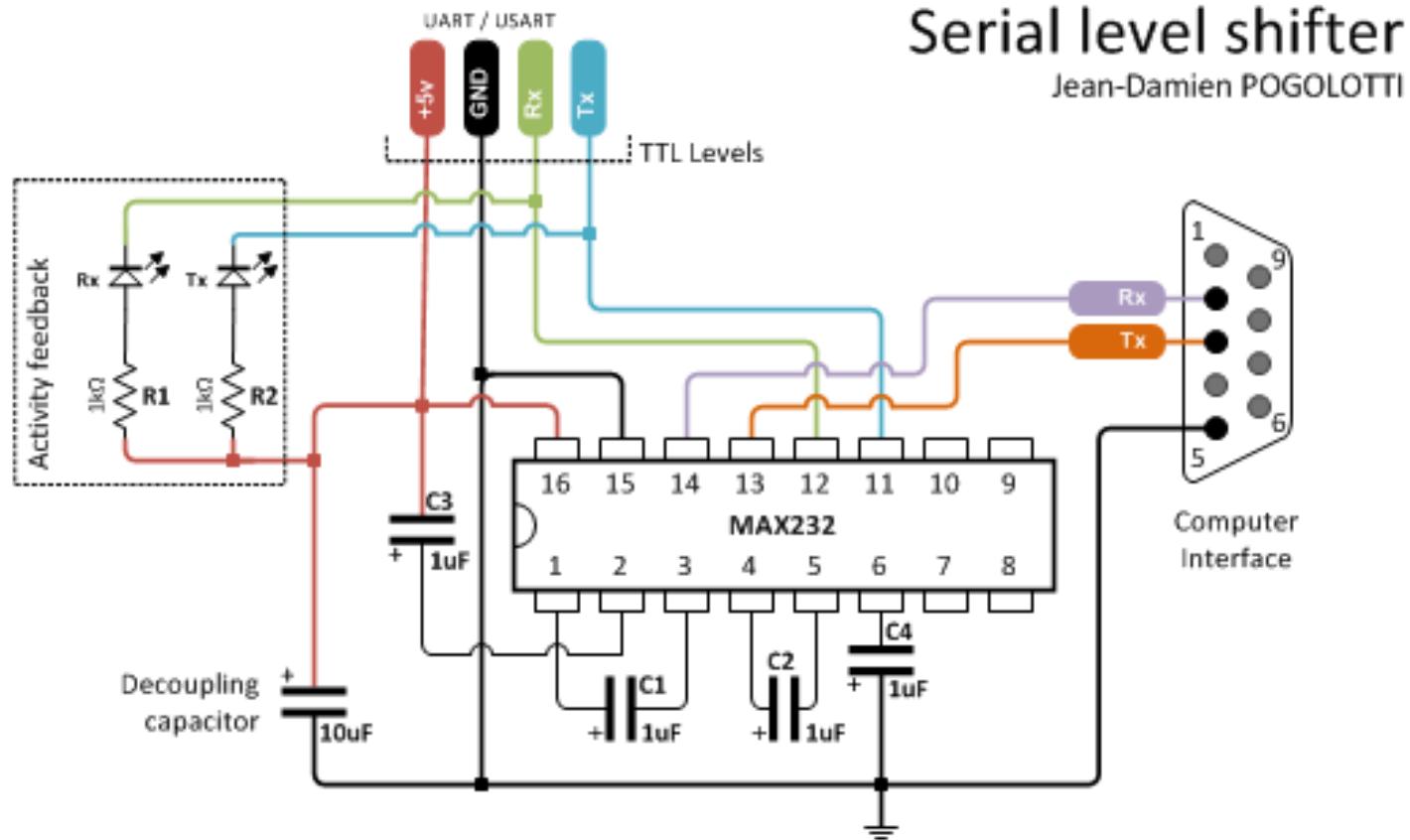


UART RS232



$\pi$

# MAX232 IC : UART <-> RS-232 Conversion



# UART Modules on STM32F767

$\pi$

31

# UARTs on STM32F767

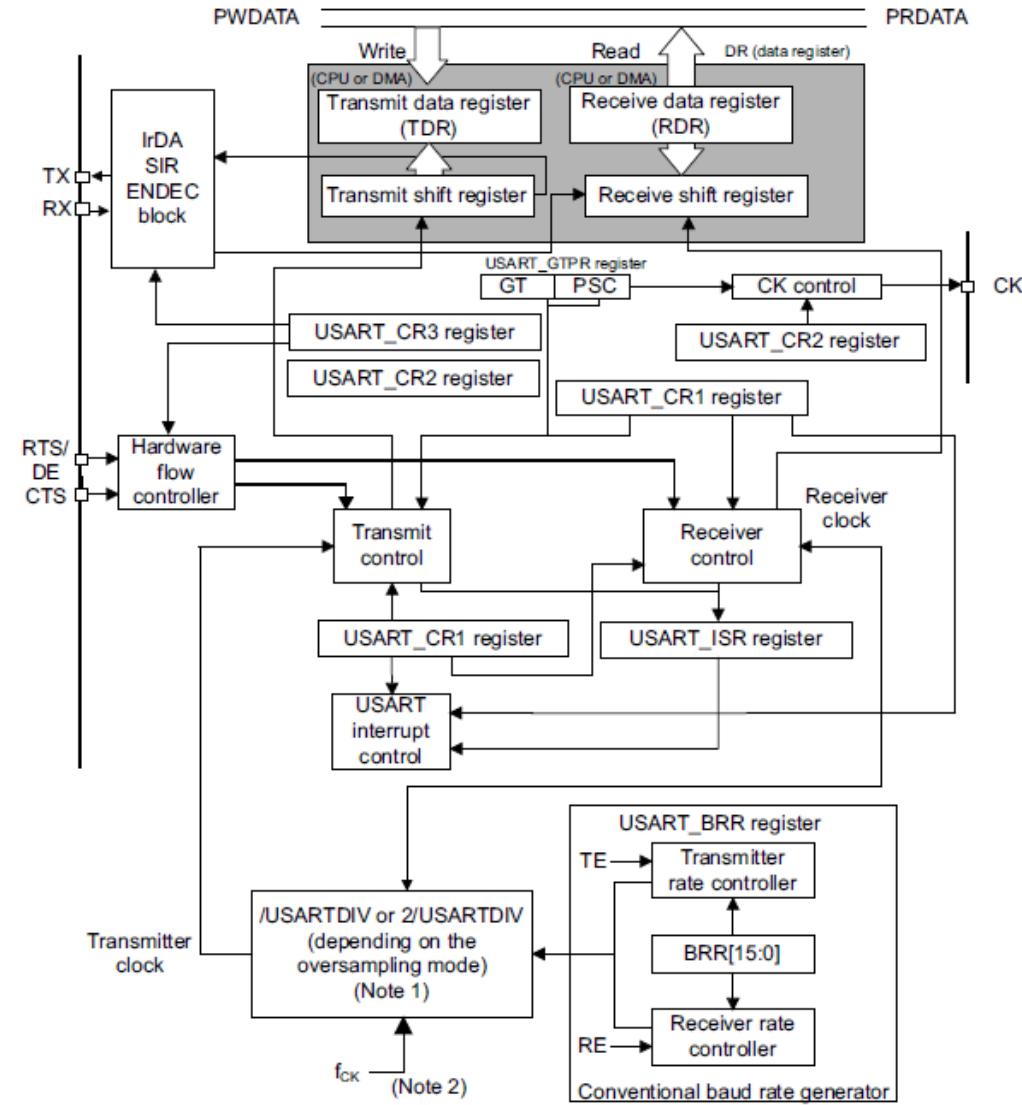
- Full duplex, asynchronous communications
- Programmable data word length (7/8/9 bits)
- Configurable stop bits (1/1.5/2 stop bits)
- Single-wire half duplex communications
- Programmable data order with MSB-first or LSB-first shifting
- Programmable parity (odd, even, no parity)
- Synchronous mode and clock output for synchronous communications

Table 8. USART implementation

features <sup>(1)</sup>	USART1/2/3/6	UART4/5/7/8
Data Length	7, 8 and 9 bits	
Hardware flow control for modem	X	X
Continuous communication using DMA	X	X
Multiprocessor communication	X	X
Synchronous mode	X	-

$\pi$

# UART Block Diagram





# UART Registers and Memory Map

No.	Register Name	Macro	Offset
1	<b>Control register 1</b>	USART_CR1	0x00
2	Control register 2	USART_CR2	0x04
3	Control register 3	USART_CR3	0x08
4	Baud rate register	USART_BRR	0x0C
5	Guard time and prescaler register	USART_GTPR	0x10
6	Receiver timeout register	USART_RTOR	0x14
7	Request register	USART_RQR	0x18
8	<b>Interrupt and status register</b>	USART_ISR	0x1C
9	Interrupt flag clear register	USART_ICR	0x20
10	<b>Receive data register</b>	USART_RDR	0x24
11	<b>Transmit data register</b>	USART_TDR	0x28

Bus	Starting Address	UART No.
APB2	0x4001 1400	USART6
	0x4001 1000	USART1
APB1	0x4000 7C00	UART8
	0x4000 7800	UART7
	0x4000 5000	UART5
	0x4000 4C00	UART4
	0x4000 4800	USART3
	0x4000 4400	USART2

### 34.8.1 Control register 1 (USART\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	M1	EOBIE	RTOIE			DEAT[4:0]					DEDT[4:0]		
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

#### Bit 28 M1: Word length

This bit, with bit 12 (M0), determines the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 data bits, n stop bits

M[1:0] = 10: 1 Start bit, 7 data bits, n stop bits

This bit can only be written when the USART is disabled (UE=0).

#### Bit 12 M0: Word length

This bit, with bit 28 (M1), determines the word length. It is set or cleared by software. See Bit 28 (M1) description.

This bit can only be written when the USART is disabled (UE=0).

#### Bit 10 PCE: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bit field can only be written when the USART is disabled (UE=0).

#### Bit 15 OVER8: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE=0).

#### Bit 9 PS: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity will be selected after the current byte.

0: Even parity

1: Odd parity

This bit field can only be written when the USART is disabled (UE=0).

### 34.8.2 Control register 2 (USART\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD[1:0]		ABREN	MSBF RST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw				

#### Bits 13:12 STOP[1:0]: STOP bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit

10: 2 stop bits

11: 1.5 stop bits

This bit field can only be written when the USART is disabled (UE=0).

$\pi$

### 34.8.8 Interrupt and status register (USART\_ISR)

Address offset: 0x1C

Reset value: 0x0200 00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
						r			r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	Res.	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
r	r		r	r	r	r	r	r	r	r	r	r	r	r	r

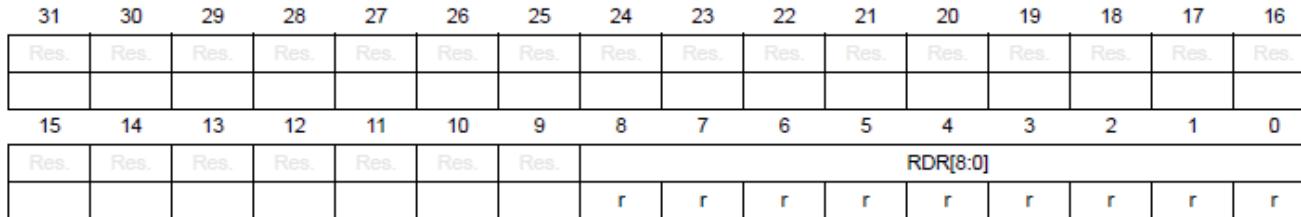
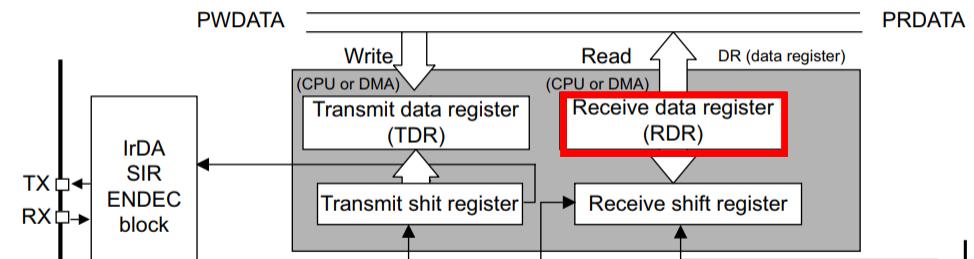
Bit	Flag	Set by	Reset by	Logic 0	Logic 1
16	<b>BUSY:</b> Busy flag	HW	HW	USART is idle	Reception on going
7	<b>TXE:</b> Transmit data register empty	HW	SW	data is not transferred to the shift register	data is transferred to the shift register)
6	<b>TC:</b> Transmission complete	HW	write to USART_TDR	Transmission is not complete	Transmission is complete
5	<b>RXNE:</b> Read data register not empty	HW	Read USART_RDR	data is not received	Received data is ready to be read.
4	<b>IDLE:</b> Idle line detected	HW	SW	No Idle line is detected	Idle line is detected
2	<b>NF:</b> START bit Noise detection flag	HW	SW	No noise is detected	Noise is detected
0	<b>PE:</b> Parity error	HW	SW	No parity error	Parity error



### 34.8.10 Receive data register (USART\_RDR)

Address offset: 0x24

Reset value: 0x0000 0000



Bits	Name	Description
31:9	Reserved	Must be kept at reset value.
8:0	RDR[8:0]: Receive data value	<p>Contains the received data character.</p> <p>The RDR register provides the parallel interface between the input shift register and the internal bus.</p> <p>When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.</p>

### 34.8.11 Transmit data register (USART\_TDR)

Address offset: 0x28

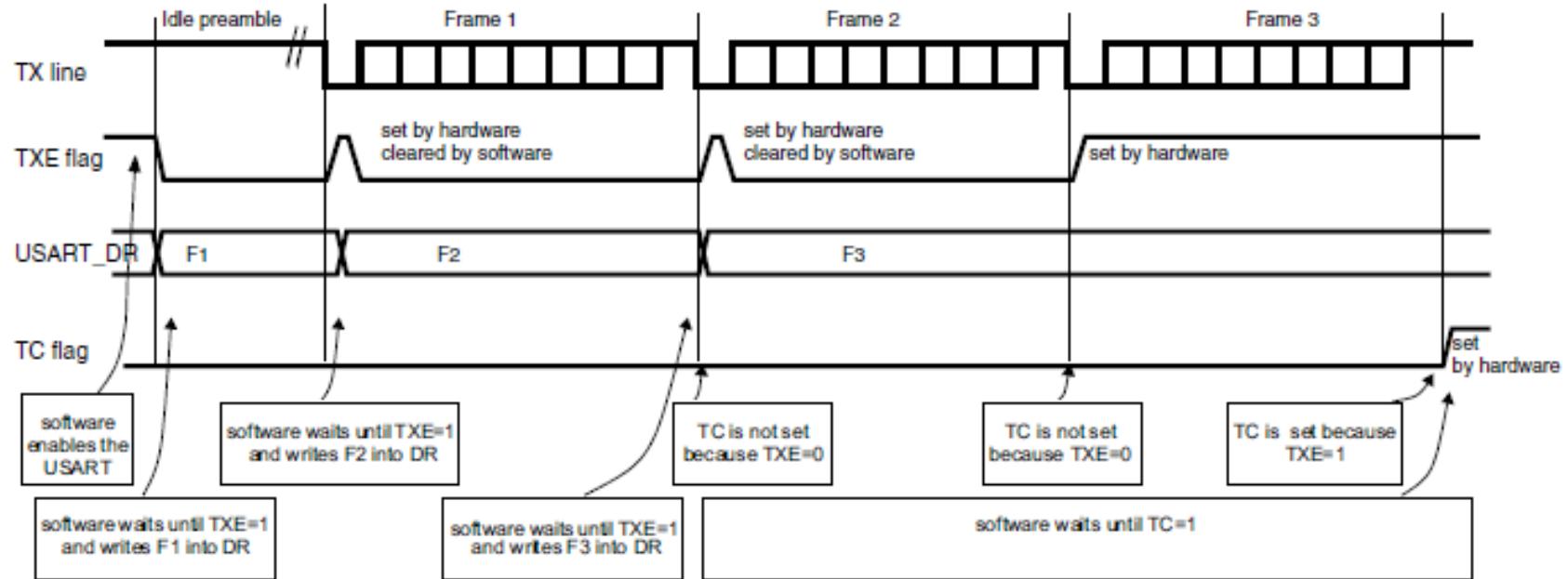
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDR[8:0]														
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Name	Description
31:9	Reserved	Must be kept at reset value.
8:0	TDR[8:0]: Transmit data value	<p>Contains the data character to be transmitted.</p> <p>The TDR register provides the parallel interface between the internal bus and the output shift register.</p> <p>When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.</p> <p><i>Note: This register must be written only when TXE=1</i></p>

$\pi$

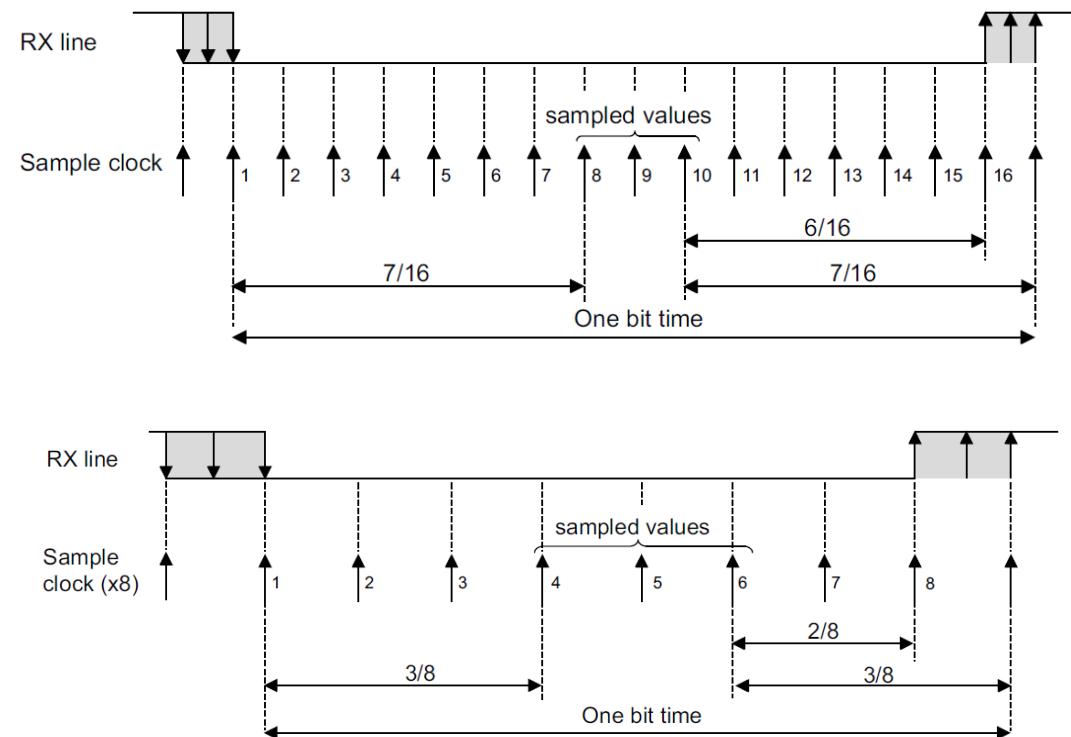
# Transmitter



$\pi$

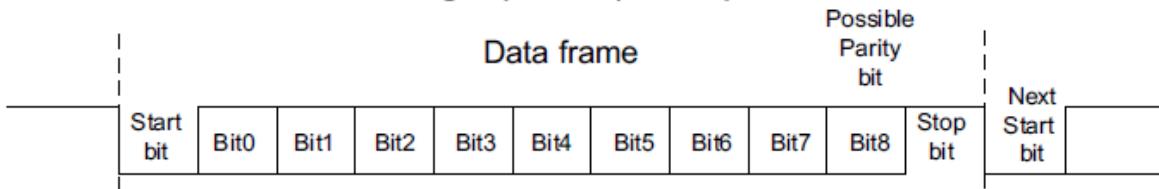
# Receiver

Sampled value	NE status	Received bit value	Data validity
000	0	0	Valid
001	1	0	Not Valid
010	1	0	Not Valid
011	1	1	Not Valid
100	1	0	Not Valid
101	1	1	Not Valid
110	1	1	Not Valid
111	0	1	Valid

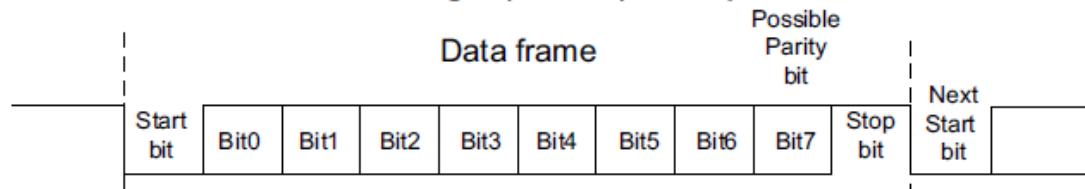


# Word Length Programming

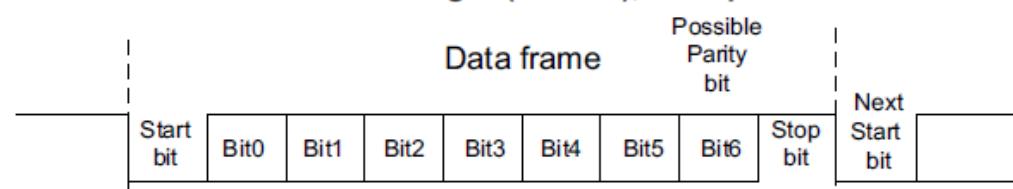
9-bit word length ( $M = 01$ ), 1 Stop bit



8-bit word length ( $M = 00$ ), 1 Stop bit



7-bit word length ( $M = 10$ ), 1 Stop bit

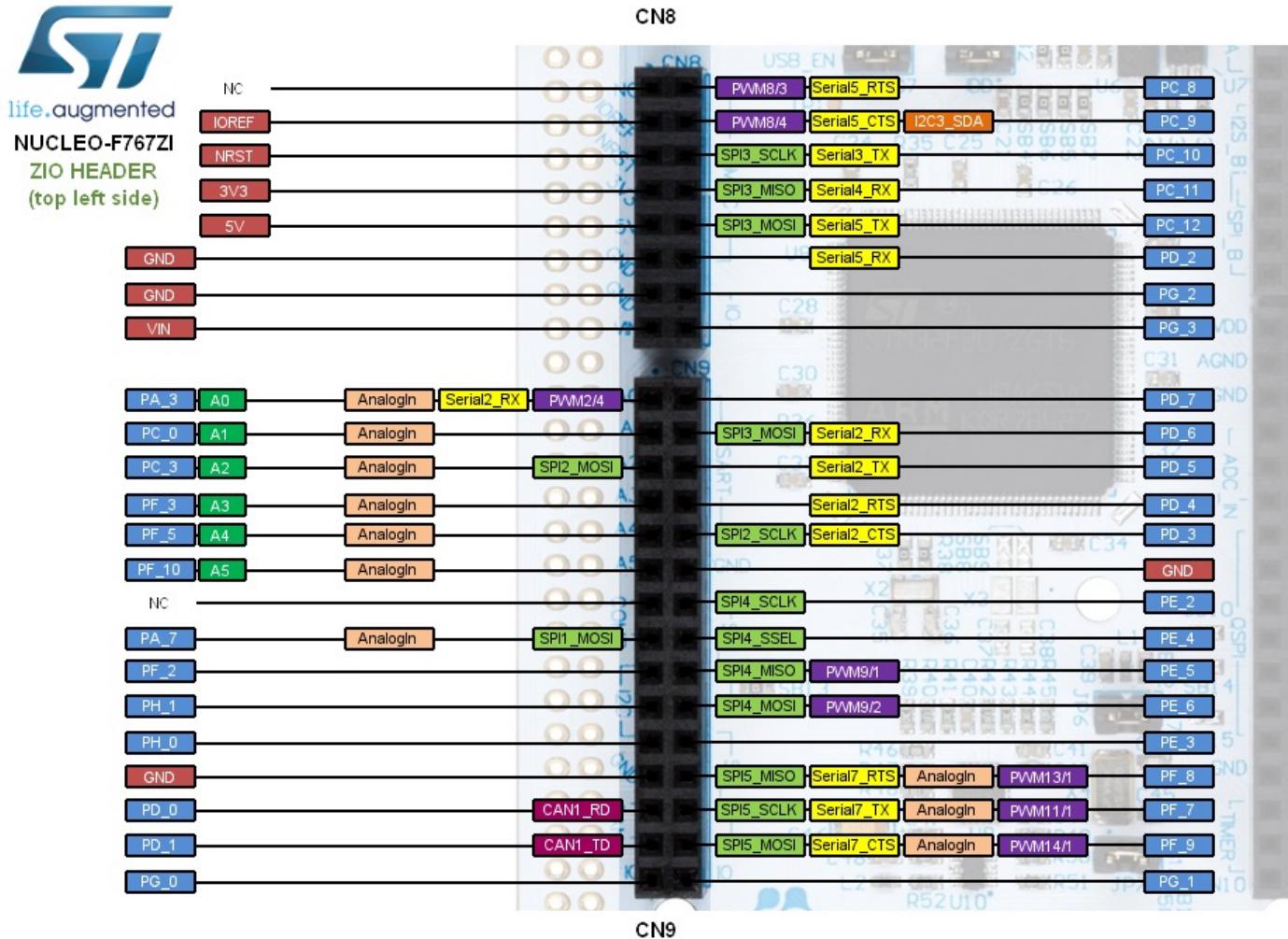


$\pi$

# PIN Definition : MCU Datasheet

Pin No.	Main Function (After Reset)	Alternate Functions	Additional Function
18	PF6	TIM10_CH1, SPI5_NSS, SAI1_SD_B, <b>UART7_RX</b> , QUADSPI_BK1_IO3, EVENTOUT	ADC3_IN4
19	PF7	TIM11_CH1, SPI5_SCK, SAI1_MCLK_B, <b>UART7_TX</b> , QUADSPI_BK1_IO2, EVENTOUT	ADC3_IN5
34	PA0- WKUP	TIM2_CH1/TIM2_ETR, TIM5_CH1, TIM8_ETR, USART2_CTS, <b>UART4_TX</b> , SAI2_SD_B, ETH_MII_CRS, EVENTOUT	ADC1_IN0, ADC2_IN0, ADC3_IN0, WKUP1
35	PA1	TIM2_CH2, TIM5_CH2, USART2_RTS, <b>UART4_RX</b> , QUADSPI_BK1_IO3, SAI2_MCLK_B, ETH_MII_RX_CLK/ETH_RMII_REF_CLK, LCD_R2, EVENTOUT	ADC1_IN1, ADC2_IN1, ADC3_IN1
77	PD8	DFSDM1_CKIN3, <b>USART3_TX</b> , SPDIF_RX1, FMC_D13, EVENTOUT	-
78	PD9	DFSDM1_DATIN3, <b>USART3_RX</b> , FMC_D14, EVENTOUT	-

$\pi$

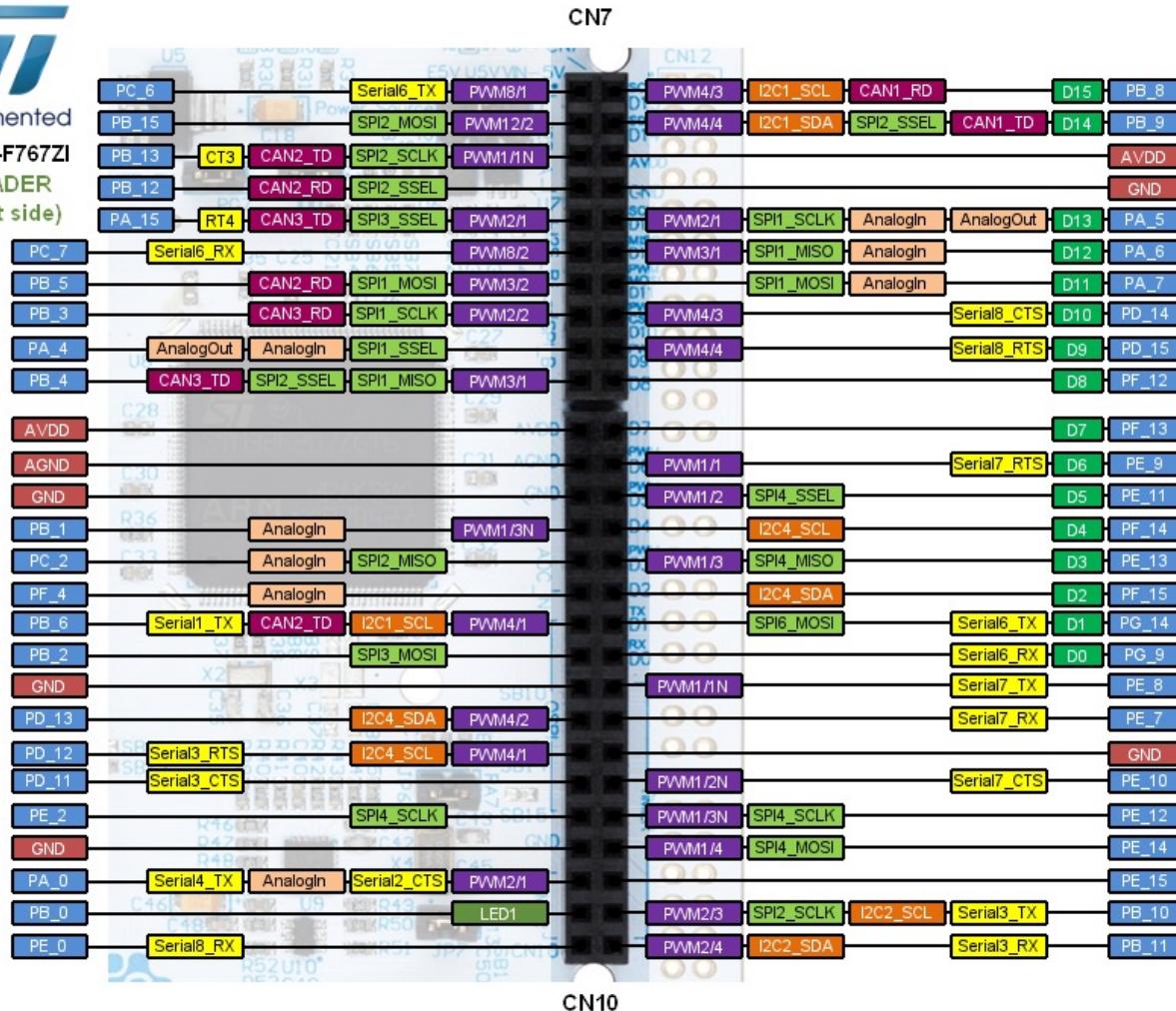




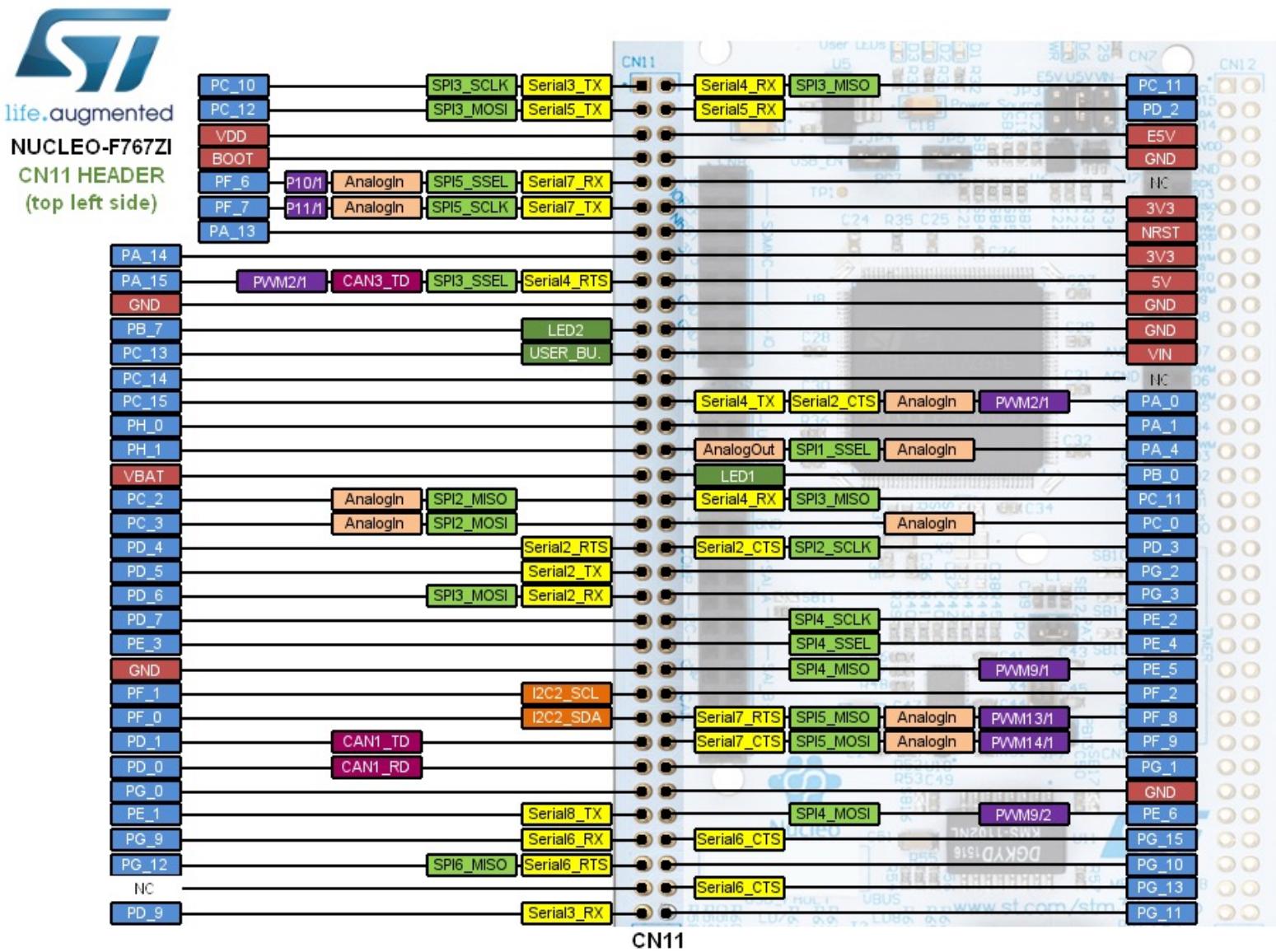
life.augmented

NUCLEO-F767ZI

ZIO HEADER  
(top right side)



$\pi$

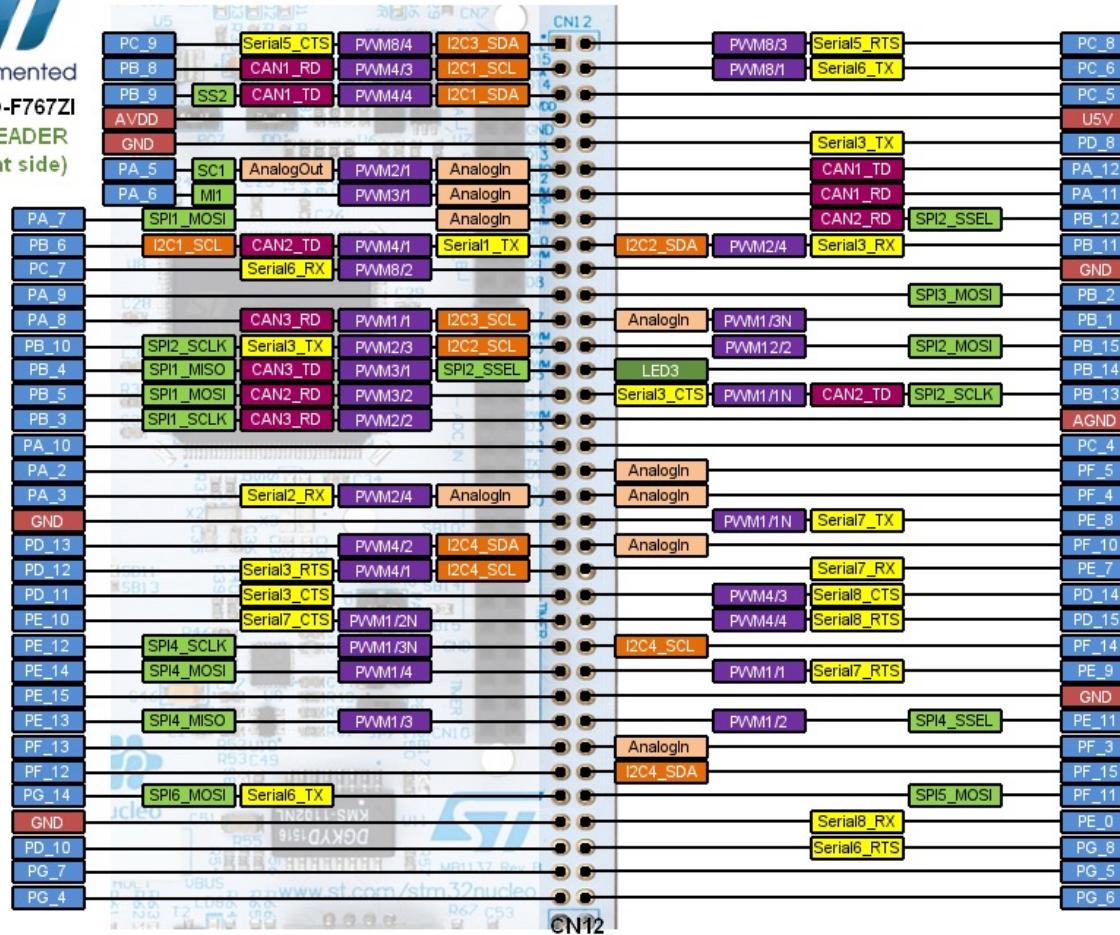


$\pi$

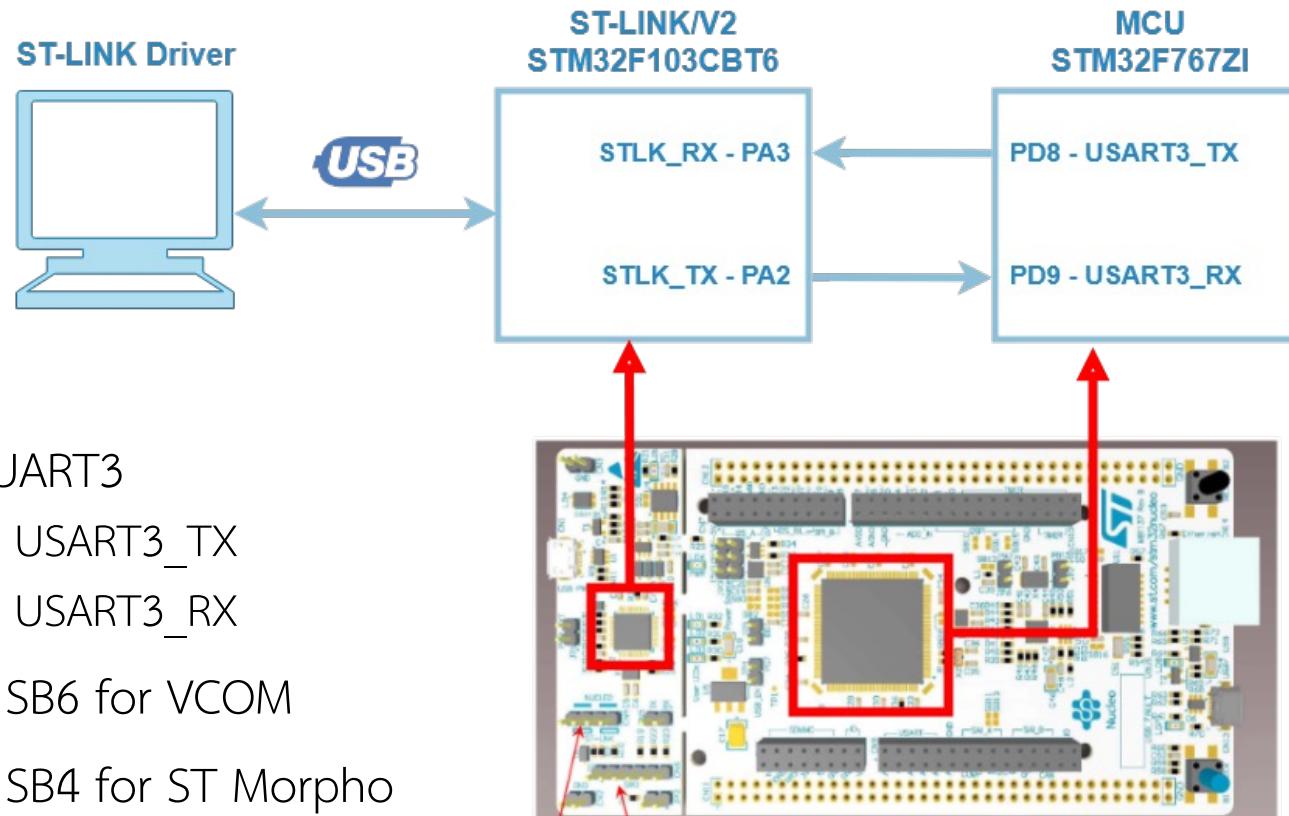


life.augmented

NUCLEO-F767ZI  
CN12 HEADER  
(top right side)

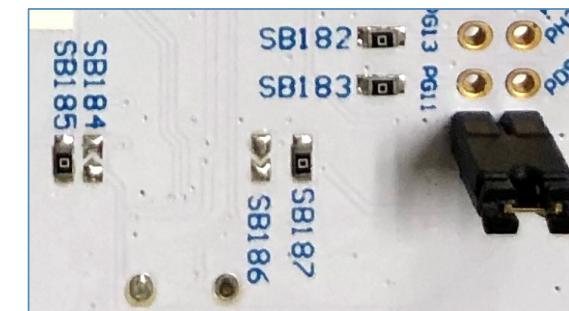
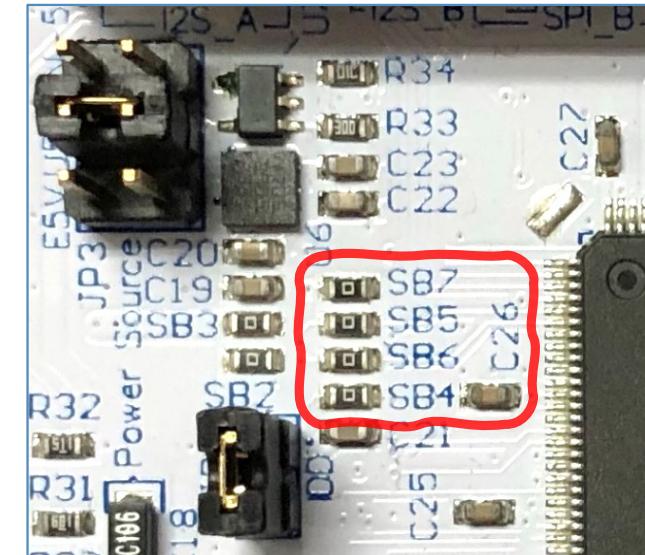
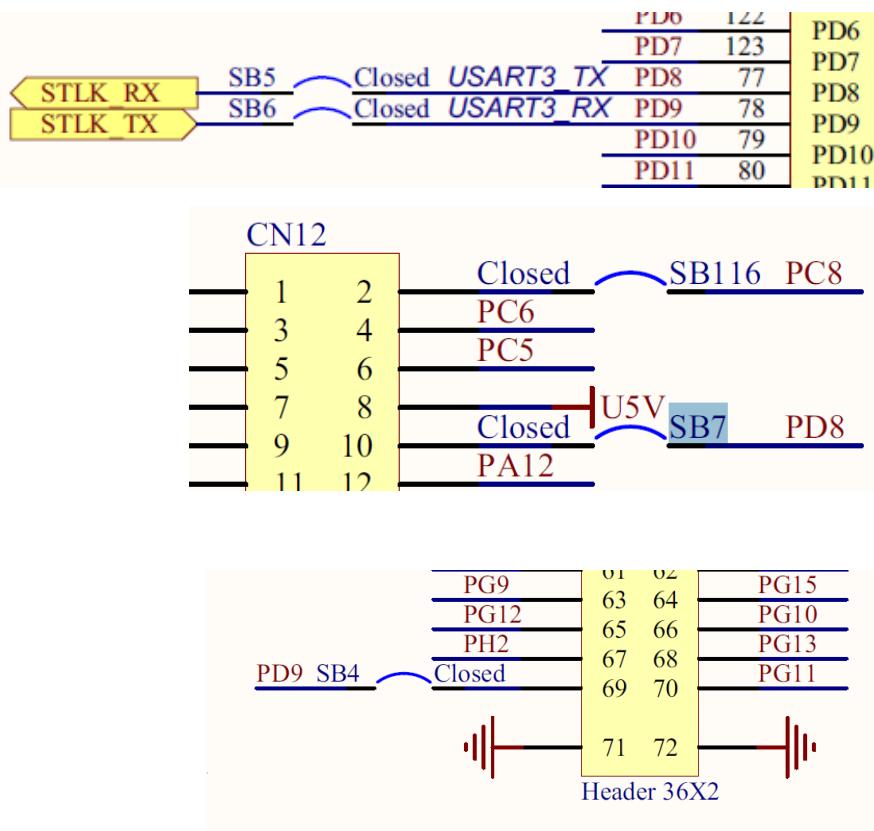


# Virtual Communication Port / ST-Link



π

# SB4, SB5, SB6 and SB7 ON (Solder Bridge)





# UART3 Configuration (1)

Pinout & Configuration

Clock Configuration

Project Manager

1

2

3

4

5

uart.c

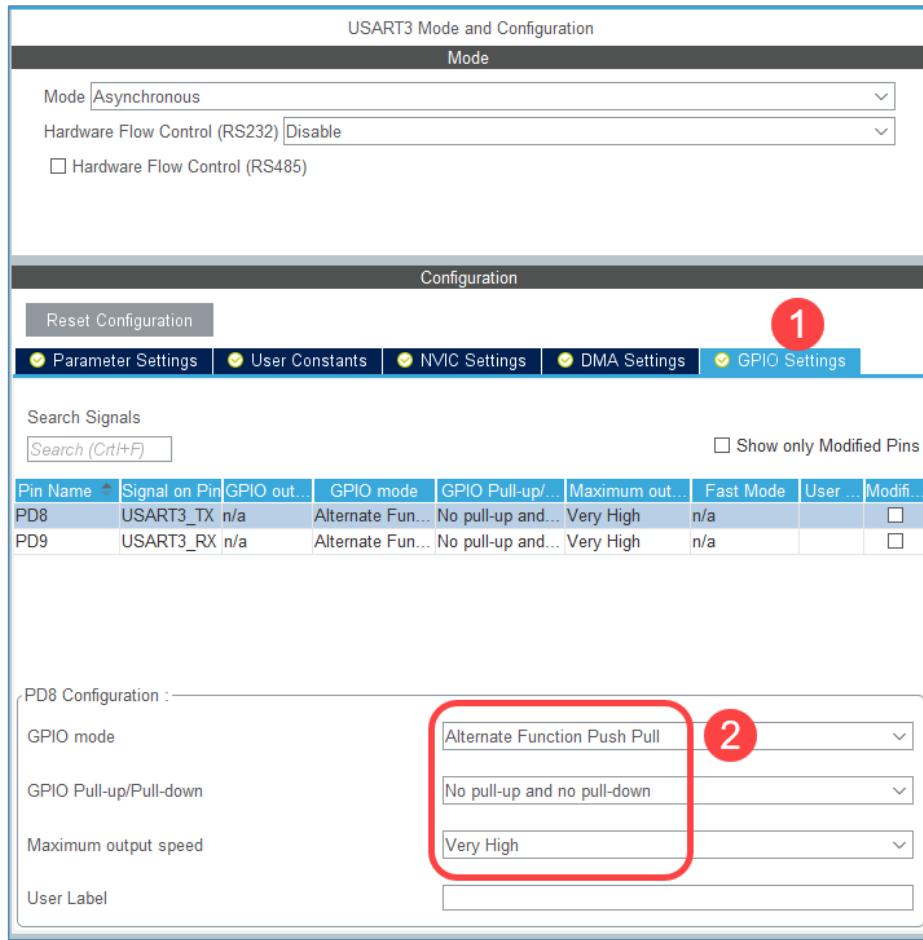
```
UART_HandleTypeDef huart3;

/* USART3 init function */

void MX_USART3_UART_Init(void)
{
    huart3.Instance = USART3;
    huart3.Init.BaudRate = 115200;
    huart3.Init.WordLength = UART_WORDLENGTH_8B;
    huart3.Init.StopBits = UART_STOPBITS_1;
    huart3.Init.Parity = UART_PARITY_NONE;
    huart3.Init.Mode = UART_MODE_TX_RX;
    huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart3.Init.OverSampling = UART_OVERSAMPLING_16;
    huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
    huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
    if (HAL_UART_Init(&huart3) != HAL_OK)
    {
        Error_Handler();
    }
}
```

π

# UART3 Configuration (2)



## uart.c

```
void HAL_UART_MspInit(UART_HandleTypeDef* uartHandle)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    if(uartHandle->Instance==USART3)
    {
        /* USER CODE BEGIN USART3_MspInit 0 */

        /* USER CODE END USART3_MspInit 0 */
        /* USART3 clock enable */
        __HAL_RCC_USART3_CLK_ENABLE();

        __HAL_RCC_GPIOD_CLK_ENABLE();
        /**USART3 GPIO Configuration
        PD8      -----> USART3_TX
        PD9      -----> USART3_RX
        */
        GPIO_InitStruct.Pin = GPIO_PIN_8|GPIO_PIN_9;
        GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
        GPIO_InitStruct.Pull = GPIO_NOPULL;
        GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
        GPIO_InitStruct.Alternate = GPIO_AF7_USART3;
        HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

        /* USER CODE BEGIN USART3_MspInit 1 */

        /* USER CODE END USART3_MspInit 1 */
    }
}
```



## HAL\_UART\_Transmit

Function name      **HAL\_StatusTypeDef HAL\_UART\_Transmit**  
**(UART\_HandleTypeDef \* huart, uint8\_t \* pData, uint16\_t Size,**  
**uint32\_t Timeout)**

Function description      Send an amount of data in blocking mode.

- Parameters
- **huart:** UART handle.
  - **pData:** Pointer to data buffer.
  - **Size:** Amount of data to be sent.
  - **Timeout:** Timeout duration.
- Return values
- **HAL:** status

```
/**  
 * @brief  HAL Status structures definition  
 */  
typedef enum  
{  
    HAL_OK          = 0x00U,  
    HAL_ERROR       = 0x01U,  
    HAL_BUSY        = 0x02U,  
    HAL_TIMEOUT     = 0x03U  
} HAL_StatusTypeDef;
```

## HAL\_UART\_Receive

Function name      **HAL\_StatusTypeDef HAL\_UART\_Receive**  
**(UART\_HandleTypeDef \* huart, uint8\_t \* pData, uint16\_t Size,**  
**uint32\_t Timeout)**

Function description      Receive an amount of data in blocking mode.

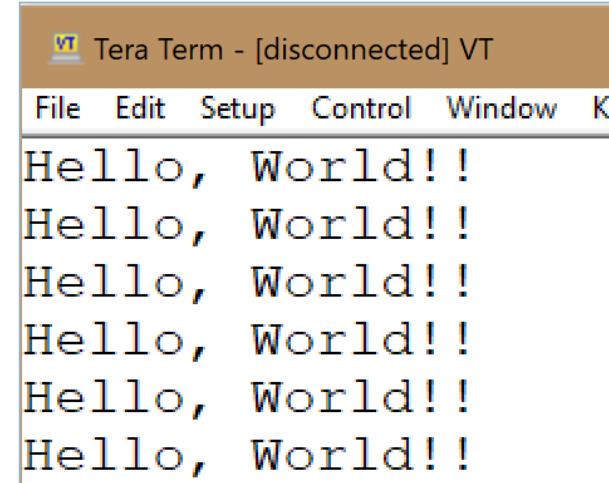
- Parameters
- **huart:** UART handle.
  - **pData:** pointer to data buffer.
  - **Size:** amount of data to be received.
  - **Timeout:** Timeout duration.
- Return values
- **HAL:** status

## stm32f7xx\_hal\_uart.h

```
/** @brief Check whether the specified UART flag is set or not.
 * @param __HANDLE__ specifies the UART Handle.
 * @param __FLAG__ specifies the flag to check.
 *   This parameter can be one of the following values:
 *     @arg @ref UART_FLAG_TEACK Transmit enable acknowledge flag
 *     @arg @ref UART_FLAG_RXWU Receiver wake up flag (if the UART in mute mode)
 *     @arg @ref UART_FLAG_SBKF Send Break flag
 *     @arg @ref UART_FLAG_CMF Character match flag
 *     @arg @ref UART_FLAG_BUSY Busy flag
 *     @arg @ref UART_FLAG_ABRF Auto Baud rate detection flag
 *     @arg @ref UART_FLAG_ABRE Auto Baud rate detection error flag
 *     @arg @ref UART_FLAG_CTS CTS Change flag
 *     @arg @ref UART_FLAG_LBDF LIN Break detection flag
 *     @arg @ref UART_FLAG_TXE Transmit data register empty flag
 *     @arg @ref UART_FLAG_TC Transmission Complete flag
 *     @arg @ref UART_FLAG_RXNE Receive data register not empty flag
 *     @arg @ref UART_FLAG_IDLE Idle Line detection flag
 *     @arg @ref UART_FLAG_ORE Overrun Error flag
 *     @arg @ref UART_FLAG_NE Noise Error flag
 *     @arg @ref UART_FLAG_FE Framing Error flag
 *     @arg @ref UART_FLAG_PE Parity Error flag
 * @retval The new state of __FLAG__ (TRUE or FALSE).
 */
#define __HAL_UART_GET_FLAG(__HANDLE__, __FLAG__) (((__HANDLE__)->Instance->ISR & (__FLAG__)) == (__FLAG__))
```

# Transmitting A String

```
char str[] = "Hello, World!!\n\r";  
  
while(__HAL_UART_GET_FLAG(&huart3, UART_FLAG_TC)==RESET) {}  
  
HAL_UART_Transmit(&huart3, (uint8_t*) str, strlen(str),1000);  
  
HAL_Delay(300);
```



# Receiving A Character

```
char ch1;

while(__HAL_UART_GET_FLAG(&huart3,UART_FLAG_RXNE)== RESET) {}

HAL_UART_Receive(&huart3, (uint8_t*) &ch1, 1, 1000);

if (ch1 == . . .)
```

# Summary

- UART is asynchronous transmission. (No Clock)
- Data frame is comprised of a start bit (logic 0), multiple data bits, maybe a parity bit and stop bit(s)
- Can config amount of data bits, parity bit and stop bit
- Main structure is shift register, buffer and baud rate control
- RS-232 uses +/- 15 volts
- 8 UARTs on STM32F767
- UART for Human Interface and Debugging

## Quiz

- 1) จงวาด timing diagram ของการส่งตัวอักษร ‘a’ แบบ UART และ RS-232
- 2) จงวาด timing diagram ของการส่งตัวอักษร ‘a’ และ ‘b’ แบบ UART
- 3) จงคำนวณหาระยะเวลาและ overhead (%) ในการส่งตัวอักษร 5 ตัว แบบ UART 2400/8/odd parity/1

# Reference

- [1] <https://www.slideshare.net/NaveenKumar11/uart-13407576>
- [2] <https://web.eecs.umich.edu/~prabal/teaching/eecs373/slides/serial.ppt>
- [3] <https://learn.sparkfun.com/tutorials/serial-communication/all>
- [4] <https://slideplayer.com/slide/4468742/>