Universal Asynchronous Receiver Transmitter (UART)

Embedded System 2561, KU CSC

Adapted by Sorayut Glomglome

Outline

1. Universal Asynchronous Receiver/Transmitter
2. Waveform
3. Transmitting & Receiving Mechanism
4. UART Block Diagram
5. RS-232
6. STM32F411 UART
7. Coding

Learning Outcomes

1. Understanding asynchronous 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.

Why 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.

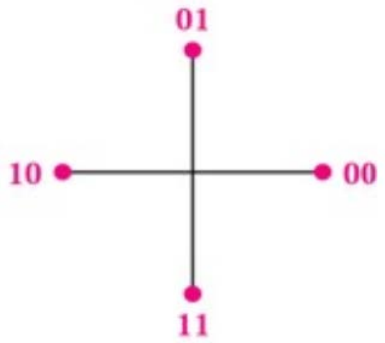
Overhead

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

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

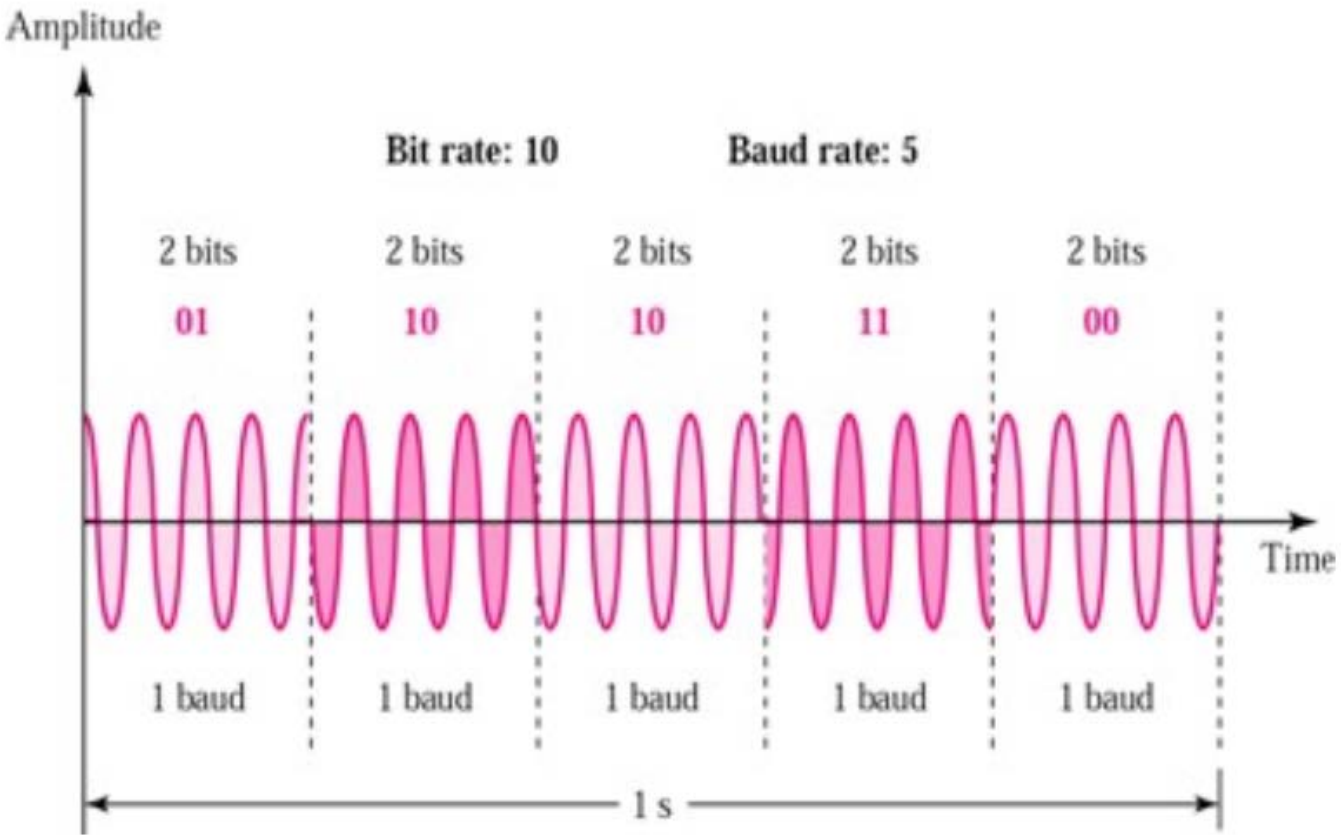
Dibit	Phase
00	0
01	90
10	180
11	270

Dibit
(2 bits)



Constellation diagram

1 symbol carries 2 bits

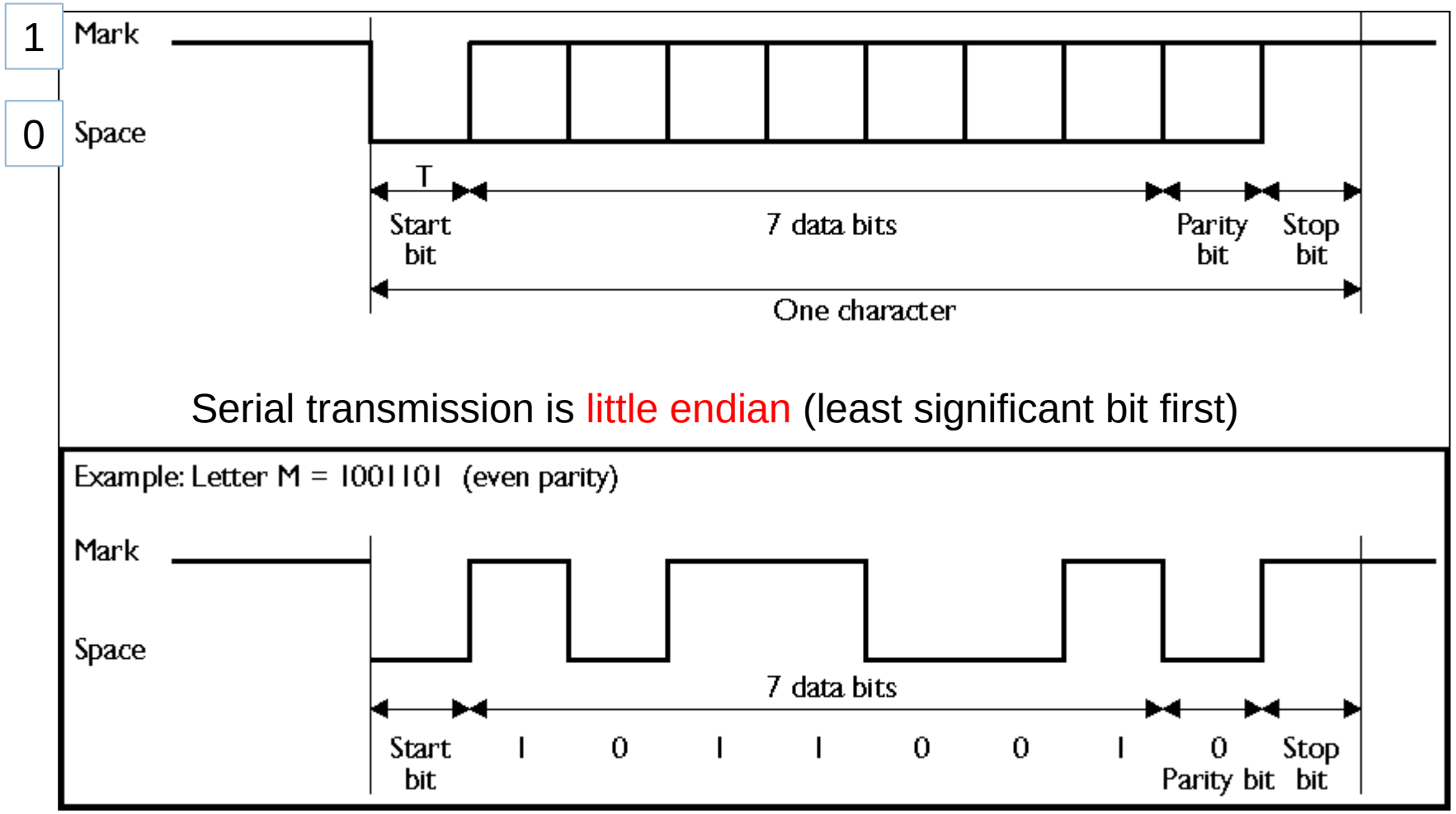


https://slideplayer.com/slide/4468742/

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



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	[ENG 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

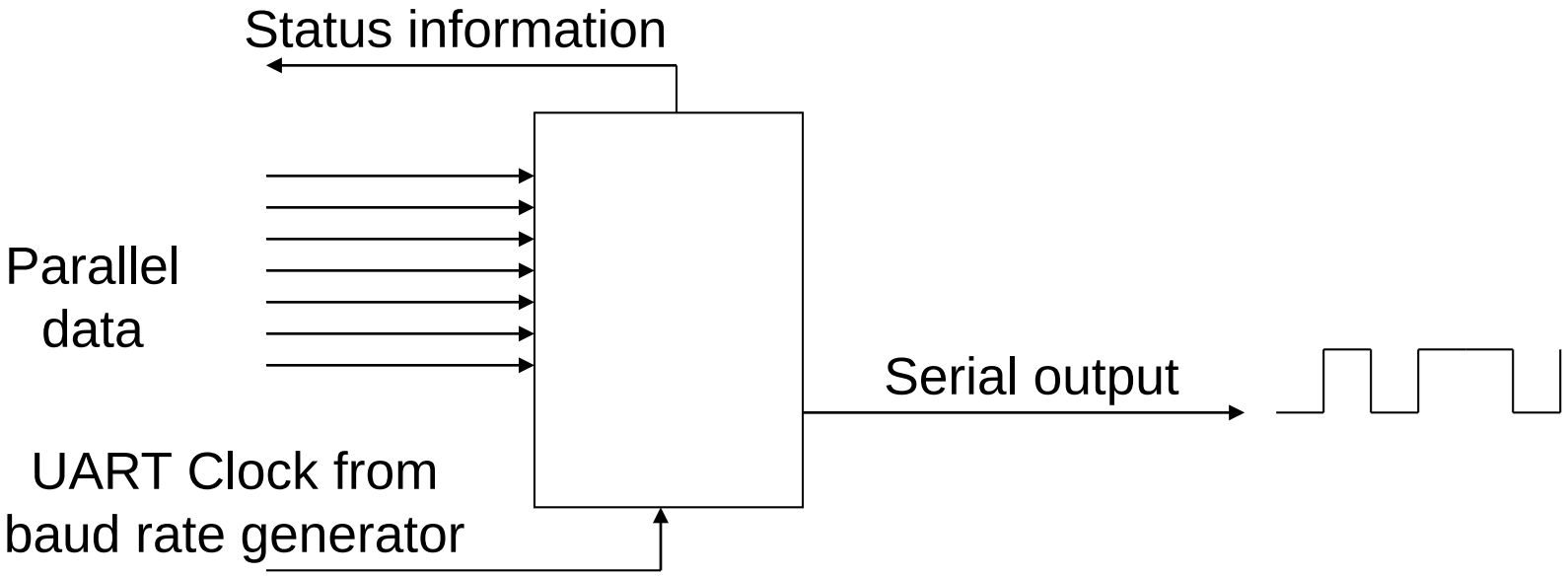
- Baud rate (bits per second)
  - 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 128000, 256000, . . .
- Data bits
  - 5, 6, 7 or 8 bits
- Stop Bits
  - 1, 1.5 or 2 stop bits
- Error Checking
  - Even, odd or no parity bit

9600 8N1 – Most Default Configuration

- 9600 baud, 8 data bits, no parity, and 1 stop bit
- Little endian
- What are the transmitted characters?

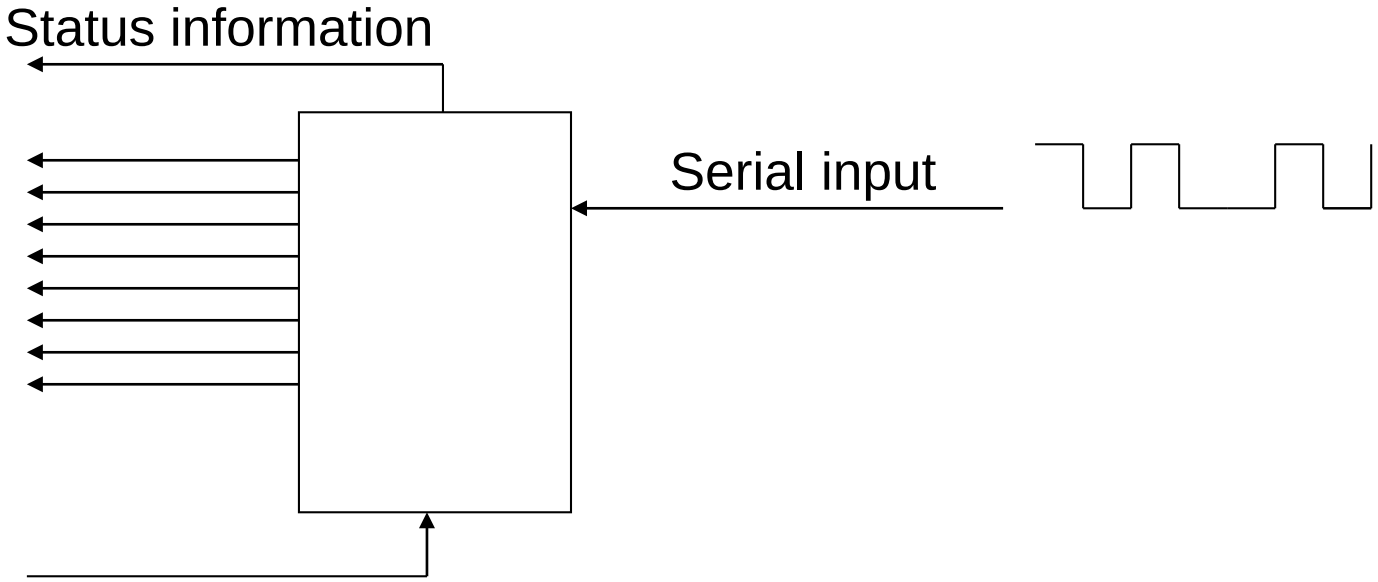
UART - Transmitter

- 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

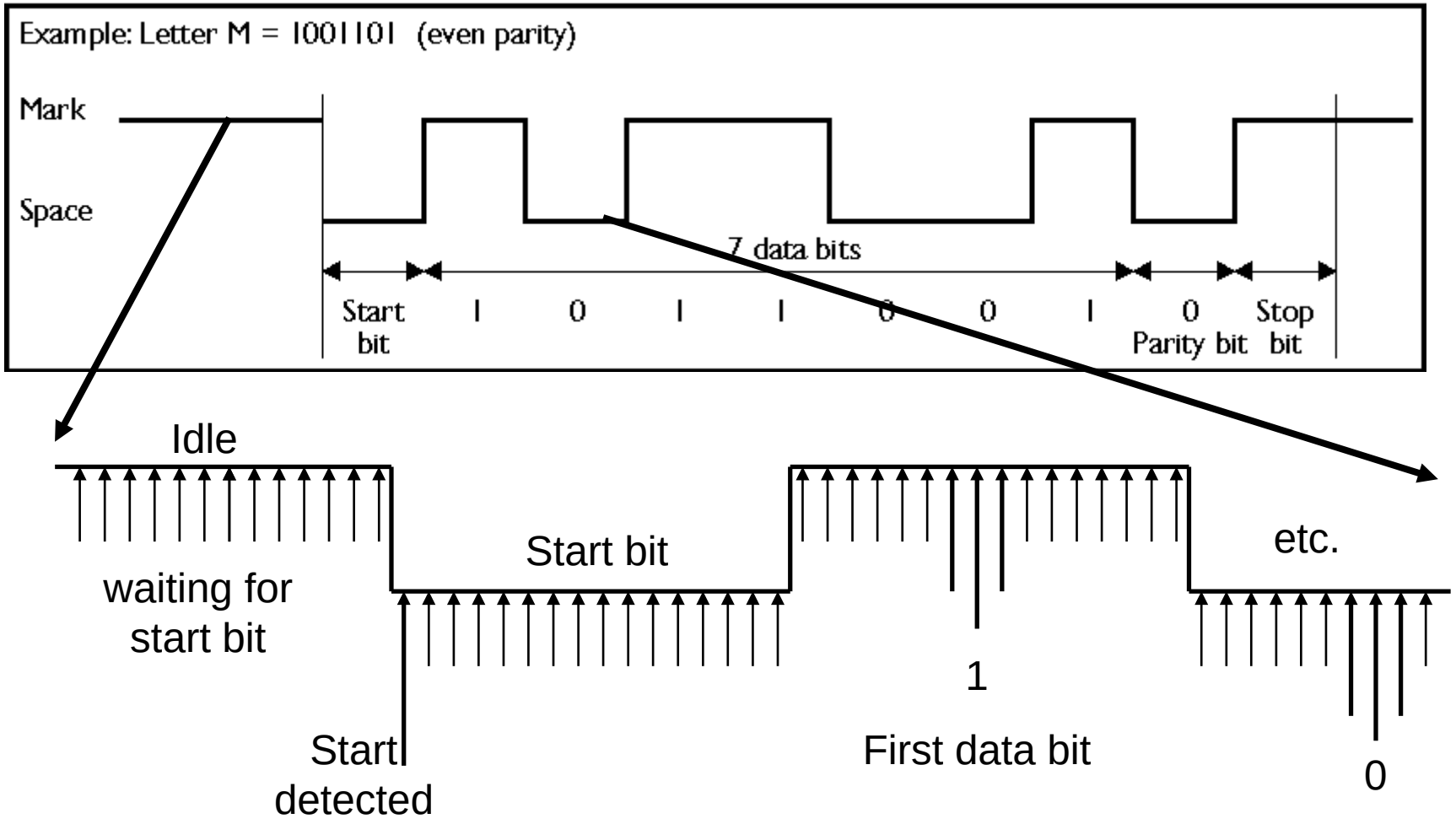


UART - The Receiver

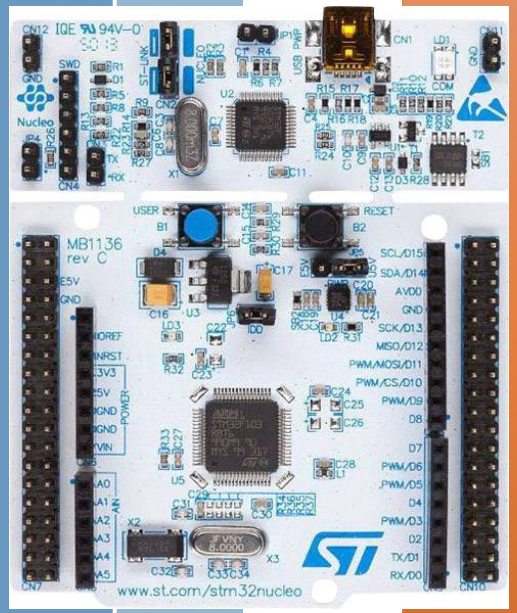
- Synchronises with transmitter using the falling edge of the start bit.
- Samples the input data line at a clock rate that is normally a multiple of baud rate, typically 16 times the baud rate.
- Removes the start and stop bits, optional calculates and checks the parity bit.
- Presents the received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



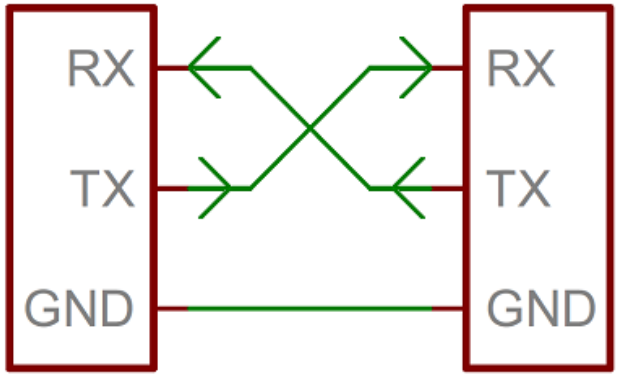
Asynchronous serial reception



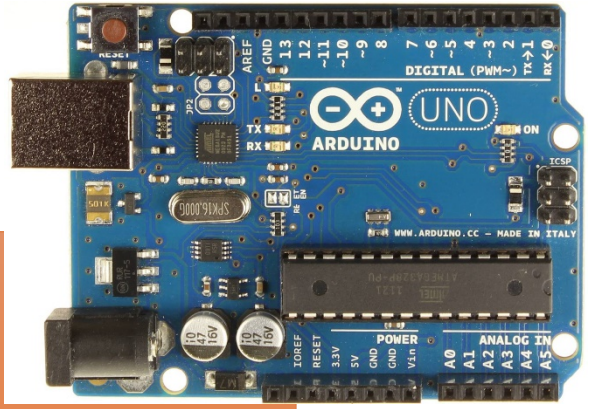
UART Application 1

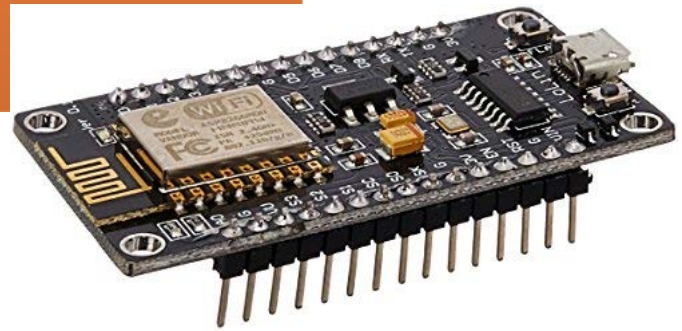


MCU 1



MCU 2

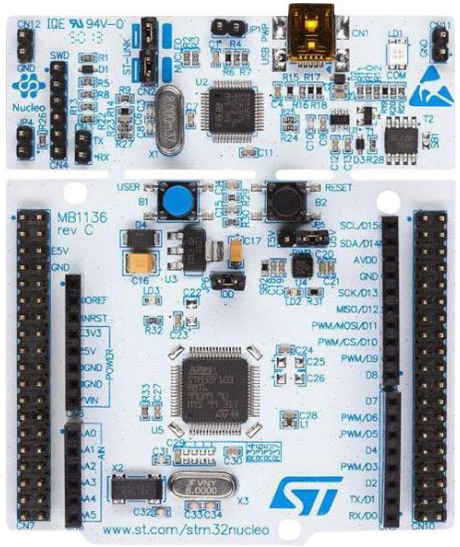


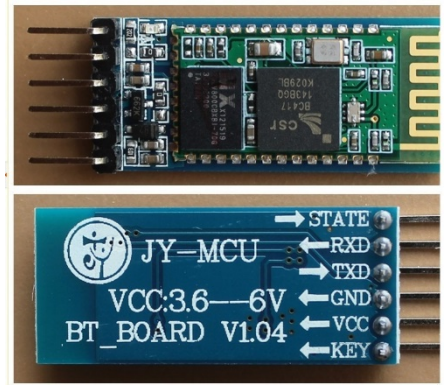


- Signal Level?
- Work between different MCU Models.

UART Application 2

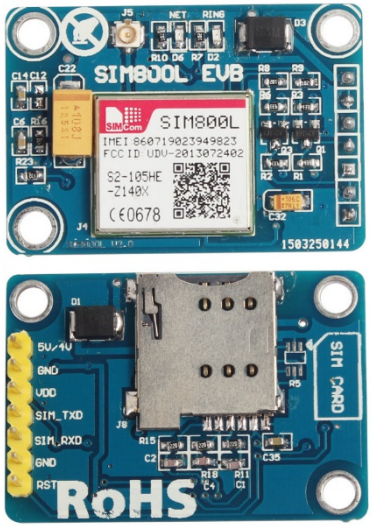
MCU





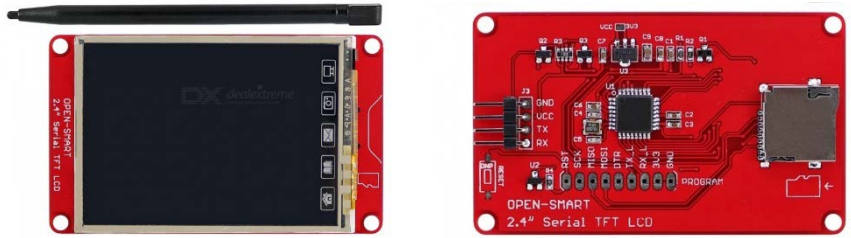
Bluetooth Module

HC-05



GPRS Module

SIM800L



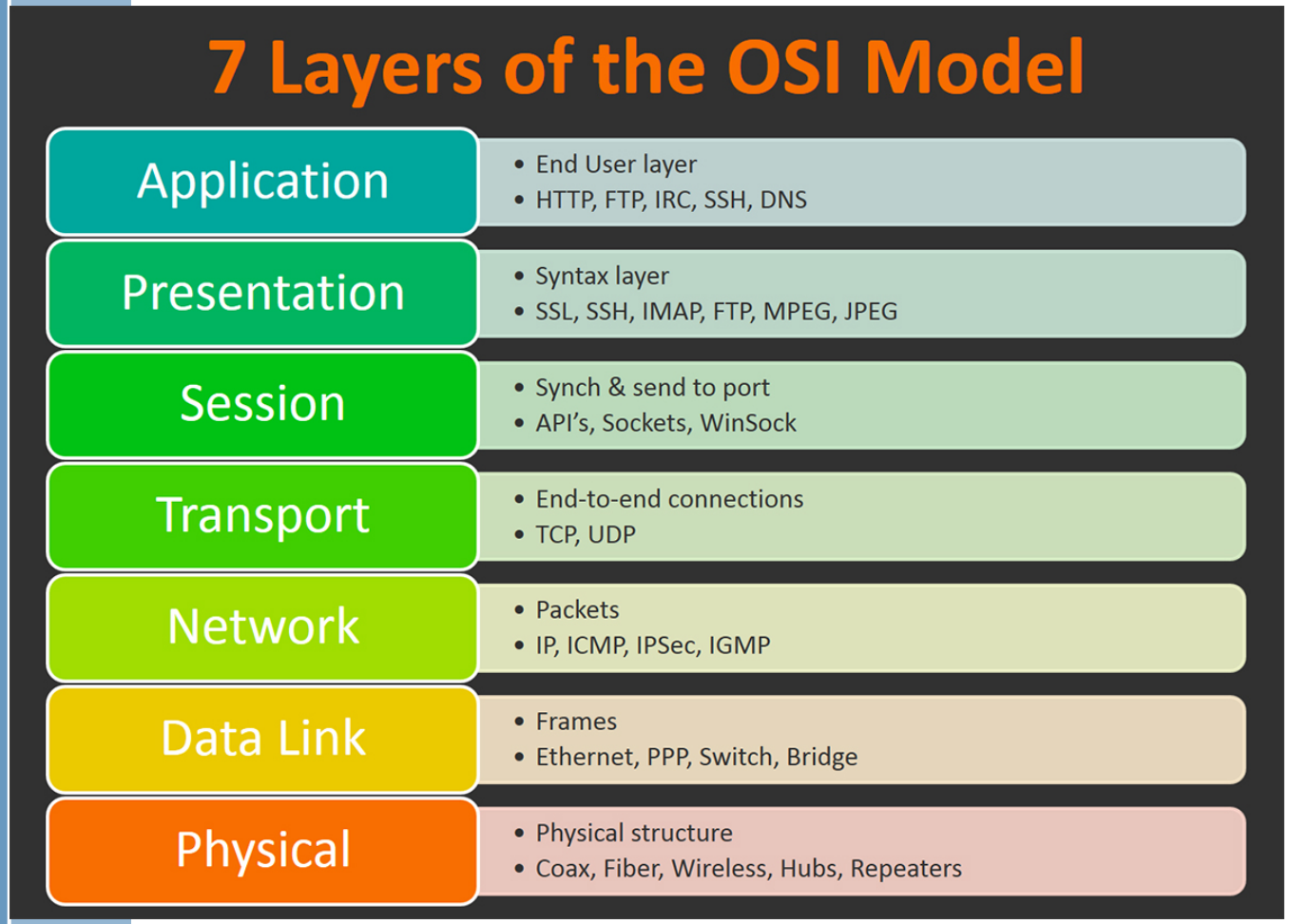
2.4” UART TFT LCD with Touch Sensor

UART Application 3

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

UART over RS232

The Open Systems Interconnection (OSI) Reference Model



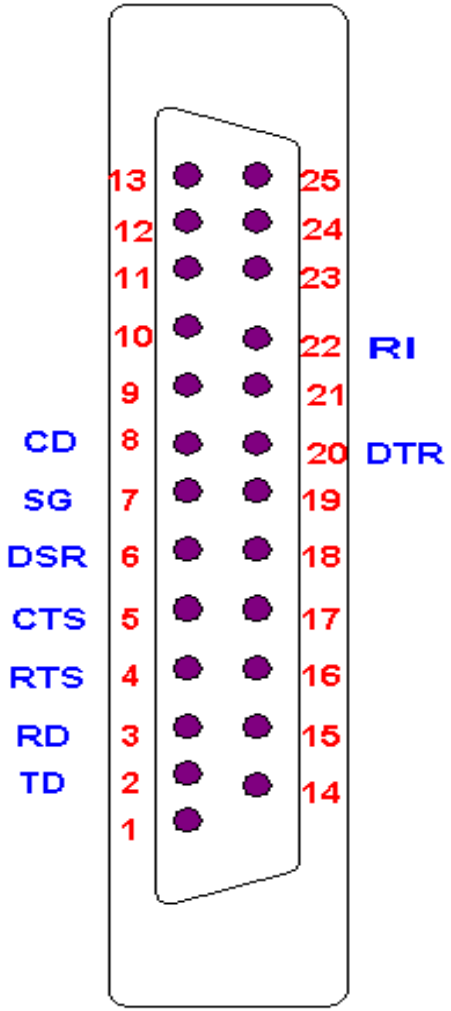
L2: UART

L1: TTL / RS232

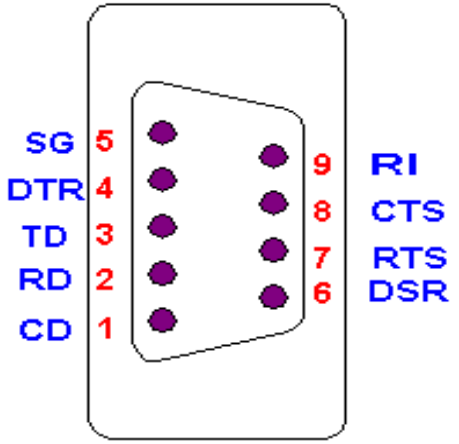
EIA RS232C Serial Interface Standard

- 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.

RS-232 Connectors

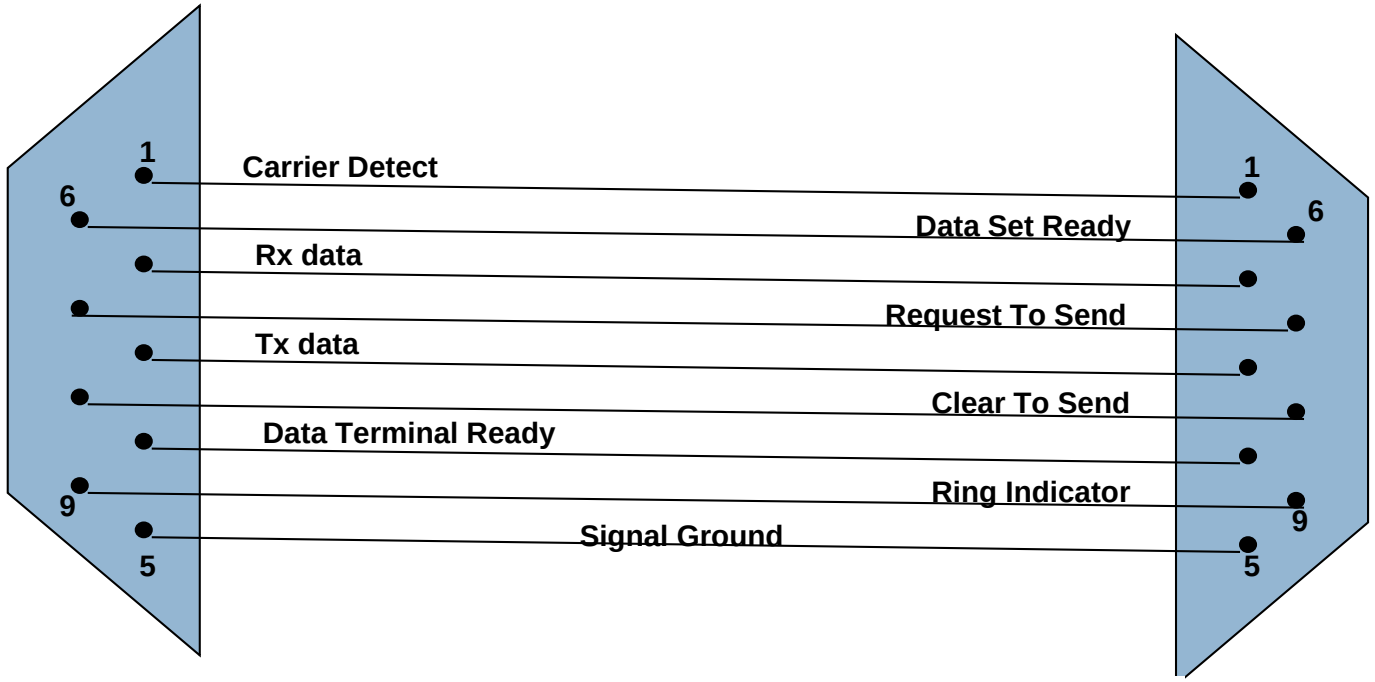


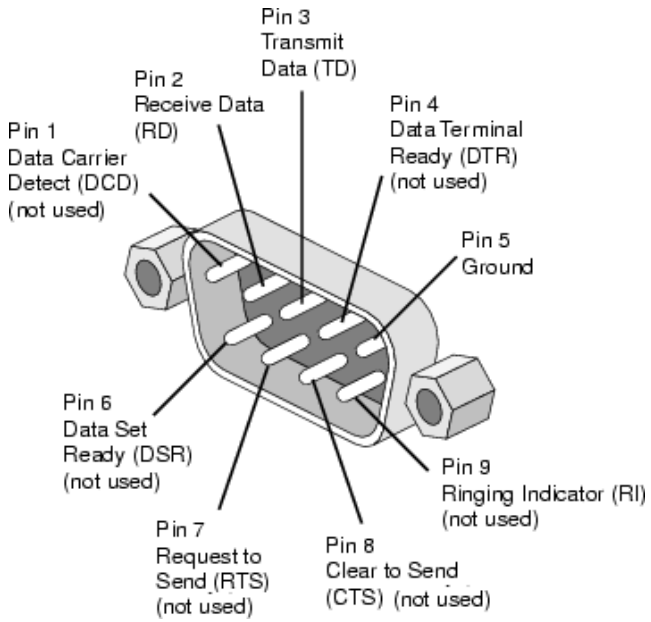
DB 25 Male Connecter

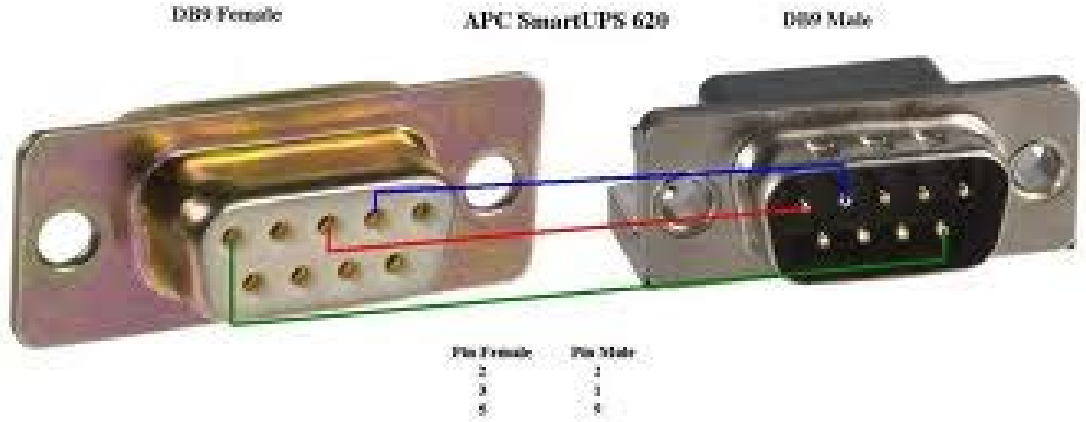


DB 9 Male Connecter

DB-9 Pin Assignment





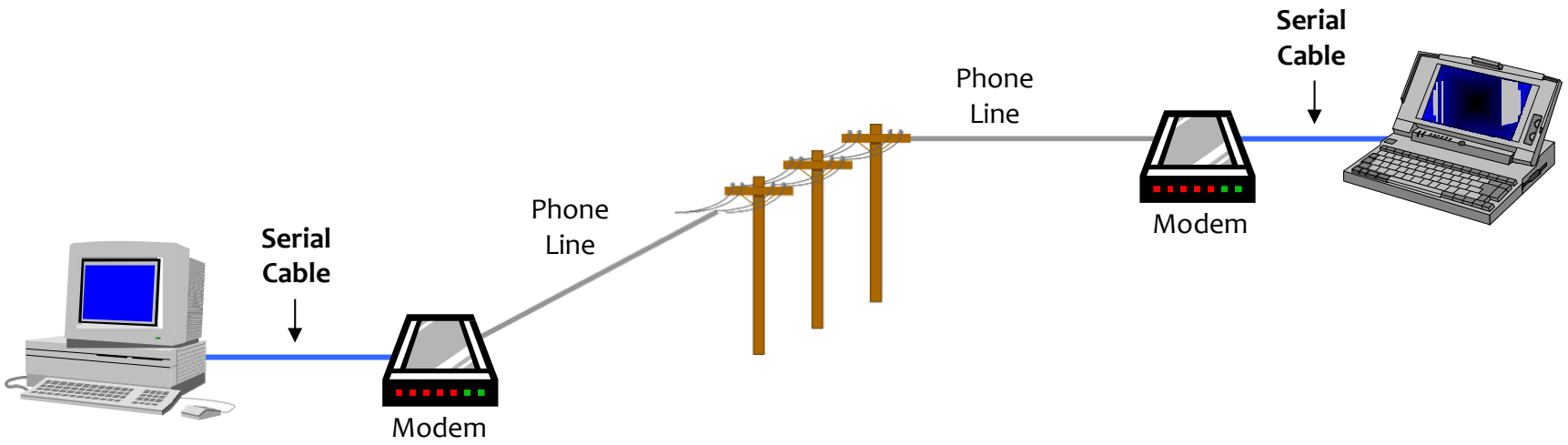


Pin Functions

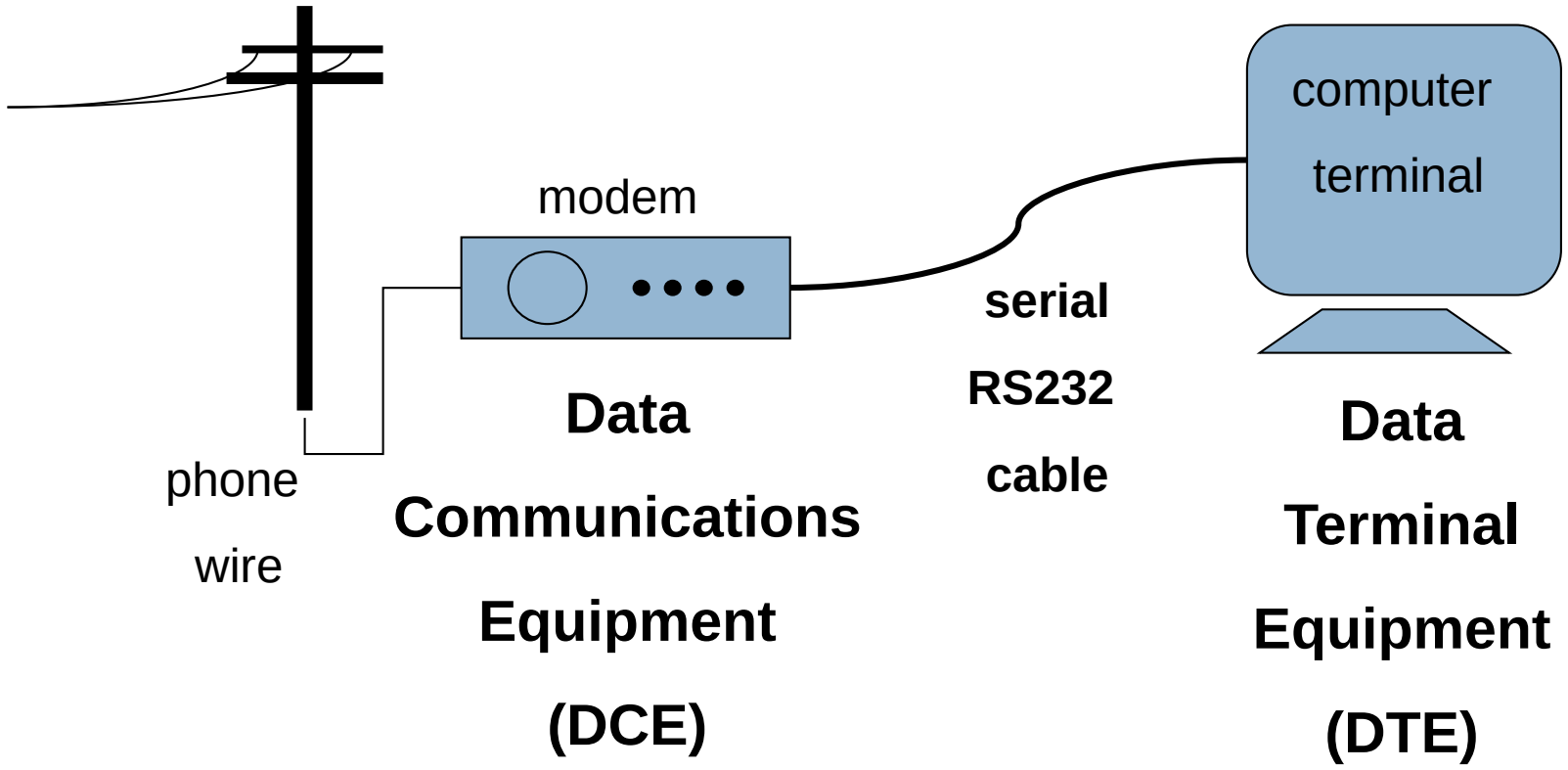
Pin No.	Signal	Name	Functions
1	CD	Carrier Detect	It is used by Modem to inform PC that it has detected Carrier on Phone Line.
2	RD	Received Data	Serial data is received on this line by PC.
3	TD	Transmit Data	Serial Data is transmitted on this pin by PC.
4	DTR	Data Terminal Ready	When terminal (computer) 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

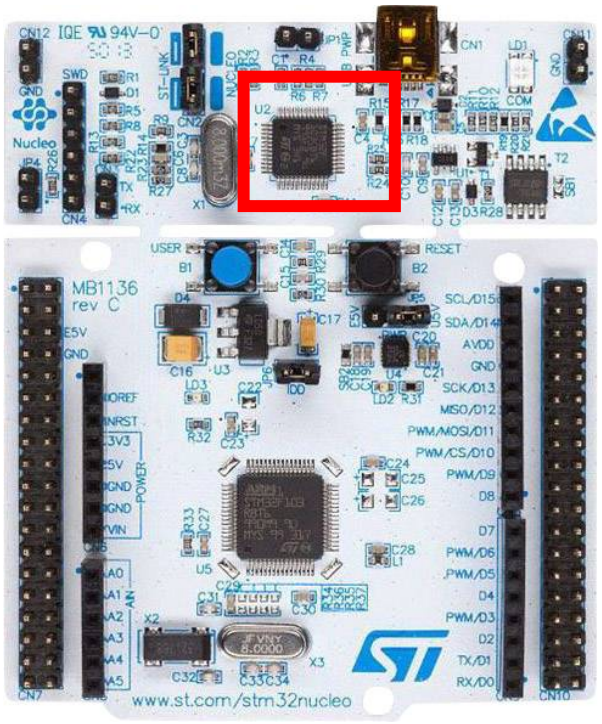
- A PC with a modem

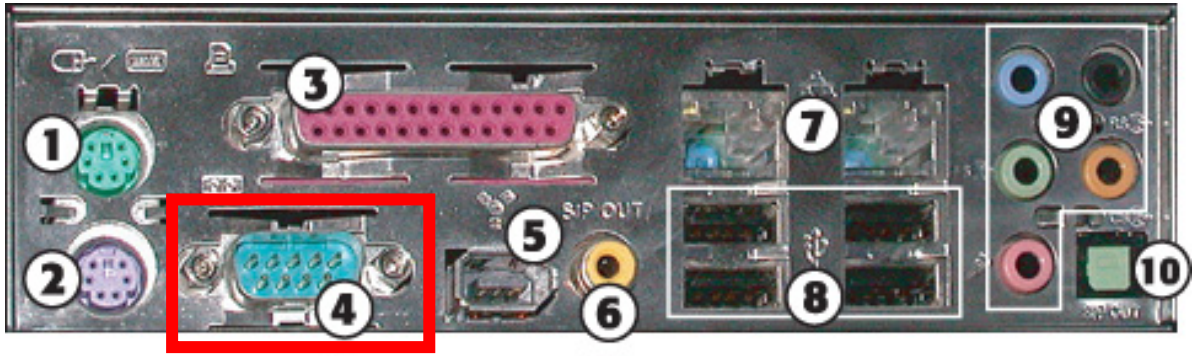


DCE/DTE


computer
terminal
modem
serial
RS232
cable
phone
wire
Data
Communications
Equipment
(DCE)
Data
Terminal
Equipment
(DTE)

COM Port Legacy





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

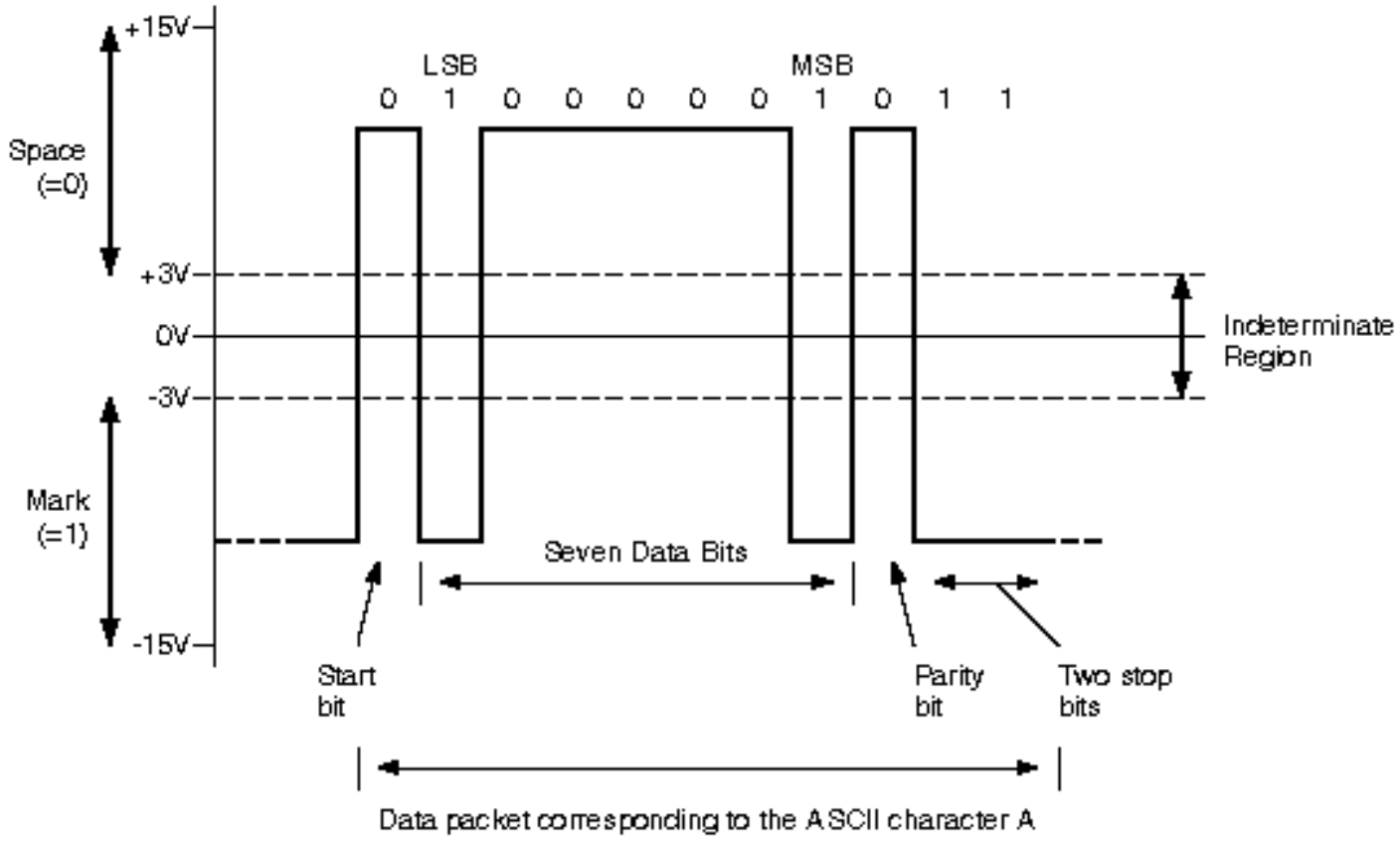
USB to Serial

ST-Link

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

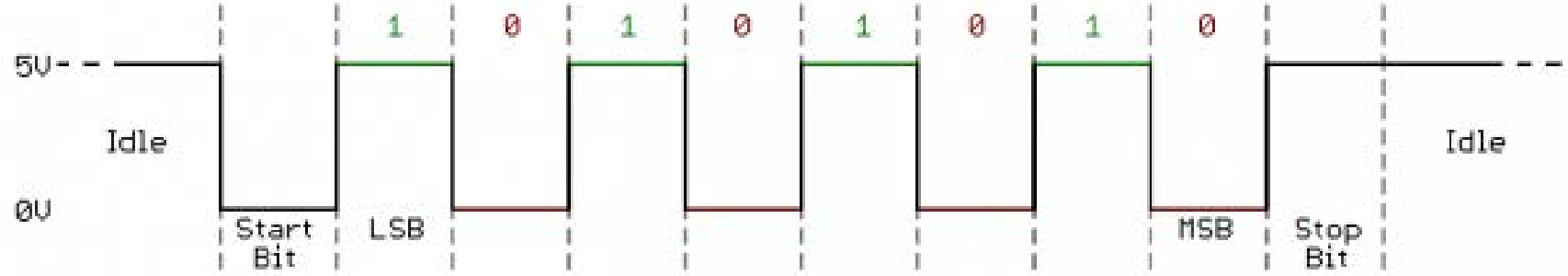
RS232 Signal Voltage Levels

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

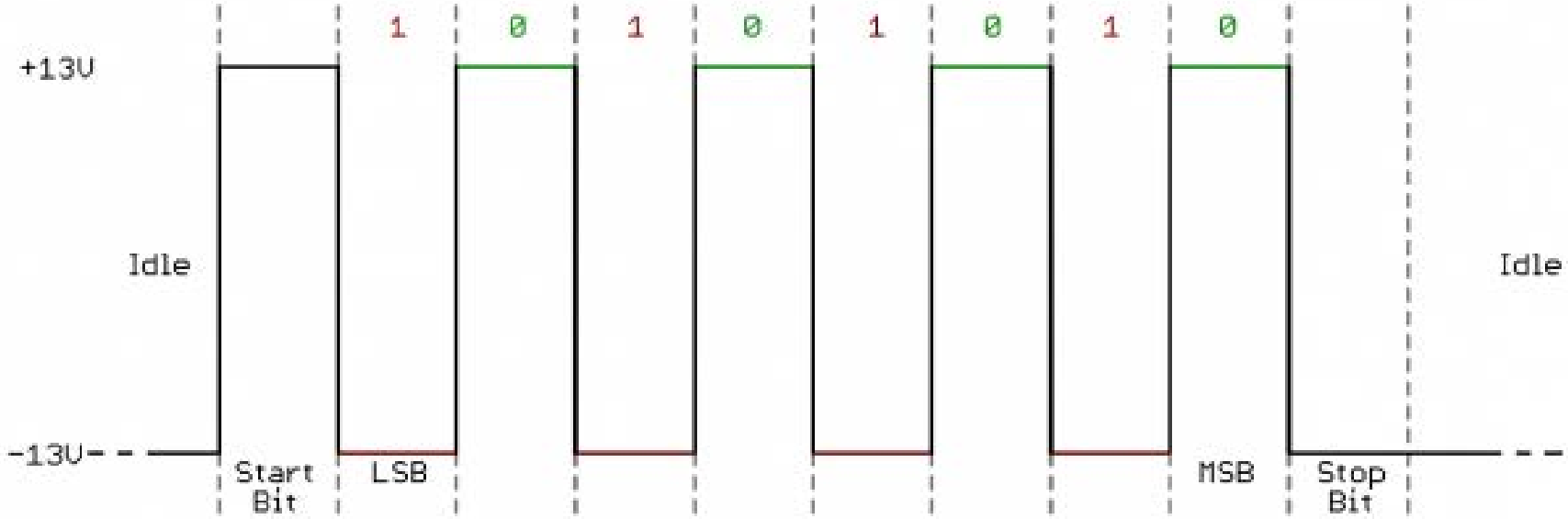


UART TTL&RS232 Signal Voltage Level

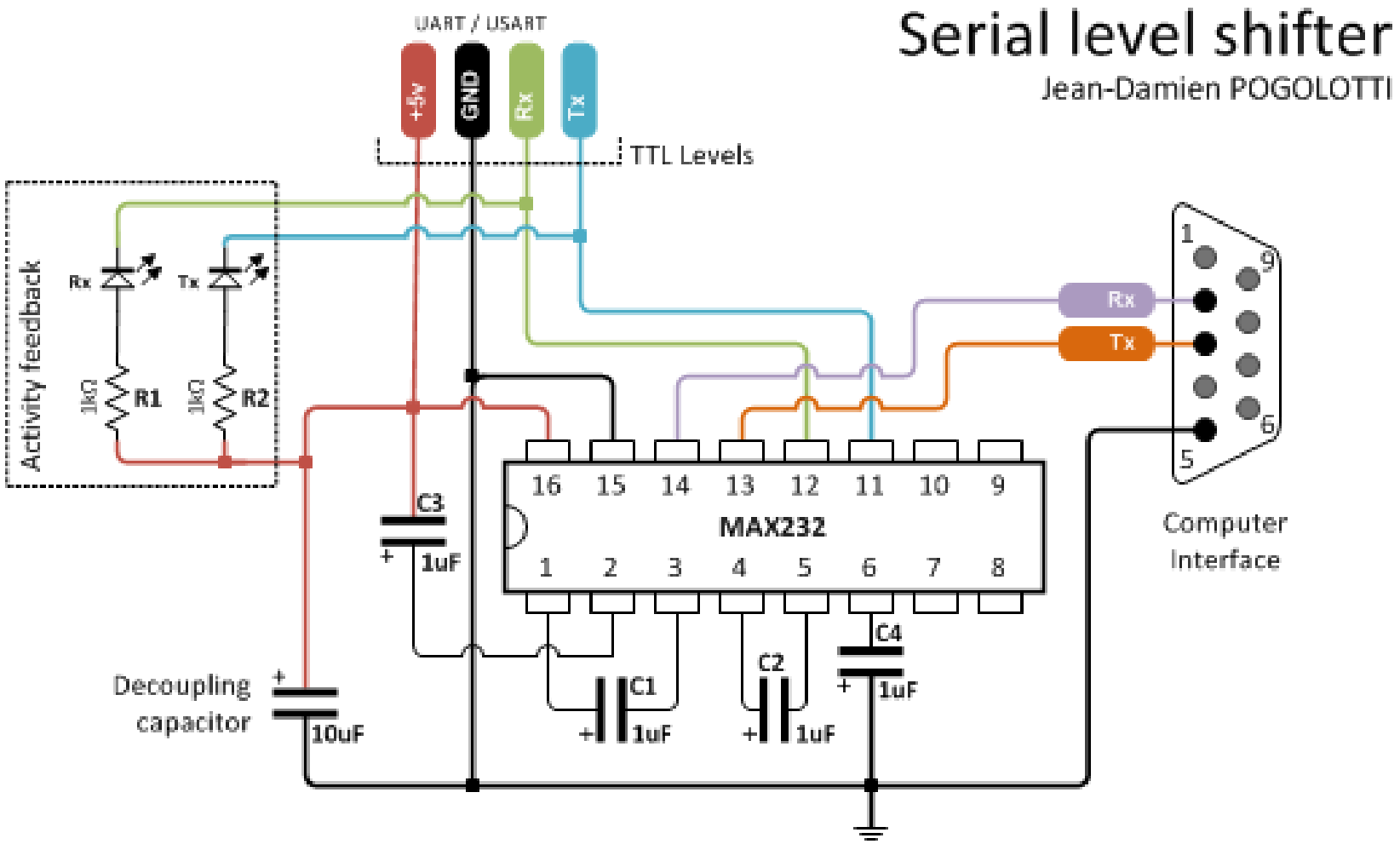
UART TTL



UART RS232



MAX232 IC : UART <-> RS-232 Conversion



UART Modules on STM32F411

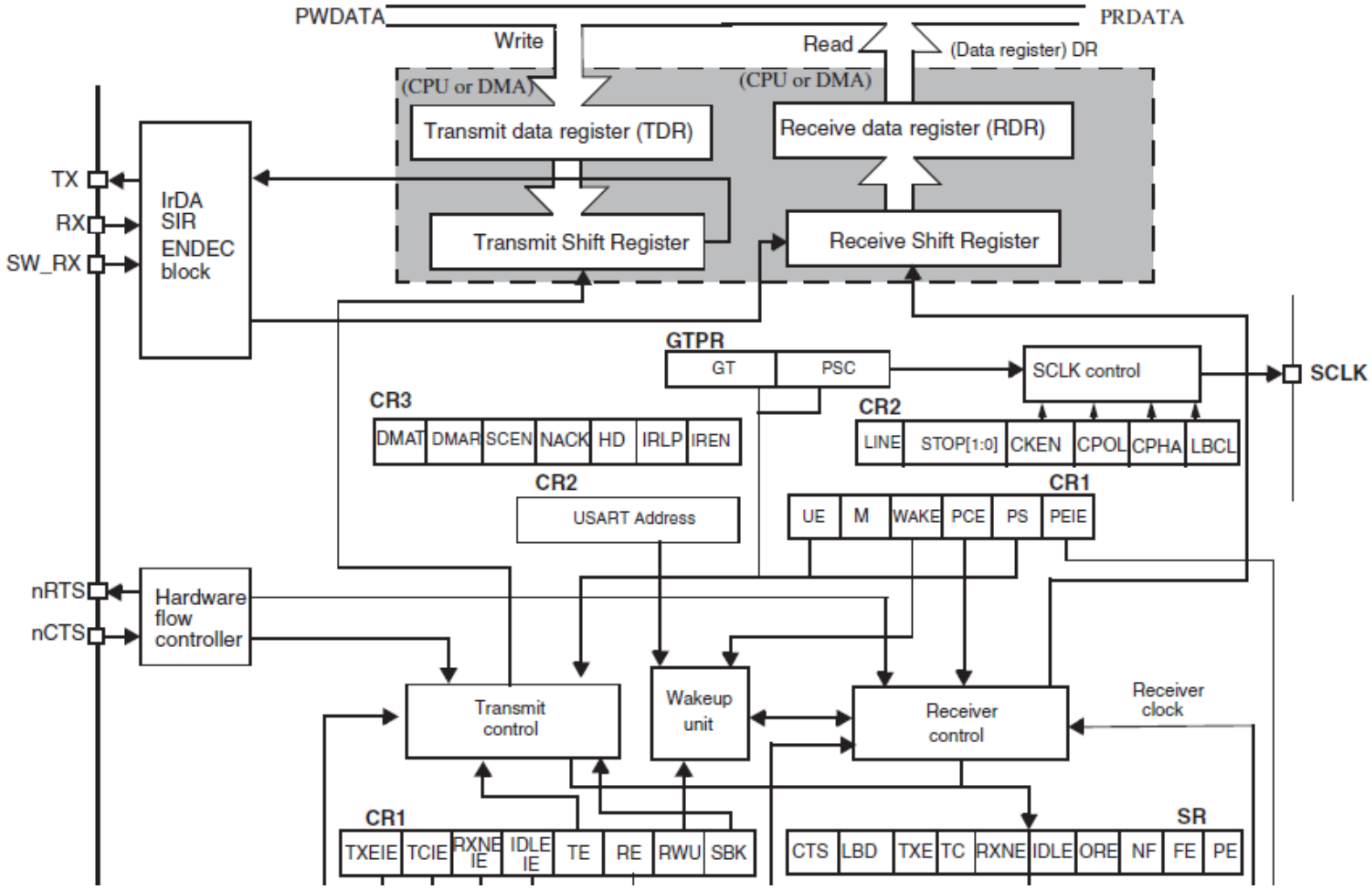
UARTs on STM32F411

- Full duplex, asynchronous communications
- Programmable data word length (8/9 bits)
- Configurable stop bits (1/2 stop bits)
- Single-wire half duplex communications
- Configurable multibuffer communication using DMA

Table 6. USART feature comparison

USART name	Standard features	Modem (RTS/CTS)	LIN	SPI master	irDA	Smartcard (ISO 7816)	Max. baud rate in Mbit/s (oversampling by 16)	Max. baud rate in Mbit/s (oversampling by 8)	APB mapping
USART1	X	X	X	X	X	X	6.25	12.5	APB2 (max. 100 MHz)
USART2	X	X	X	X	X	X	3.12	6.25	APB1 (max. 50 MHz)
USART6	X	N.A	X	X	X	X	6.25	12.5	APB2 (max. 100 MHz)

UART Block Diagram



UART Registers and Memory Map

No.	Register Name	Macro	Offset
1	**Status register**	USART_SR	0x00
2	**Data register**	USART_DR	0x04
3	Baud rate register	USART_BRR	0x08
4	**Control register 1**	USART_CR1	0x0C
5	Control register 2	USART_CR2	0x10
6	Control register 3	USART_CR3	0x14
7	Guard time and prescaler register	USART_GTPR	0x18

Bus	Starting Address	UART No.
APB2	0x4001 1400	USART6
	0x4001 1000	USART1
APB1	0x4000 4400	USART2

19.6.1 Status register (USART_SR)

Address offset: 0x00

Reset value: 0x00C0 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

Bit	Flag	Set by	Reset by	Logic 0	Logic 1
7	**TXE**: Transmit data register empty	HW	SW	data is not transferred to the shift register	data is transferred to the shift register)
6	**TC**: Transmission complete	HW	write to USART_TDR	Transmission is not complete	Transmission is complete
5	**RXNE**: Read data register not empty	HW	Read USART_DR	data is not received	Received data is ready to be read.
4	**IDLE**: Idle line detected	HW	SW	No Idle line is detected	Idle line is detected
2	**NF**: START bit Noise detection flag	HW	SW	No noise is detected	Noise is detected
0	**PE:** Parity error	HW	SW	No parity error	Parity error

19.6.2 Data register (USART_DR)

Address offset: 0x04

Reset value: 0xXXXX XXXX

Bits 31:9 Reserved, must be kept at reset value

Bits 8:0 **DR[8:0]**: Data value

Contains the Received or Transmitted data character, depending on whether it is read from or written to.

The Data register performs a double function (read and write) since it is composed of two registers, one for transmission (TDR) and one for reception (RDR)

The TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 1).

The RDR register provides the parallel interface between the input shift register and the internal bus.

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.

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

19.6.4 Control register 1 (USART_CR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
rw	Res.	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 12 **M**: Word length

This bit determines the word length. It is set or cleared by software.

0: 1 Start bit, 8 Data bits, n Stop bit

1: 1 Start bit, 9 Data bits, n Stop bit

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

19.6.5 Control register 2 (USART_CR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

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

Bits 31:15 Reserved, must be kept at reset value

Bits 13:12 **STOP**: STOP bits

These bits are used for programming the stop bits.

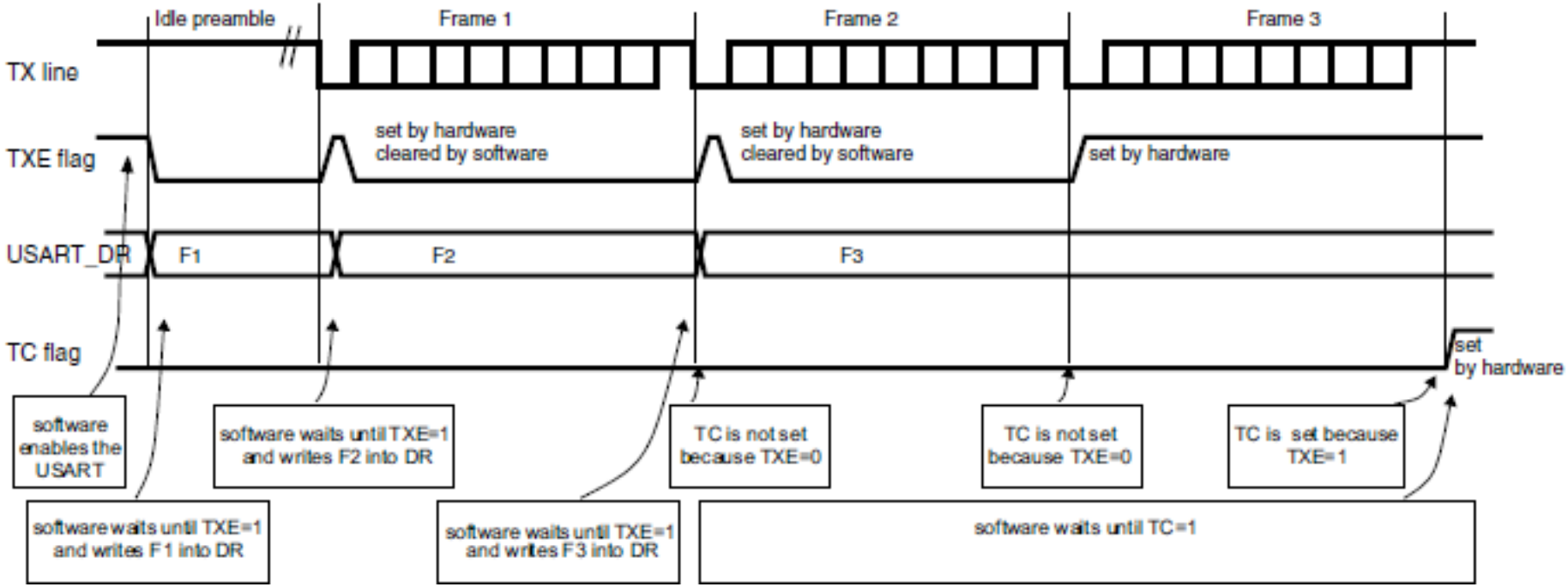
00: 1 Stop bit

01: 0.5 Stop bit

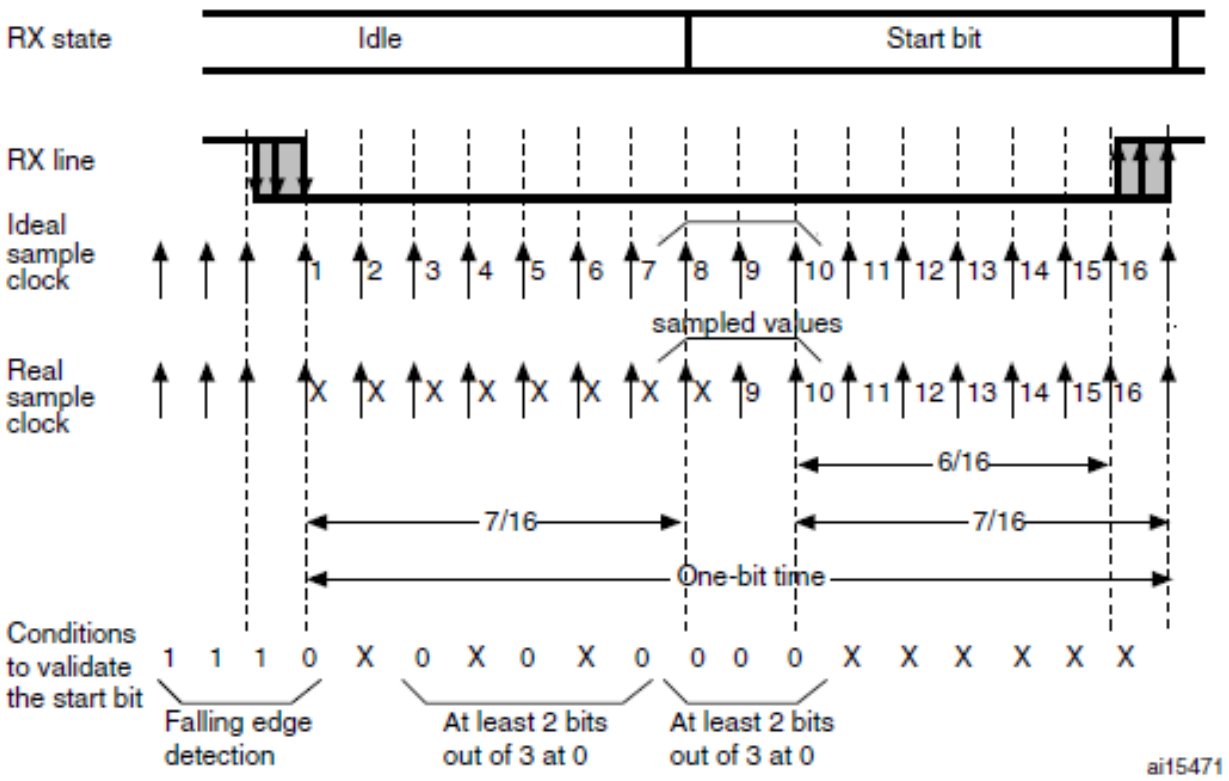
10: 2 Stop bits

Note: 11: 1.5 Stop bit

Transmitter



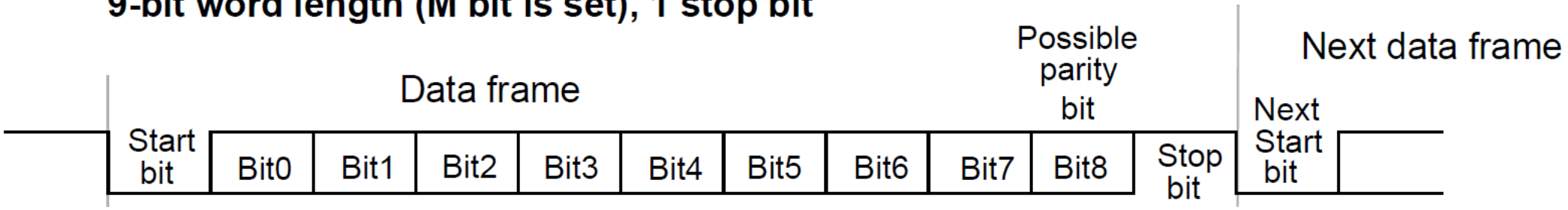
Receiver



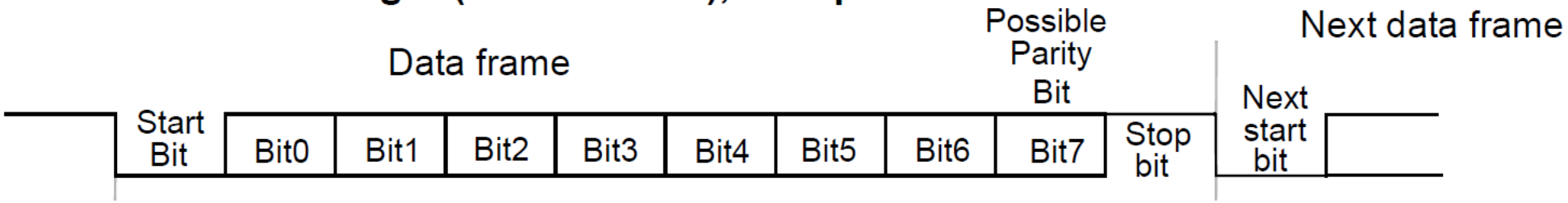
Sampled value	NE status	Received bit value	Data validity
000	0	0	Valid
001	1	0	Not Valid
010	1	0	Not Valid
011	1	1	Not Valid
100	1	0	Not Valid
101	1	1	Not Valid
110	1	1	Not Valid
111	0	1	Valid

Word Length Programming

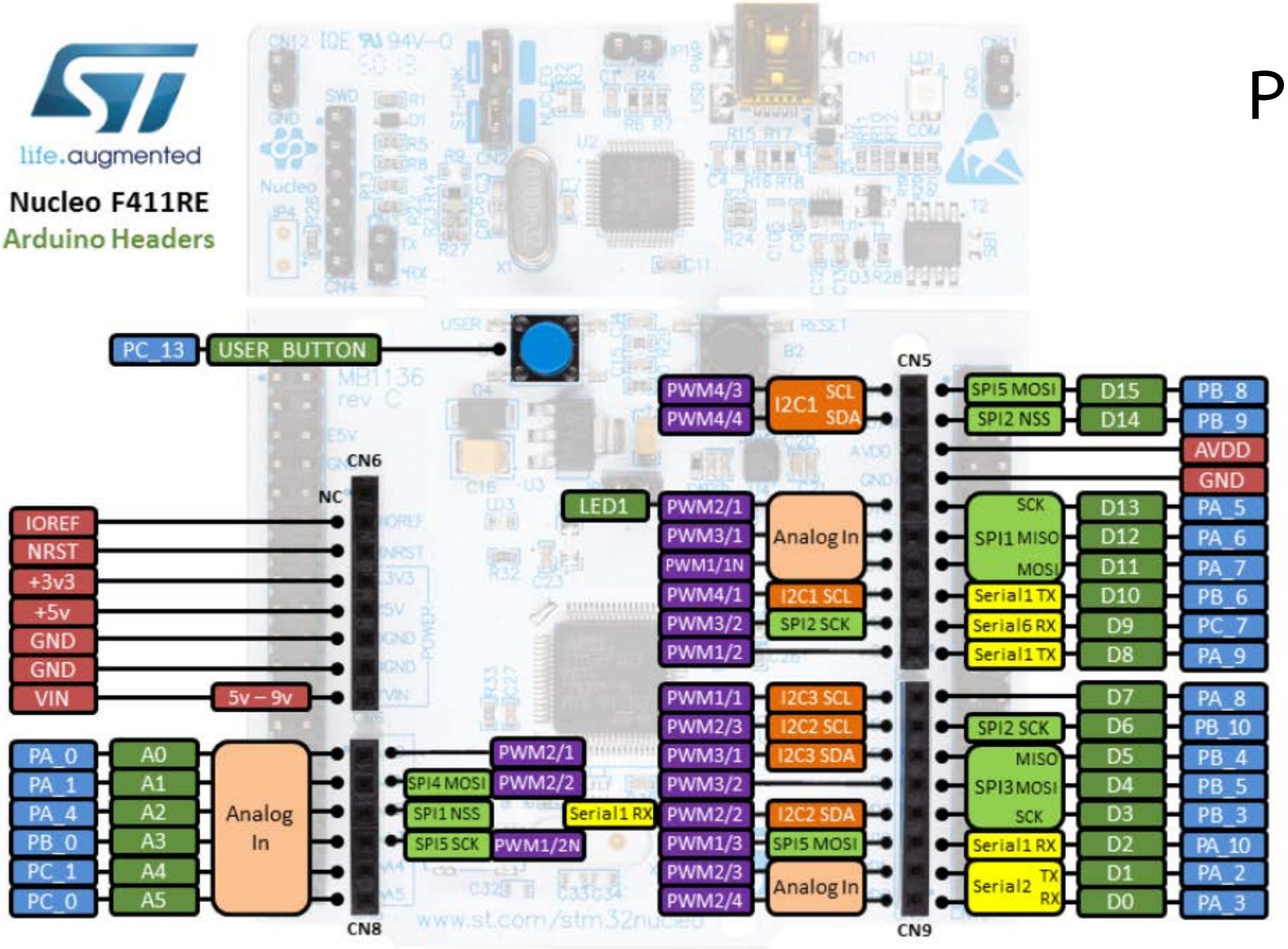
9-bit word length (M bit is set), 1 stop bit



8-bit word length (M bit is reset), 1 stop bit



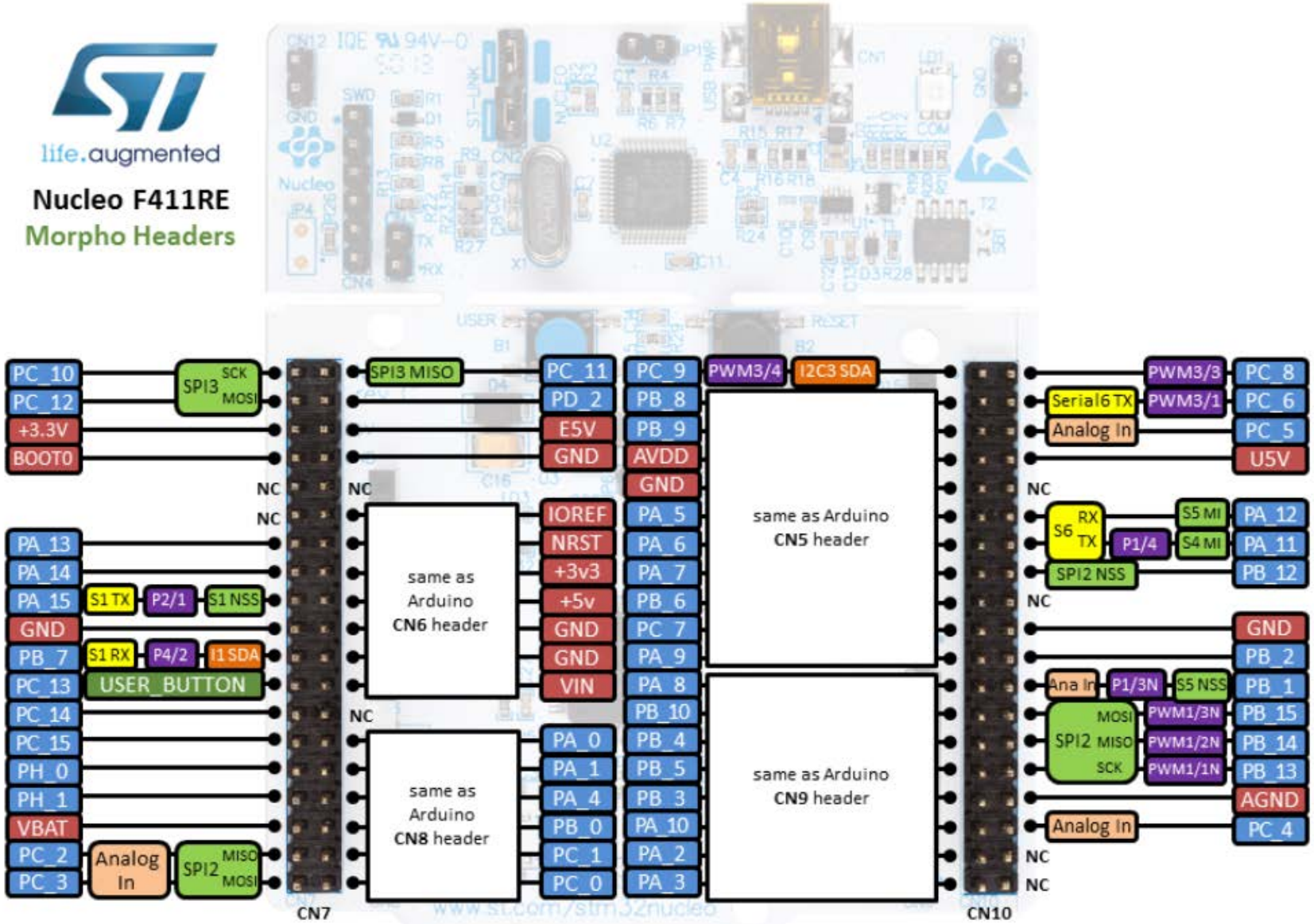
Pinout


life.augmented
Nucleo F411RE
Arduino Headers
PC_13
USER_BUTTON
CN5
PWM4/3
PWM4/4
I2C1 SCL SDA
SPI5 MOSI
SPI2 NSS
D15
D14
PB_8
PB_9
AVDD
GND
CN6
NC
IOREF
NRST
+3v3
+5v
GND
GND
VIN
5v – 9v
LED1
PWM2/1
PWM3/1
PWM1/1N
PWM4/1
PWM3/2
PWM1/2
Analog In
I2C1 SCL
SPI2 SCK
SCK
SPI1 MISO
MOSI
Serial1 TX
Serial6 RX
Serial1 TX
D13
D12
D11
D10
D9
D8
PA_5
PA_6
PA_7
PB_6
PC_7
PA_9
PA_0
PA_1
PA_4
PB_0
PC_1
PC_0
A0
A1
A2
A3
A4
A5
Analog In
PWM2/1
SPI4 MOSI
PWM2/2
SPI1 NSS
Serial1 RX
SPI5 SCK
PWM1/2N
CN8
PWM1/1
PWM2/3
PWM3/1
PWM3/2
PWM2/2
PWM1/3
PWM2/3
PWM2/4
I2C3 SCL
I2C2 SCL
I2C3 SDA
I2C2 SDA
SPI5 MOSI
Analog In
SPI2 SCK
MISO
SPI3 MOSI
SCK
Serial1 RX
Serial2 TX RX
D7
D6
D5
D4
D3
D2
D1
D0
PA_8
PB_10
PB_4
PB_5
PB_3
PA_10
PA_2
PA_3
CN9
www.st.com/stm32nucleo

Morpho headers

These headers give access to all STM32 pins.

https://os.mbed.com/platforms/ST-Nucleo-F411RE/



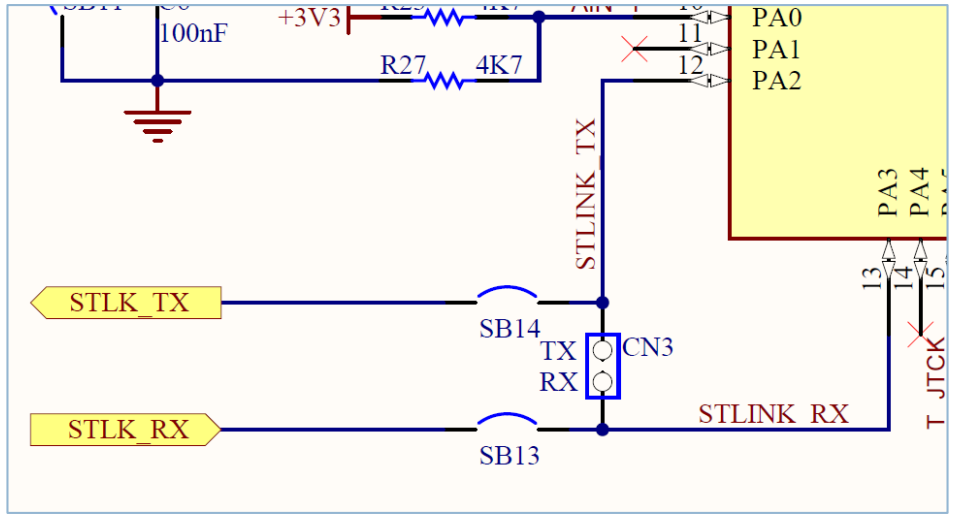
PIN Configuration

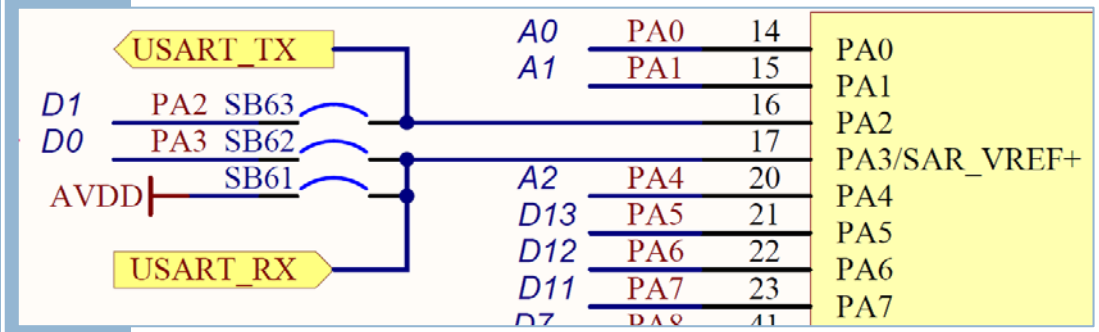
Pin NO.	Main Function (After Reset)	Alternate Functions	Additional Function
16	PA2	TIM2_CH3, TIM5_CH3, TIM9_CH1, I2S2_CKIN, USART2_TX, EVENTOUT	ADC1_2
17	PA3	TIM2_CH4, TIM5_CH4, TIM9_CH2, I2S2_MCK, USART2_RX, EVENTOUT	ADC1_3
37	PC6	TIM3_CH1, I2S2_MCK, USART6_TX, SDIO_D6, EVENTOUT	-
38	PC7	TIM3_CH2, SPI2_SCK/I2S2_CK, I2S3_MCK, USART6_RX, SDIO_D7, EVENTOUT	-
42	PA9	TIM1_CH2, I2C3_SMBA, USART1_TX, USB_FS_VBUS, SDIO_D2, EVENTOUT	OTG_FS_VBUS
43	PA10	TIM1_CH3, SPI5_MOSI/I2S5_SD, USART1_RX, USB_FS_ID, EVENTOUT	-

Virtual Communication Port

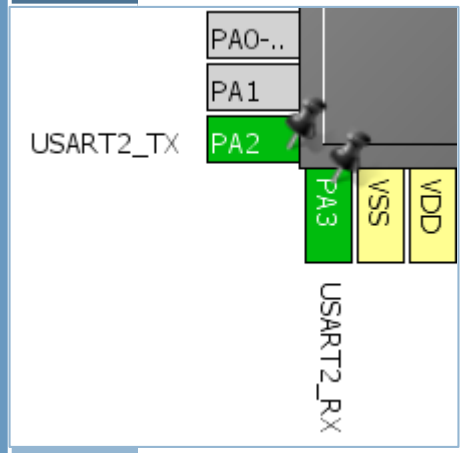
- Default UART2
  - PA2 – USART2_TX
  - PA3 – USART2_RX
- SB13 and SB14 ON, SB62 and SB63 OFF

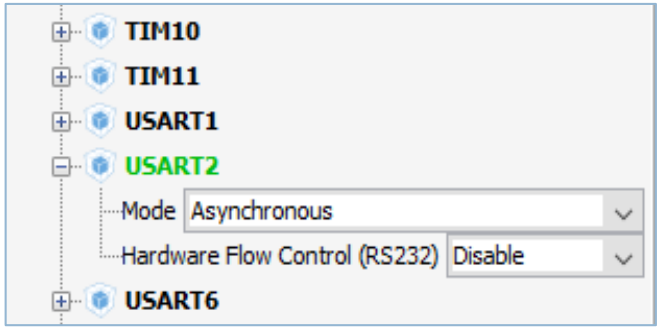
SB13 and SB14 ON, SB62 and SB63 OFF

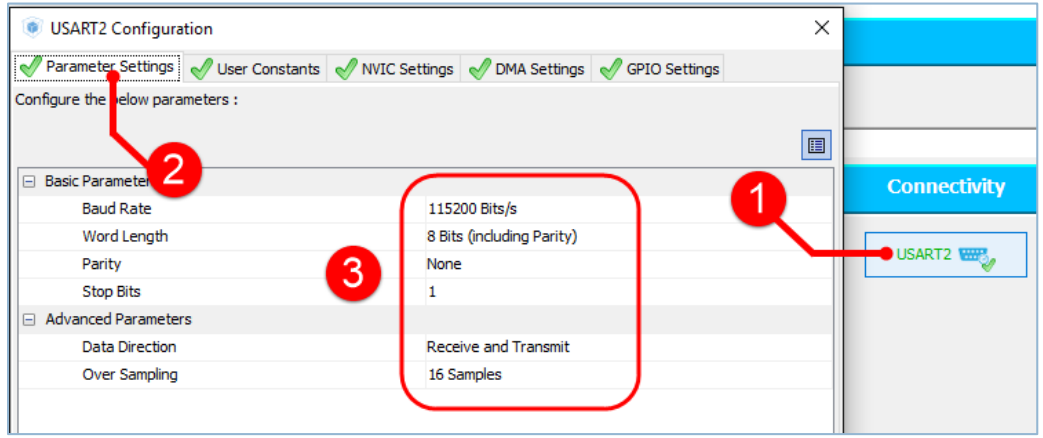


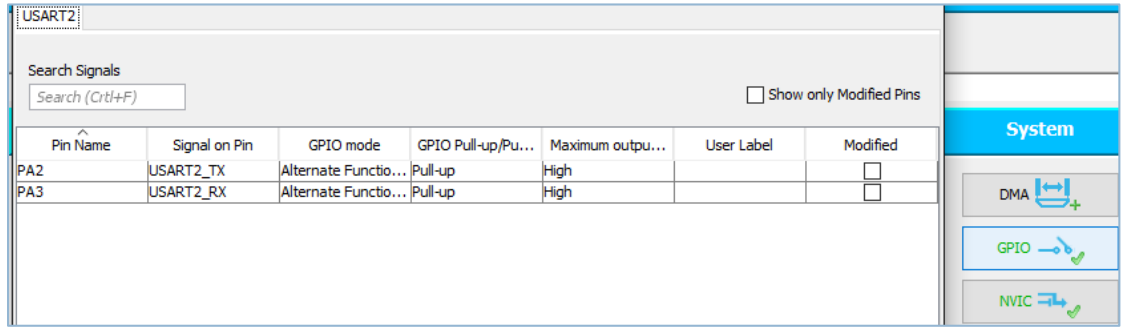


UART2 Configuration










```c
void HAL_UART_MspInit(UART_HandleTypeDef* huart)
{

  GPIO_InitTypeDef GPIO_InitStruct;
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspInit 0 */

  /* USER CODE END USART2_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_USART2_CLK_ENABLE();

    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    GPIO_InitStruct.Pin = GPIO_PIN_2|GPIO_PIN_3;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* USER CODE BEGIN USART2_MspInit 1 */

  /* USER CODE END USART2_MspInit 1 */
  }

}
```

```c
/* USART2 init function */
static void MX_USART2_UART_Init(void)
{

  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}
```

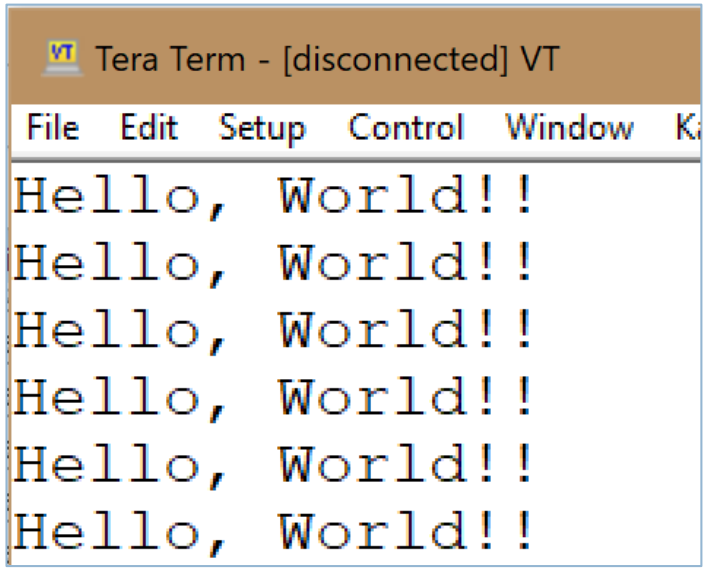
HAL_UART_Transmit

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

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

Transmitting A String




```
char str[]="Hello, World!!\n\r";
```

```
while(__HAL_UART_GET_FLAG(&huart2,UART_FLAG_TC)==RESET){}

HAL_UART_Transmit(&huart2, (uint8_t*) str, strlen(str),1000);

HAL_Delay(300);
```

Receiving A Character

```
char ch1;

while(__HAL_UART_GET_FLAG(&huart2,UART_FLAG_RXNE)== RESET){}
HAL_UART_Receive(&huart2, (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
- 3 UARTs on STM32F411
- 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/