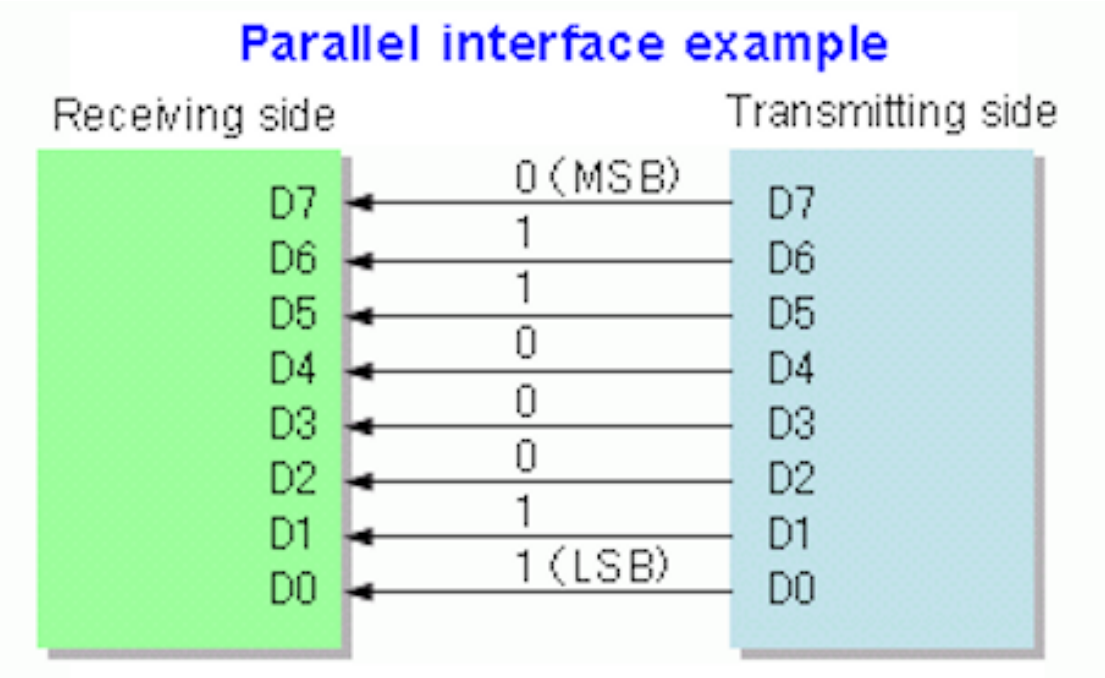
Outline

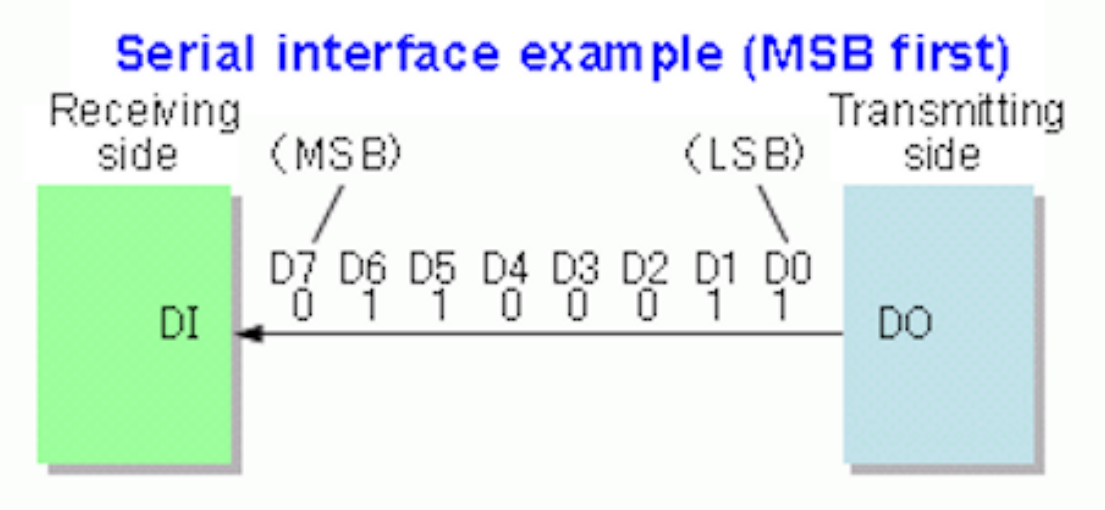
Introduction to USART

Universal (serial)
Synchronous
Asynchronous
Receiver
Transmitter

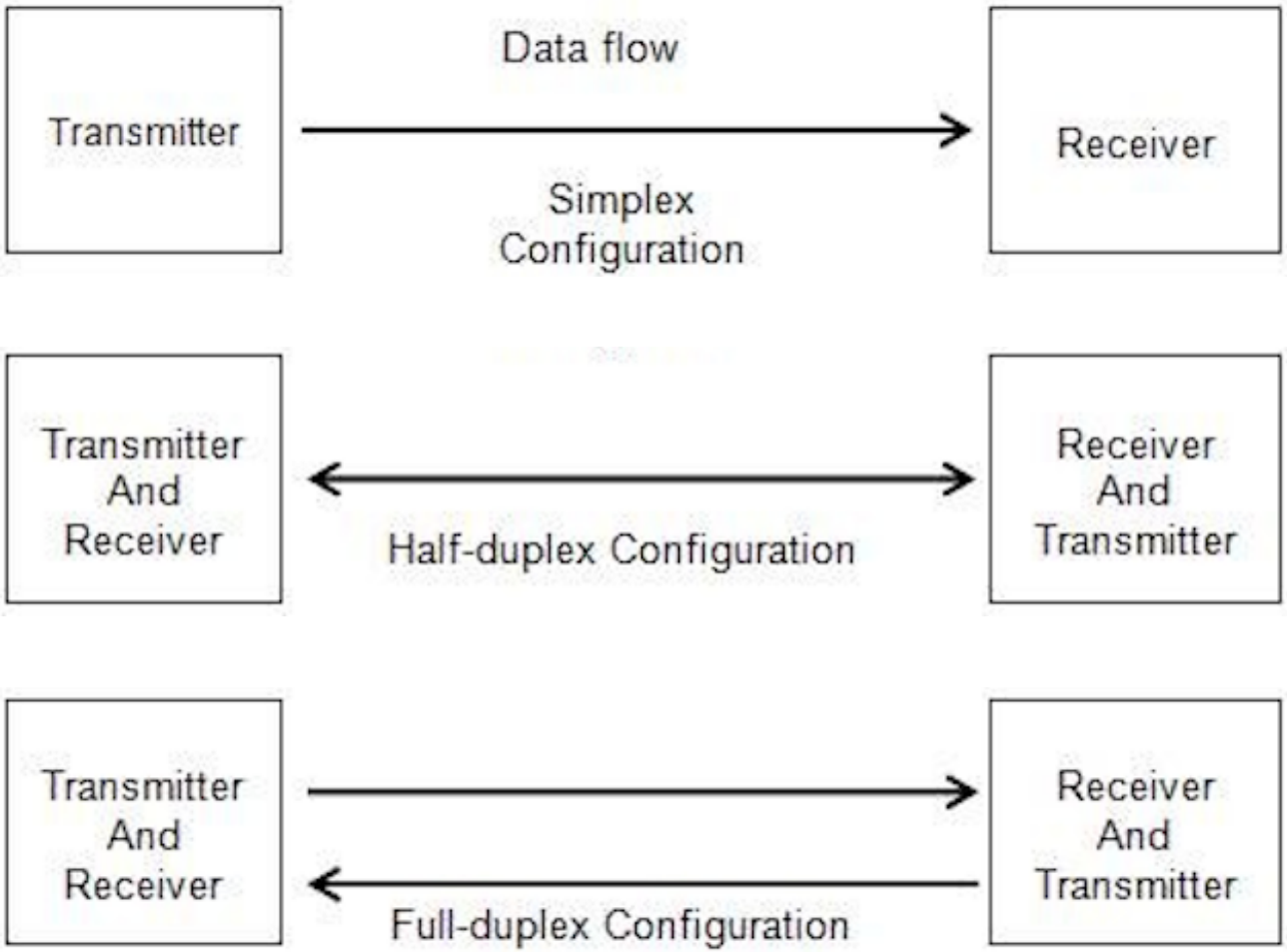
- Data is transmitted sequentially, one bit at a time
- Different synchronization methods can be exploited
- Data can be both sent or received
- The hardware takes care of all the low-level communication

Serial communication





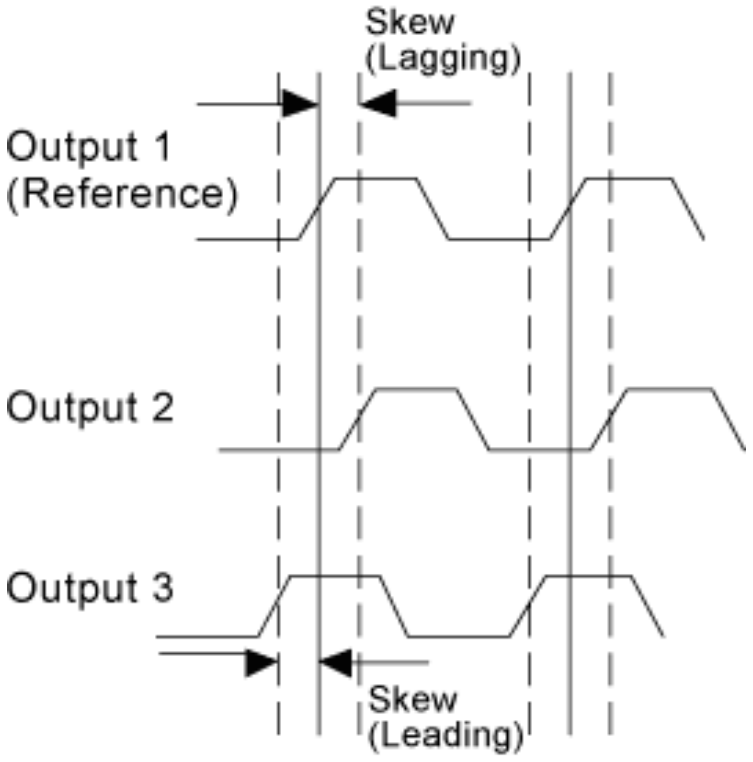
Simplex / Half-duplex / Full-duplex



Clock skew (1/2)

Definition

Clock skew is a phenomenon in synchronous circuits in which the clock signal sent from the clock circuit arrives at different endpoints at different times



Clock skew (2/2)

Clock skew can be caused by:

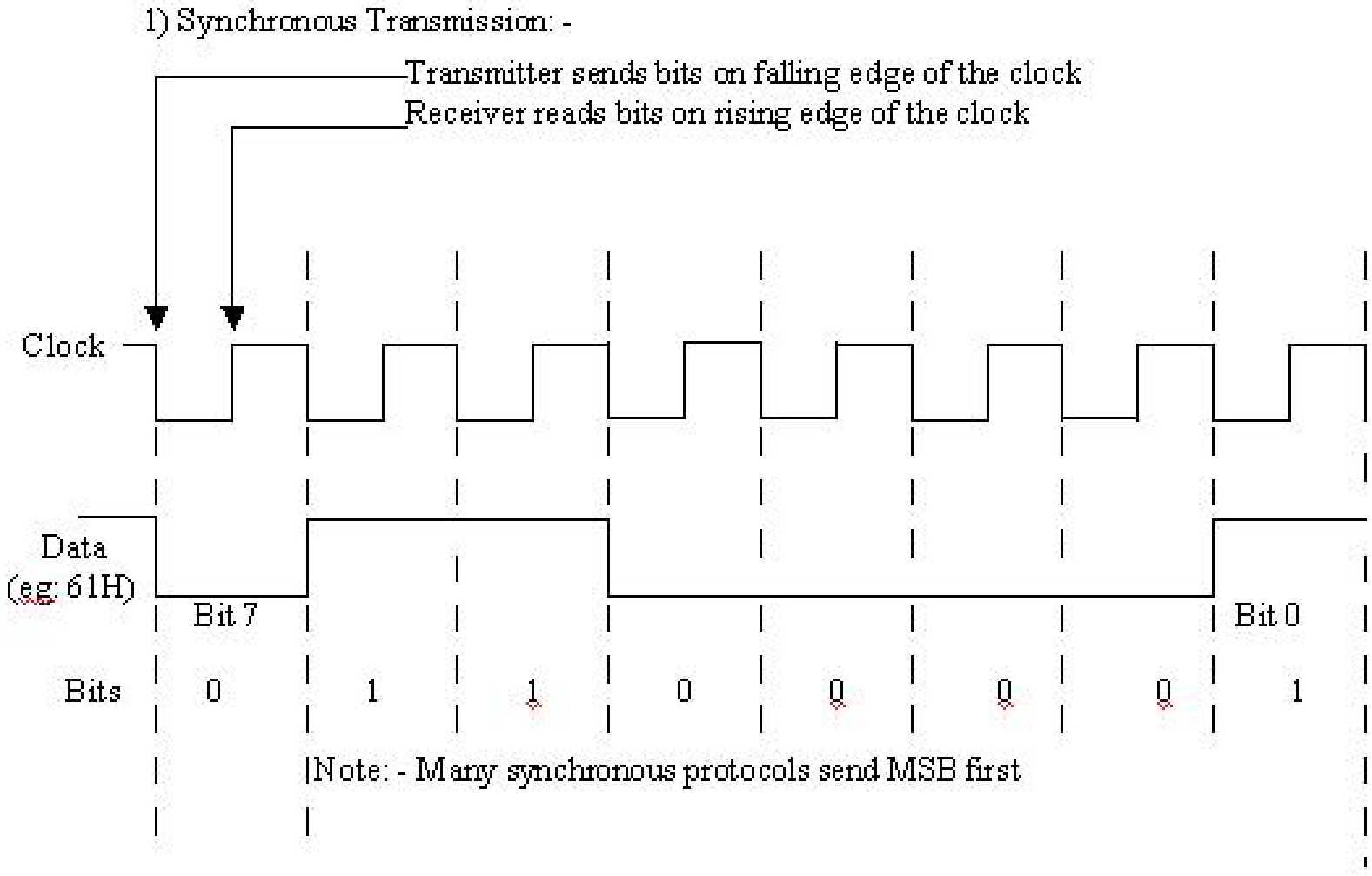
- different wire lengths
- capacitive coupling
- differences in input capacitance
- temperature variations
- variation in intermediate devices

Hold violation: the previous data is not held long enough at the destination flip-flop to be properly clocked through

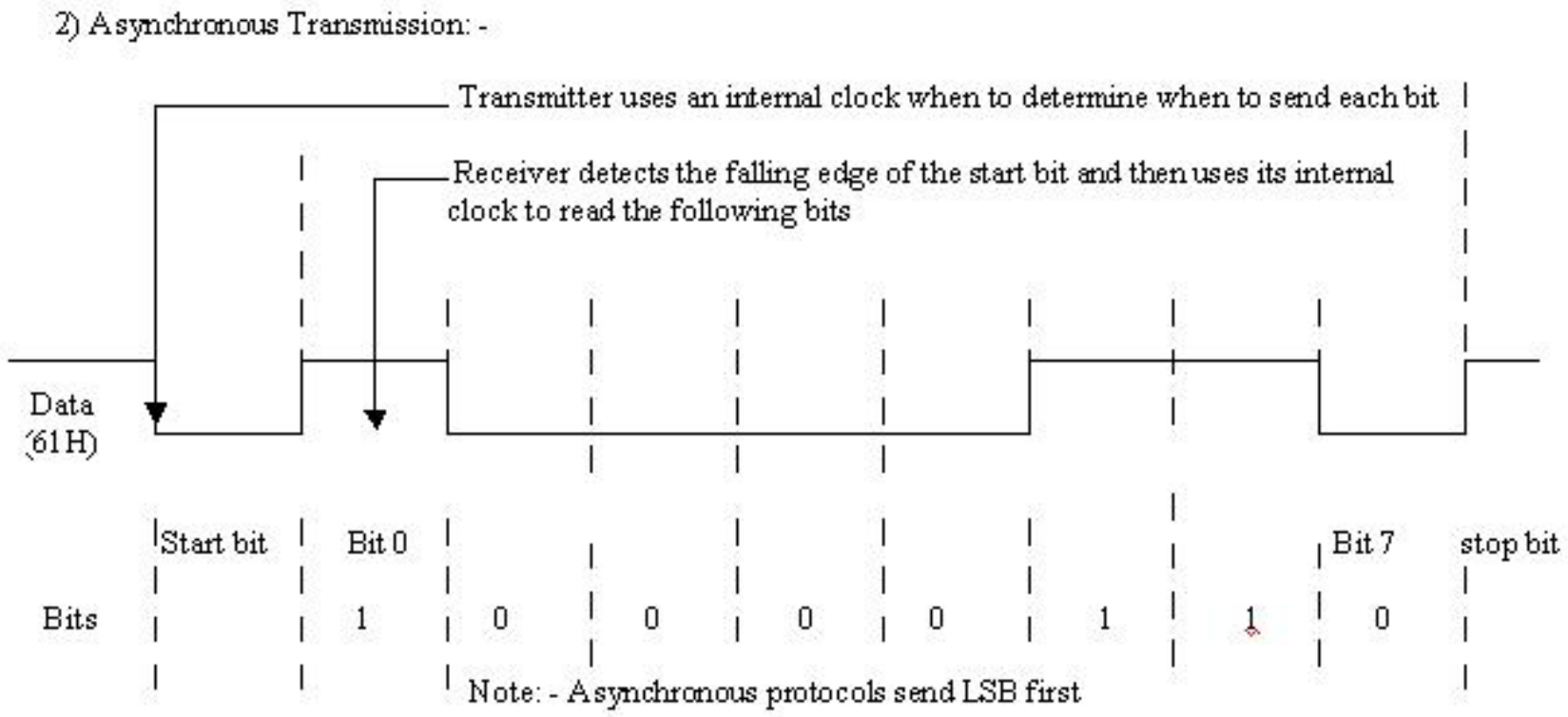
Setup violation: the new data was not set up and stable before the next clock tick arrived

Clock needs to be synchronized!

Synchronous communications



Asynchronous communications



Synchronous vs asynchronous

Synchronous pros:

- The clock is explicit, no need to know it a priori
- Lower overhead
- Greater throughput

Synchronous cons:

- One more wire needed
- Hardware is more expensive

Asynchronous pros:

- Simple and cheap
- The timing is not as critical as for synchronous transmission

Asynchronous cons:

- Clock arbitration needed
- Additional control bits
- Large relative overhead

RS232 standard(s)

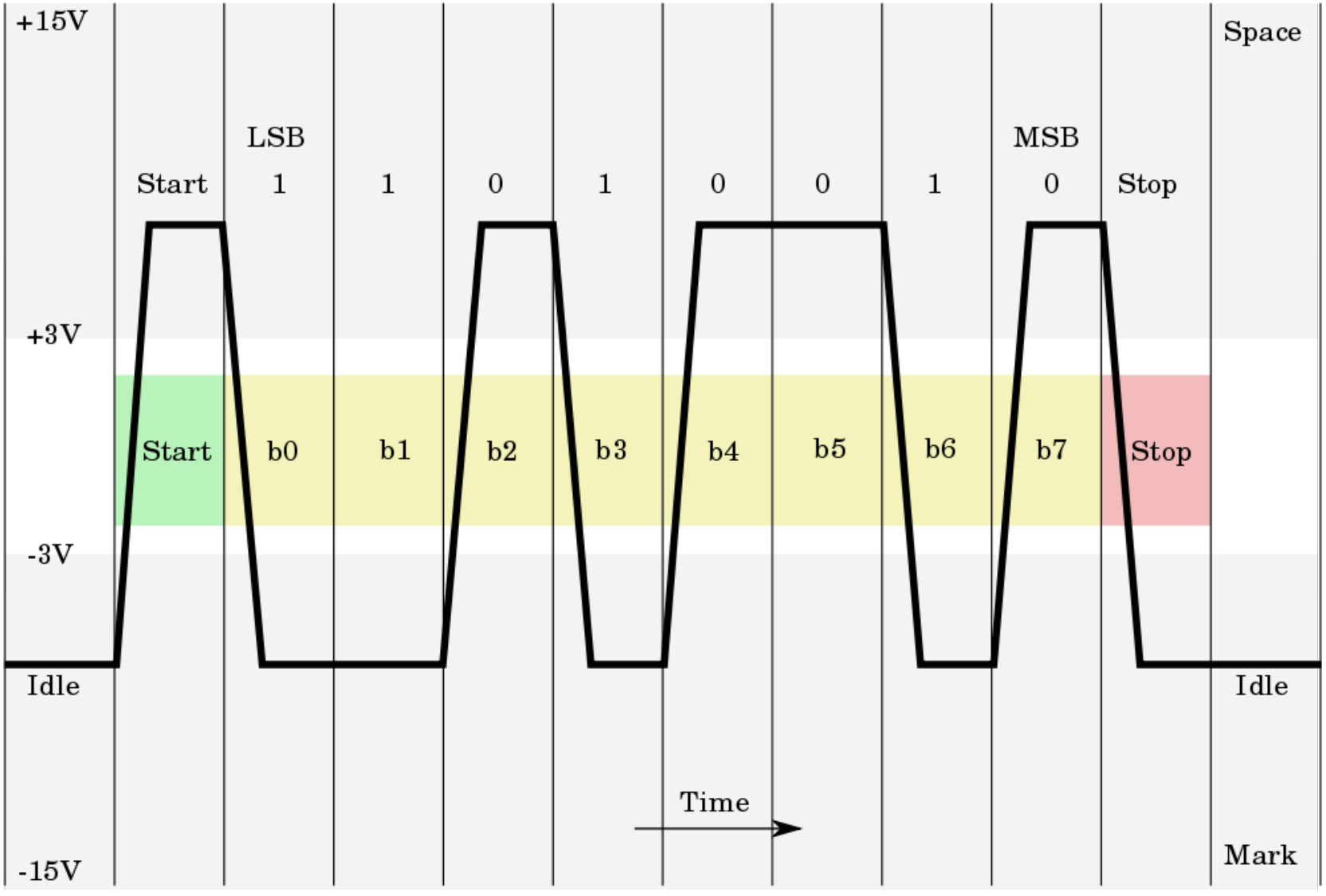
The Electronic Industries Association standard RS-232 defines:

- **electrical characteristics** such as voltage levels, signaling rate, timing and slew-rate of signals
- **mechanical characteristics**, pluggable connectors and pin identification
- **functions of each circuit** in the interface connector
- standard subsets of interface circuits for selected **telecom applications**

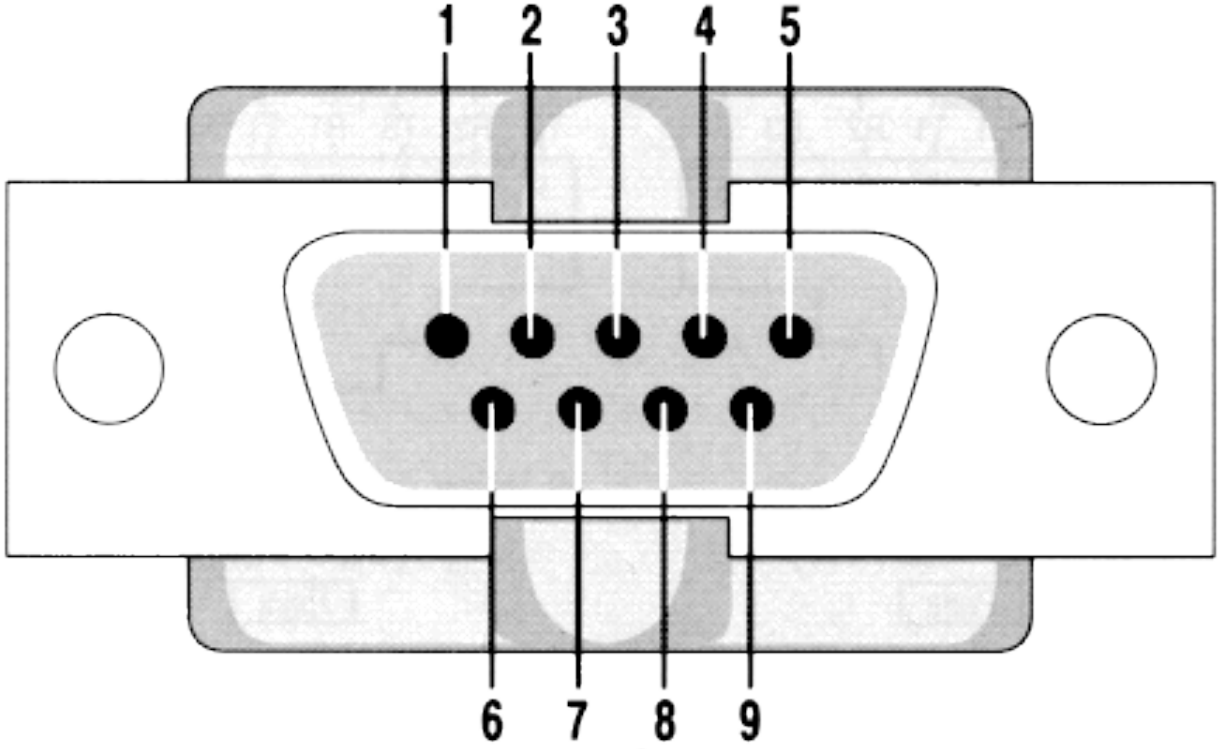
RS-232 standard issues:

- the large voltage swings increases **power consumption**
- single-ended signaling referred to a common signal ground limits the **noise immunity**
- **multi-drop connection** among more than two devices is not defined
- the connector is **huge** (it was DB-25!)

RS232 signals

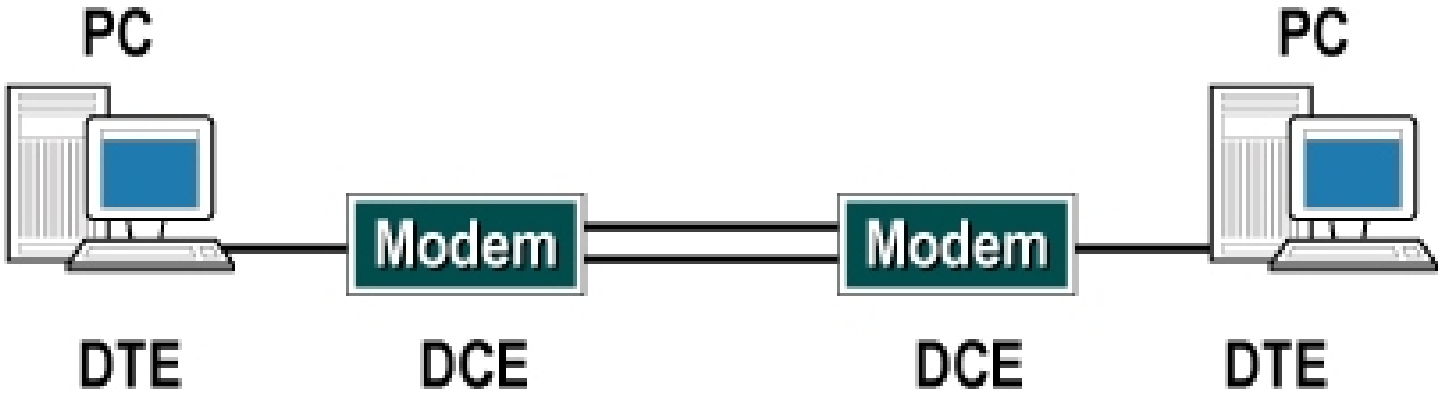


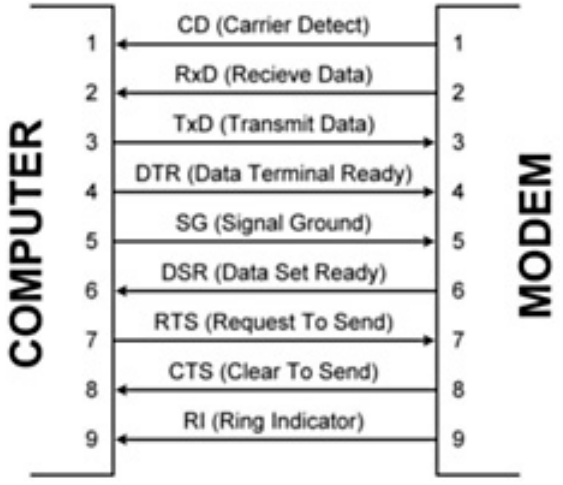
RS232 DB-9 pinout



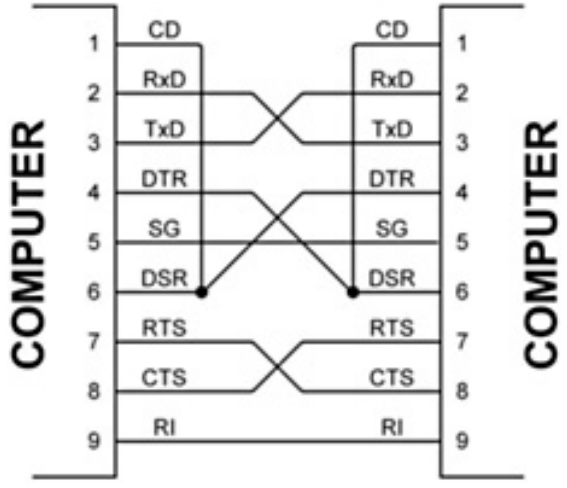
Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

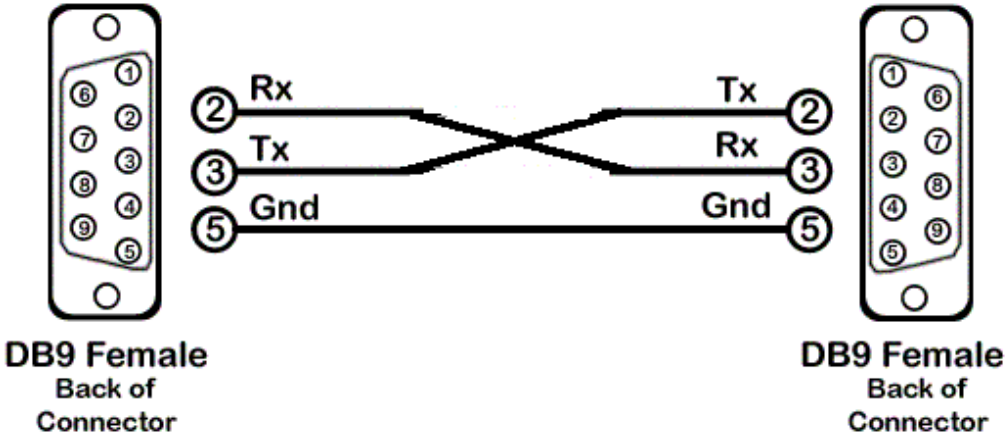
DTE and DCE


PC
PC
Modem
Modem
DTE
DCE
DCE
DTE

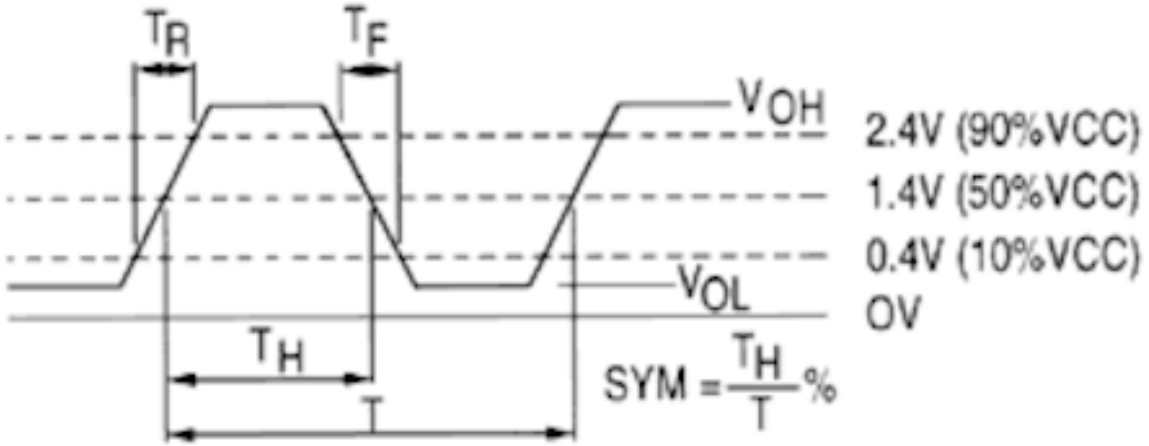

COMPUTER
MODEM
1 CD (Carrier Detect) 1
2 RxD (Recieve Data) 2
3 TxD (Transmit Data) 3
4 DTR (Data Terminal Ready) 4
5 SG (Signal Ground) 5
6 DSR (Data Set Ready) 6
7 RTS (Request To Send) 7
8 CTS (Clear To Send) 8
9 RI (Ring Indicator) 9

Null-modem connection

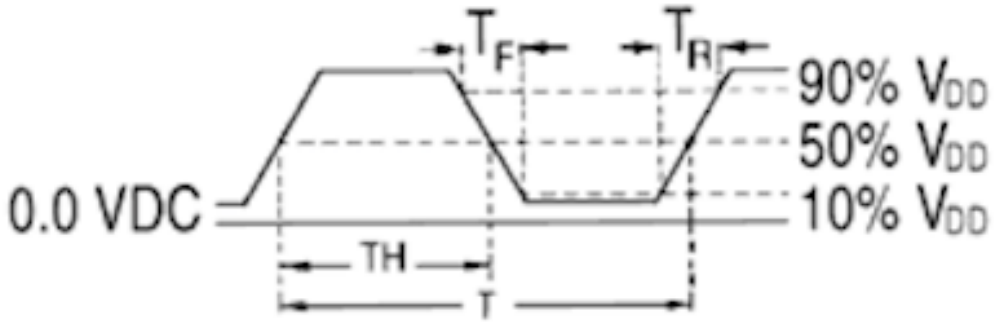




TTL and CMOS signals

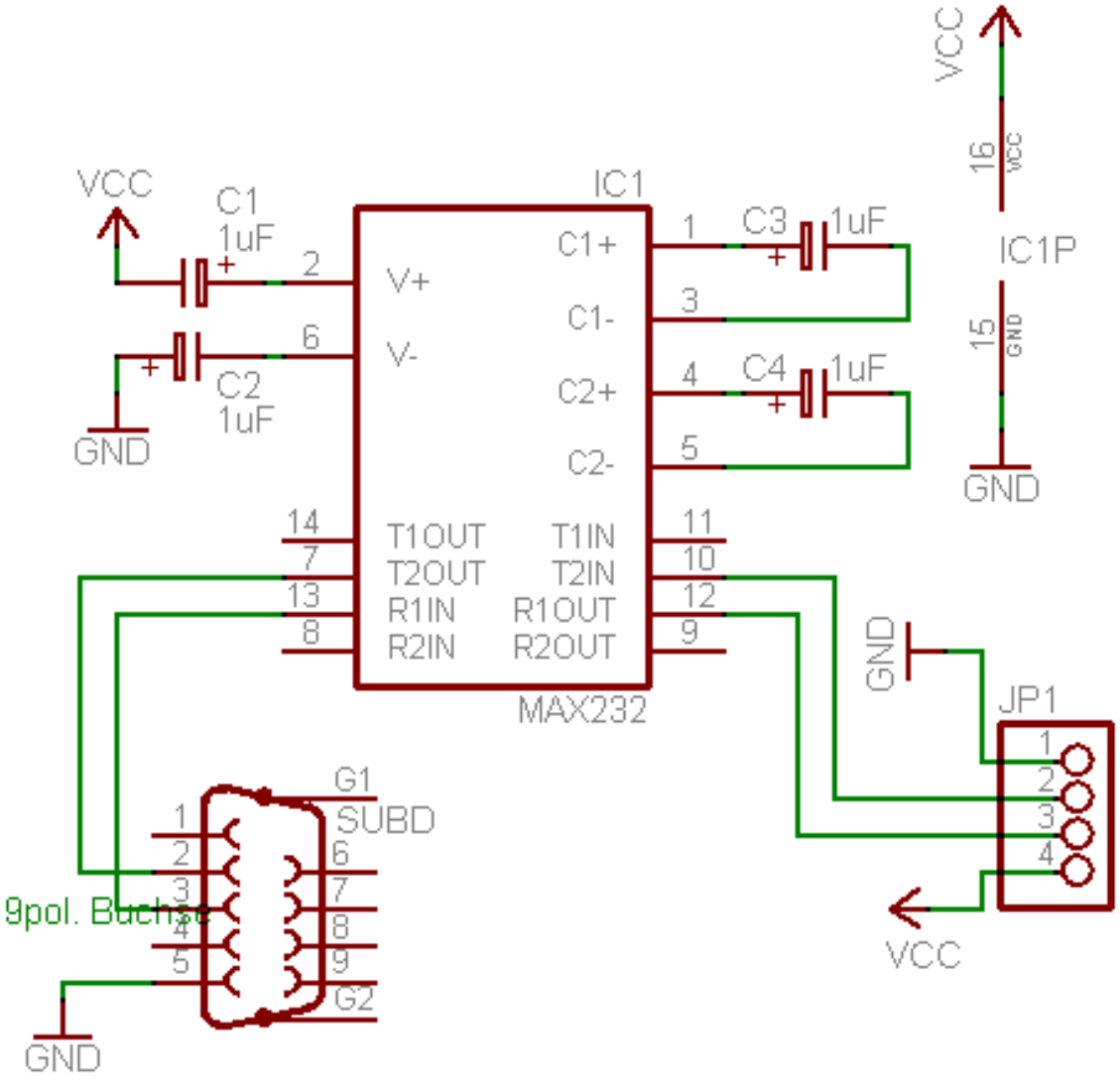


TTL OUTPUT WAVEFORM

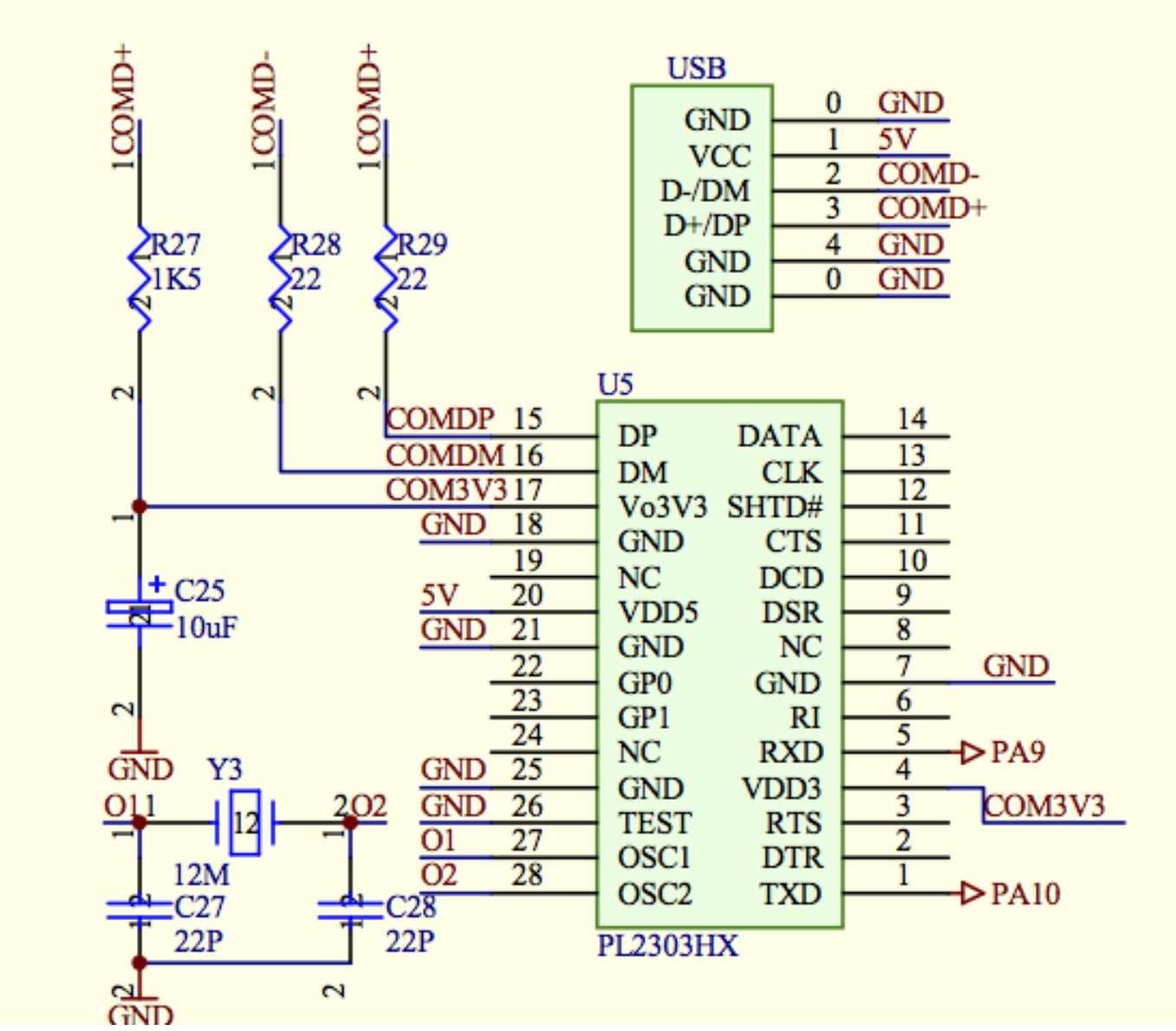


CMOS OUTPUT WAVEFORM

RS-232 / TTL converter



RS-232 / USB converter

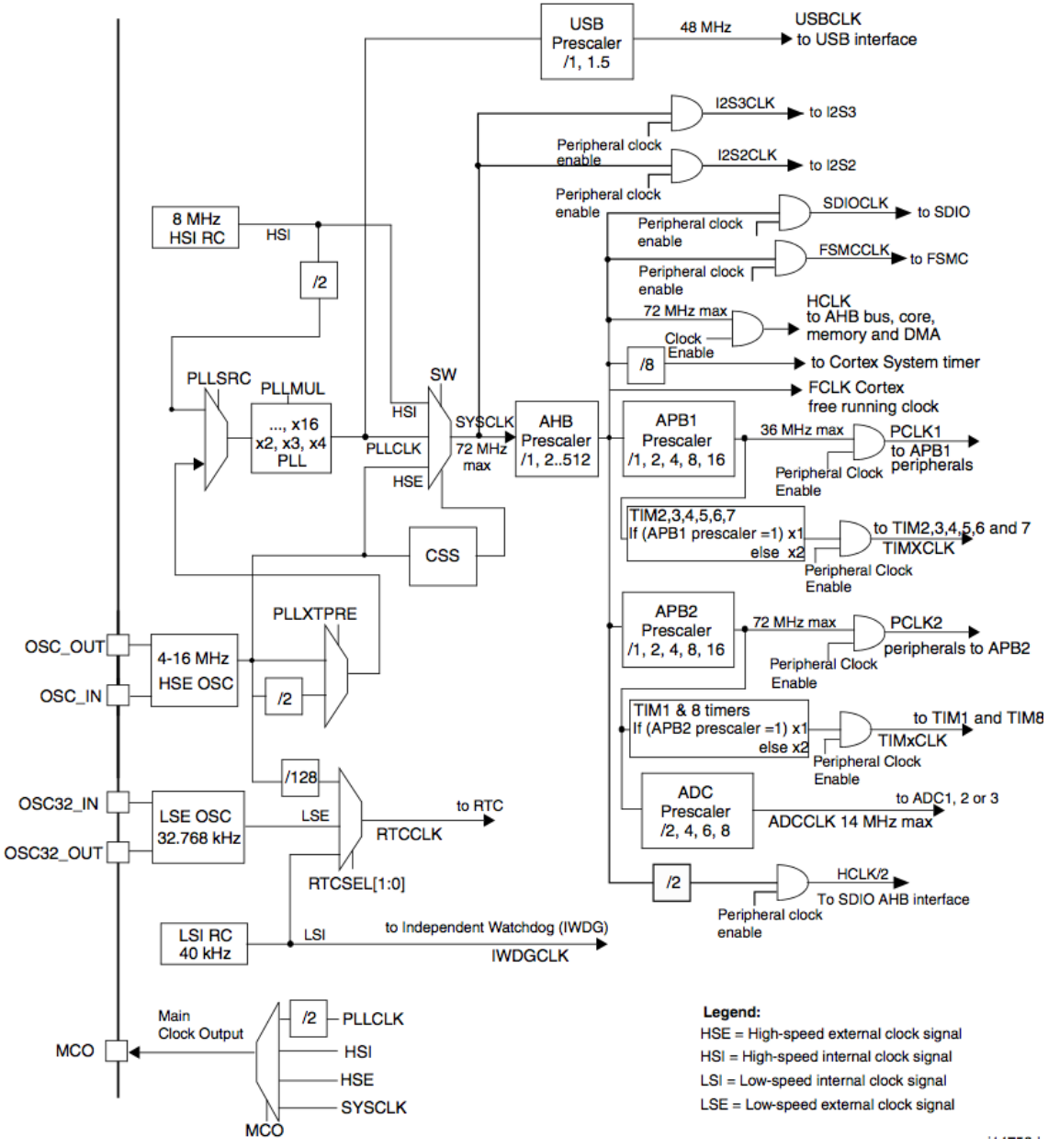


Outline

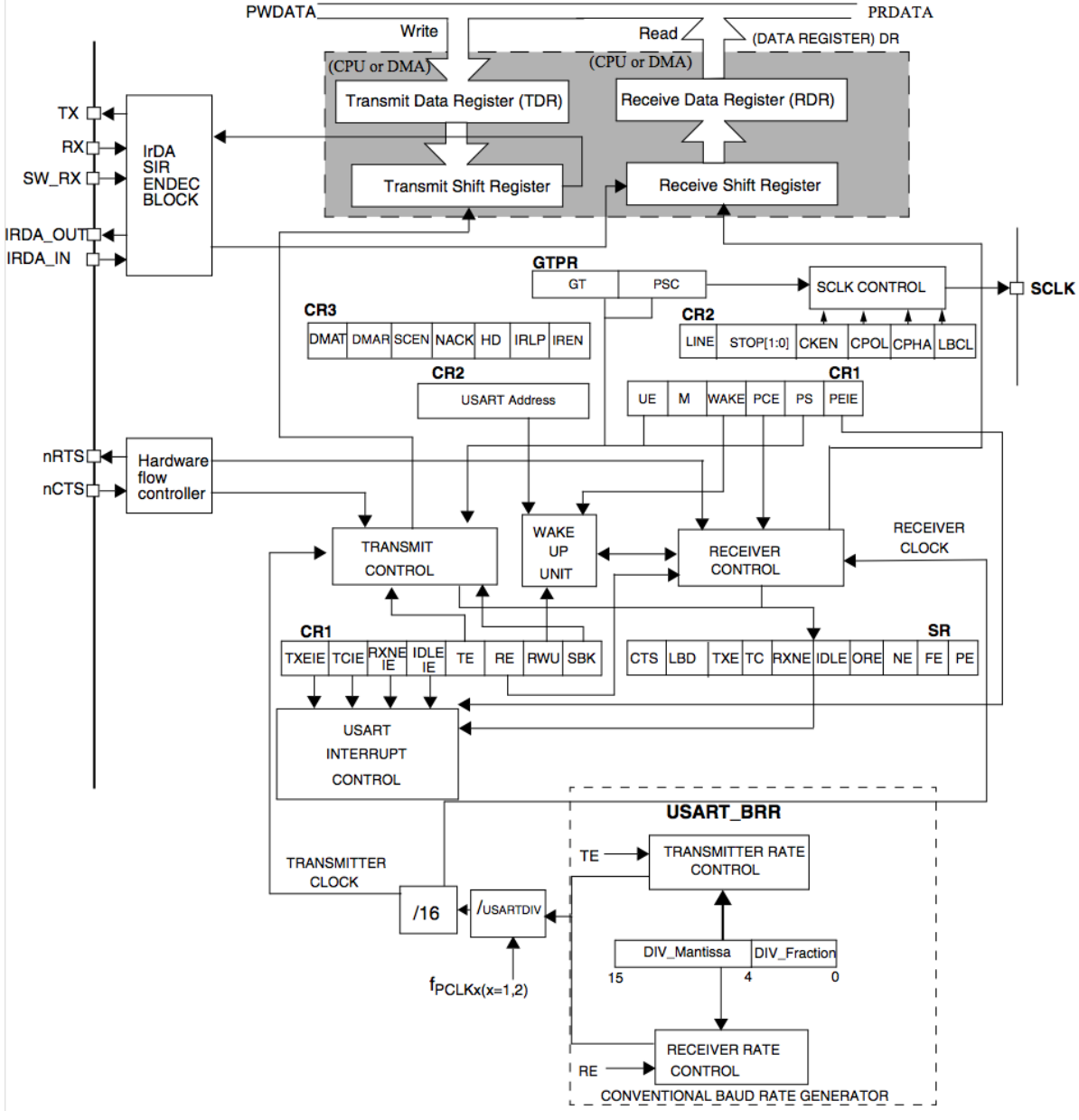
Introduction to STM32 USART

- Full duplex, asynchronous communications
- Transmit and receive baud rates up to 4.5 MBits/s
- Fractional baud rate generator
- Programmable data word length (8 or 9 bits)
- Configurable stop bits - support for 1 or 2 stop bits
- Transmitter clock output for synchronous transmission
- IrDA SIR Encoder Decoder
- Single wire half duplex communication
- Parity control
- Four error detection flags
- Ten interrupt sources with flags

STM32 clock tree



STM32 USART diagram



Fractional baud rate generator

$$f_{BAUD} = \frac{f_{CK}}{16 \times USARTDIV}$$

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register:

- 12-bit mantissa
- 4-bit fraction

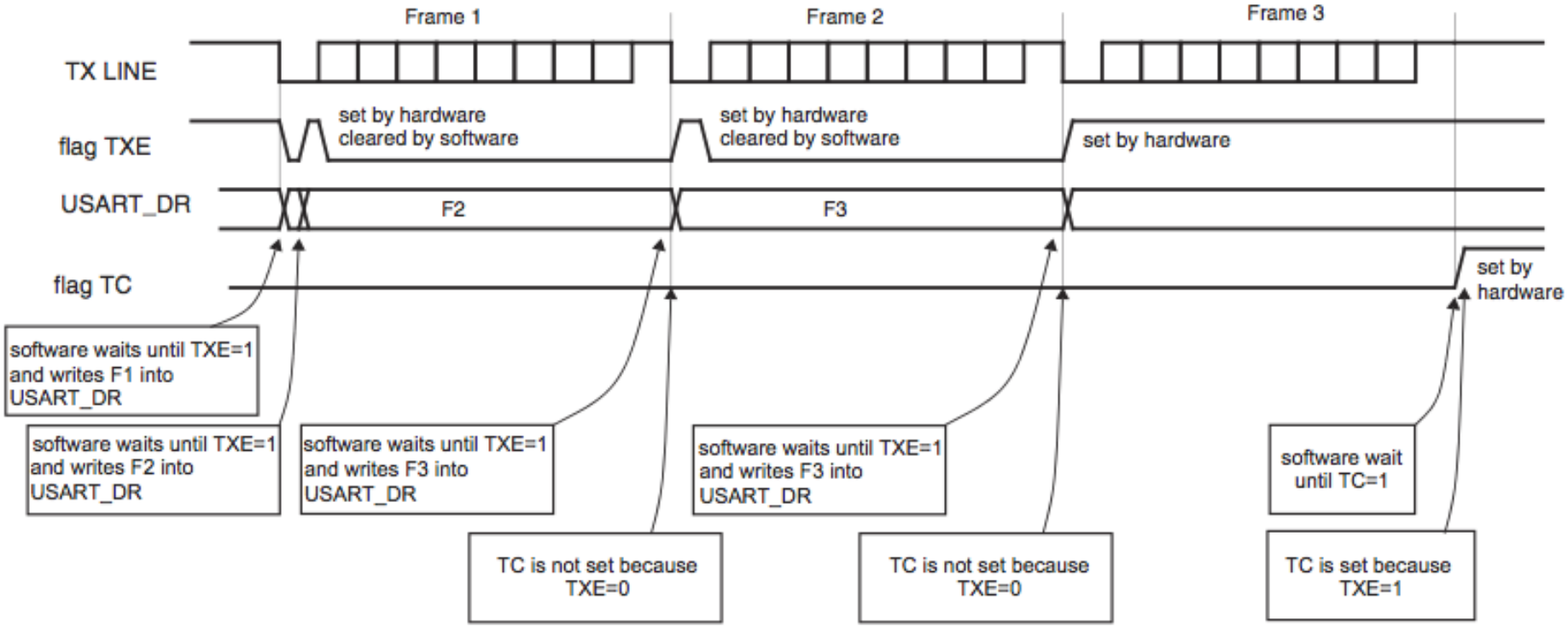
Example:

- $f_{CK} = 72$ Mhz
- mantissa $= 39$
- fraction $= 1 = 1/16$
- $f_{BAUD} = 72$ Mhz / $39.0625 = 115200$ bps

Baud rate error

Baud rate		f_{PCLK} = 36 MHz			f_{PCLK} = 72 MHz		
S.No	in Kbps	Actual	Value programmed in the Baud Rate register	% Error =(Calculated - Desired)B.Rate /Desired B.Rate	Actual	Value programmed in the Baud Rate register	% Error
1.	2.4	2.400	937.5	0%	2.4	1875	0%
2.	9.6	9.600	234.375	0%	9.6	468.75	0%
3.	19.2	19.2	117.1875	0%	19.2	234.375	0%
4.	57.6	57.6	39.0625	0%	57.6	78.125	0.%
5.	115.2	115.384	19.5	0.15%	115.2	39.0625	0%
6.	230.4	230.769	9.75	0.16%	230.769	19.5	0.16%
7.	460.8	461.538	4.875	0.16%	461.538	9.75	0.16%
8.	921.6	923.076	2.4375	0.16%	923.076	4.875	0.16%
9.	2250	2250	1	0%	2250	2	0%
10.	4500	NA	NA	NA	4500	1	0%

USART transmitter



USART receiver

When a character is received:

- the RXNE bit is set as soon as the content of the shift register is transferred to the RDR
- an interrupt is generated if the RXNEIE bit is set
- the error flags can be set if a frame error or an **overrun** error has been detected during reception
- clearing the RXNE bit is performed by a software read to the USART_DR register
- the RXNE flag can also be cleared by writing a zero to it
- the RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error
- if DMA is active, RXNE is set after every byte received and is cleared by the DMA read to the data register

Outline

USART status register (USART_SR)

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	NE	FE	PE
Res.						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

3 **ORE**: Overrun error
This bit is set by hardware when the word currently being received in the shift register is ready to be transferred into the RDR register while RXNE=1
0: No Overrun error
1: Overrun error is detected

5 **RXNE**: Read data register not empty
This bit is set by hardware when the content of the RDR shift register has been transferred to the USART_DR register.
It is cleared by a read to the USART_DR register.
0: Data is not received
1: Received data is ready to be read

6 **TC**: Transmission complete
This bit is set by hardware if the transmission of a frame containing data is complete and if TXE is set.

7 **TXE**: Transmit data register empty
This bit is set by hardware when the content of the TDR register has been transferred into the shift register. It is cleared by a write to the USART_DR register.

USART control register 1 (USART_CR1)

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		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
Res.		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

2 **RE**: Receiver enable
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

3 **TE**: Transmitter enable
0: Transmitter is disabled
1: Transmitter is enabled

5 **RXNEIE**: RXNE interrupt enable
0: Interrupt is inhibited
1: An USART interrupt is generated whenever ORE=1 or RXNE=1 in the USART_SR register

12 **M**: Word length
0: 1 Start bit, 8 Data bits, n Stop bit
1: 1 Start bit, 9 Data bits, n Stop bit

13 **UE**: USART enable
0: USART outputs disabled
1: USART enabled

USART *value* registers

USART baud rate register

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
DIV_Mantissa[11:0]												DIV_Fraction[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15:4 **DIV_Mantissa[11:0]**: mantissa of USARTDIV
These 12 bits define the mantissa of the USART Divider (USARTDIV).

3:0 **DIV_Fraction[3:0]**: fraction of USARTDIV
These 4 bits define the fraction of the USART Divider (USARTDIV)

USART data register

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							DR[8:0]								
Res.							rw	rw	rw	rw	rw	rw	rw	rw	rw

Outline

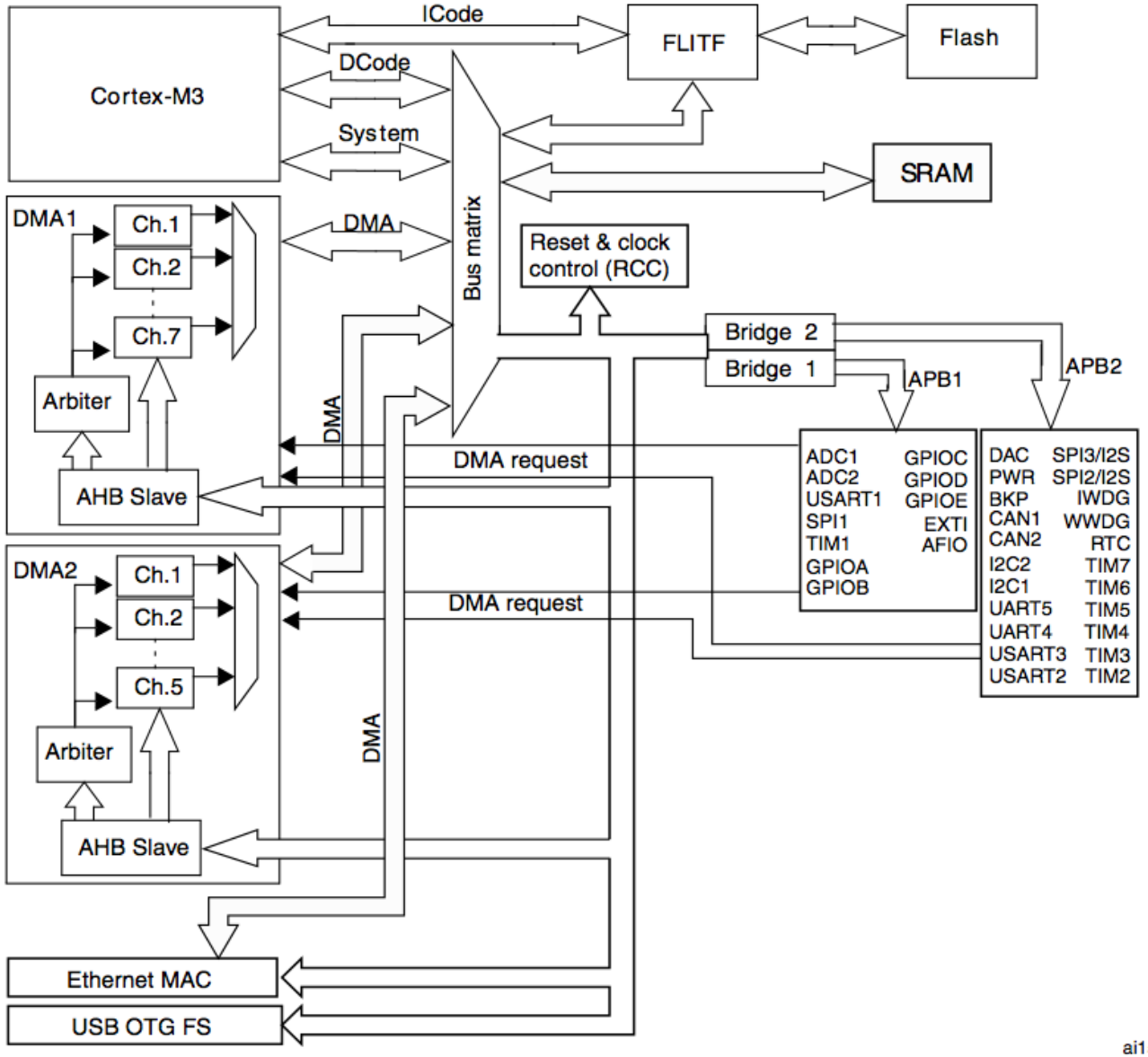
Direct memory access

Definition

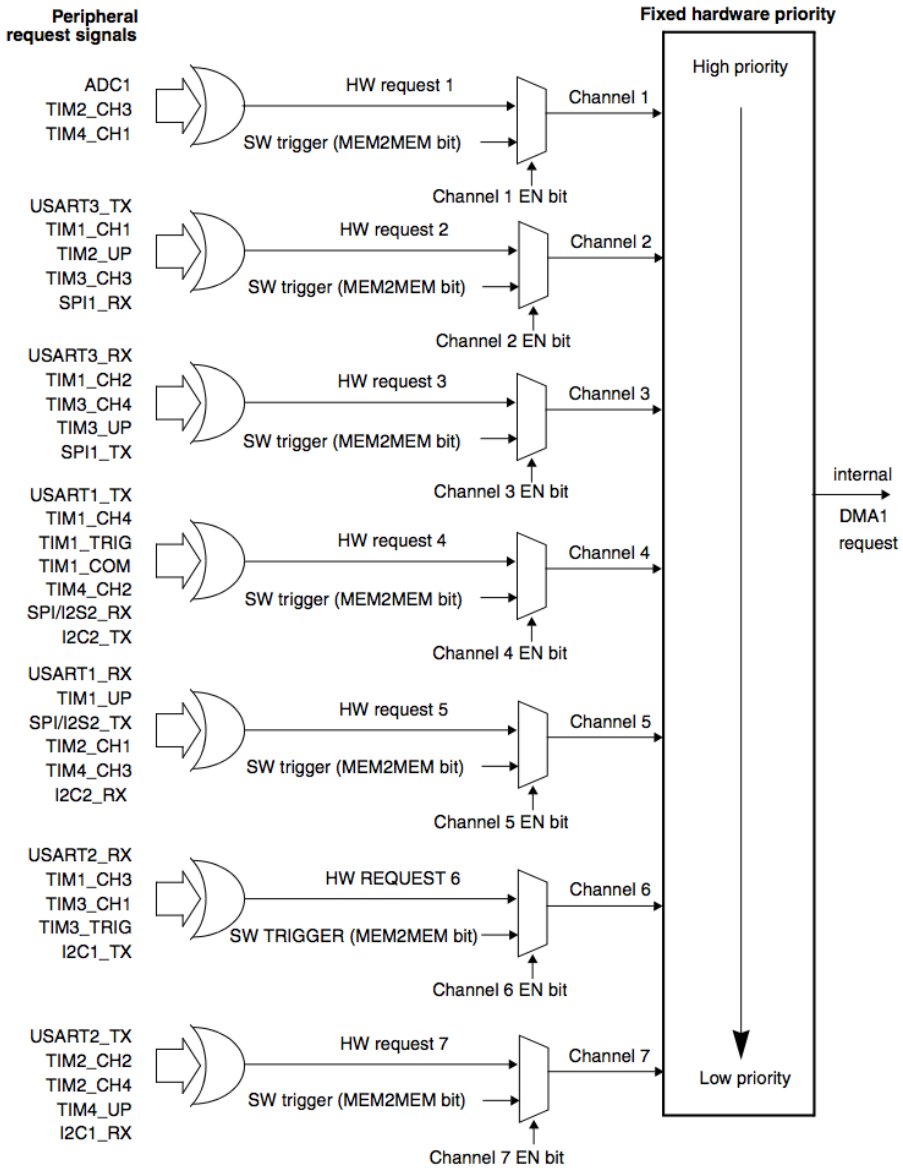
Direct memory access is used in order to provide **high-speed data transfer** between peripherals and memory as well as memory to memory **without any CPU actions**.

- 12 independently configurable channels
- Dedicated hardware DMA requests or software triggers
- Software programmable priorities
- Support for circular buffer management
- 3 event flags: DMA Half Transfer, DMA Transfer complete and DMA Transfer Error
- Memory-to-memory transfer
- Peripheral-to-memory and memory-to-peripheral, and peripheral-to-peripheral transfers
- Access to Flash, SRAM, peripheral SRAM, APB1, APB2 and AHB peripherals as source and destination

DMA diagram


Cortex-M3
ICode
DCode
System
FLITF
Flash
SRAM
Bus matrix
DMA
Reset & clock control (RCC)
Bridge 2
Bridge 1
APB1
APB2
DMA1
Ch.1
Ch.2
Ch.7
Arbiter
AHB Slave
DMA request
DMA2
Ch.1
Ch.2
Ch.5
Arbiter
AHB Slave
DMA request
ADC1
ADC2
USART1
SPI1
TIM1
GPIOA
GPIOB
GPIOC
GPIOD
GPIOE
EXTI
AFIO
DAC
PWR
BKP
CAN1
CAN2
I2C2
I2C1
UART5
UART4
USART3
USART2
SPI3/I2S
SPI2/I2S
IWDG
WWDG
RTC
TIM7
TIM6
TIM5
TIM4
TIM3
TIM2
Ethernet MAC
USB OTG FS

DMA1 request mapping



DMA1 channels

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
ADC1	ADC1						
SPI/I^2S		SPI1_RX	SPI1_TX	SPI/I2S2_RX	SPI/I2S2_TX		
USART		USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I^2C				I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1		TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
TIM2	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2 TIM2_CH4
TIM3		TIM3_CH3	TIM3_CH4 TIM3_UP			TIM3_CH1 TIM3_TRIG	
TIM4	TIM4_CH1			TIM4_CH2	TIM4_CH3		TIM4_UP

DMA2 channels

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
ADC3(1)					ADC3
SPI/I2S3	SPI/I2S3_RX	SPI/I2S3_TX			
UART4			UART4_RX		UART4_TX
SDIO(1)				SDIO	
TIM5	TIM5_CH4 TIM5_TRIG	TIM5_CH3 TIM5_UP		TIM5_CH2	TIM5_CH1
TIM6/ DAC_Channel1			TIM6_UP/ DAC_Channel1		
TIM7/ DAC_Channel2				TIM7_UP/ DAC_Channel2	
TIM8(1)	TIM8_CH3 TIM8_UP	TIM8_CH4 TIM8_TRIG TIM8_COM	TIM8_CH1		TIM8_CH2

DMA1 channels

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
ADC1	ADC1						
SPI/I^2S		SPI1_RX	SPI1_TX	SPI/I2S2_RX	SPI/I2S2_TX		
USART		USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I^2C				I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1		TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
TIM2	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2 TIM2_CH4
TIM3		TIM3_CH3	TIM3_CH4 TIM3_UP			TIM3_CH1 TIM3_TRIG	
TIM4	TIM4_CH1			TIM4_CH2	TIM4_CH3		TIM4_UP

DMA channel x configuration register (DMA_CCRx)

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.	MEM2 MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

0 **EN**: Channel enable
0: Channel disabled
1: Channel enabled

1 **TCIE**:Transfer complete interrupt enable
0: TC interrupt disabled 1: TC interrupt enabled

4 **DIR**: Data transfer direction
0: Read from peripheral
1: Read from memory

5 **CIRC**: Circular mode

6 **PINC**: Peripheral increment mode

7 **MINC**: Memory increment mode

5 **MEM2MEM**: Memory to memory mode

DMA *value* registers

- DMA channel x number of data register (DMA_CNDTRx)

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
NDT															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- DMA channel x peripheral address register (DMA_CPARx)

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

- DMA channel x memory address register (DMA_CMARx)

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

References

http://en.wikipedia.org/wiki/Serial_communication

http://en.wikipedia.org/wiki/Clock_skew

http://en.wikipedia.org/wiki/Flip-flop_(electronics)

http://en.wikipedia.org/wiki/RS-232

http://en.wikipedia.org/wiki/Direct_memory_access

STM32F10xxx Reference Manual (RM0008 - Doc ID 14611)

Using the STM32F101xx and STM32F103xx DMA controller (AN2548 - Doc ID 13529)

Communication peripheral FIFO emulation with DMA and DMA timeout in STM32F10x microcontrollers (AN3109 - Doc ID 16795)