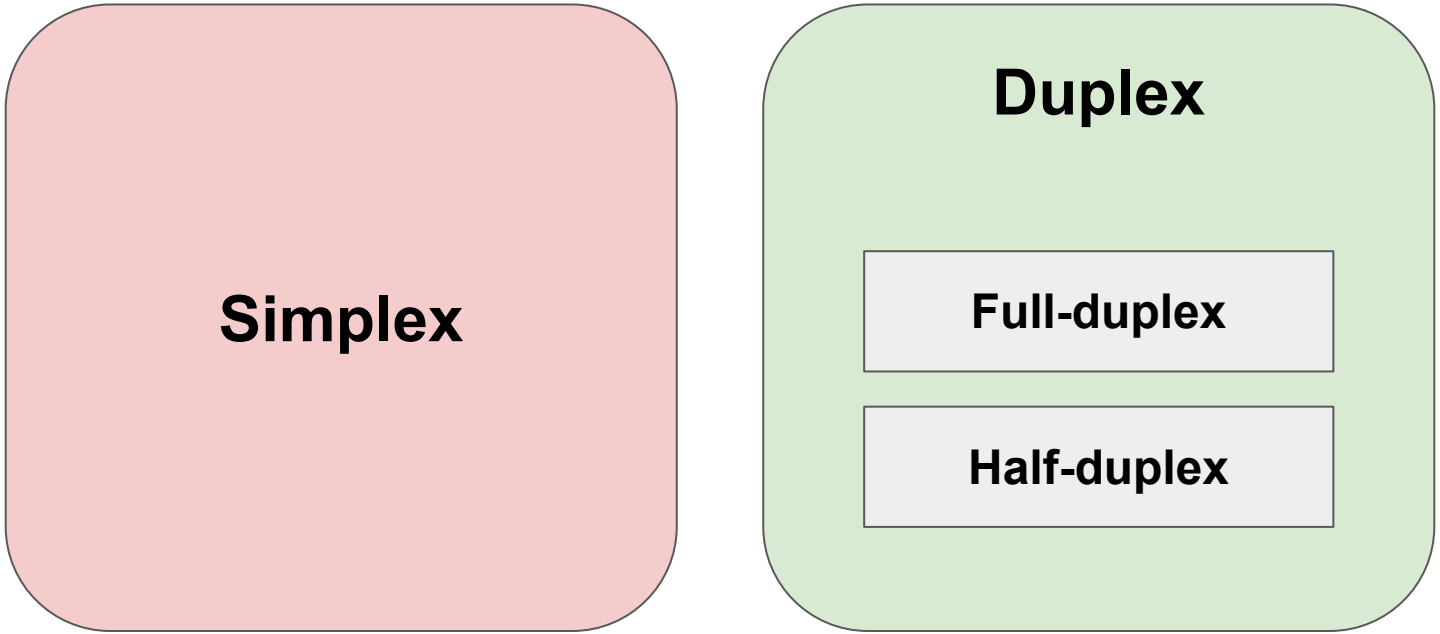
Introduction to embedded programming on STM32

Hello world!

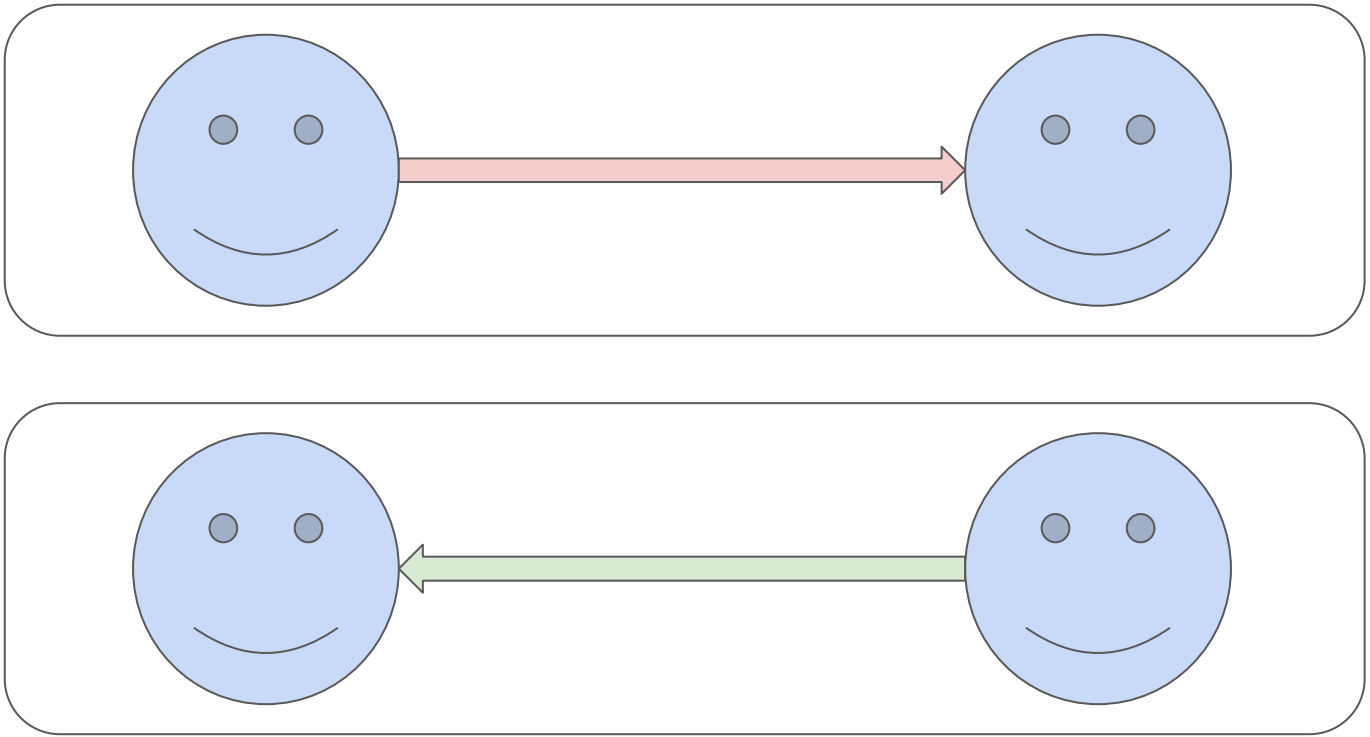
Skoltech, 2019

Type of communication service

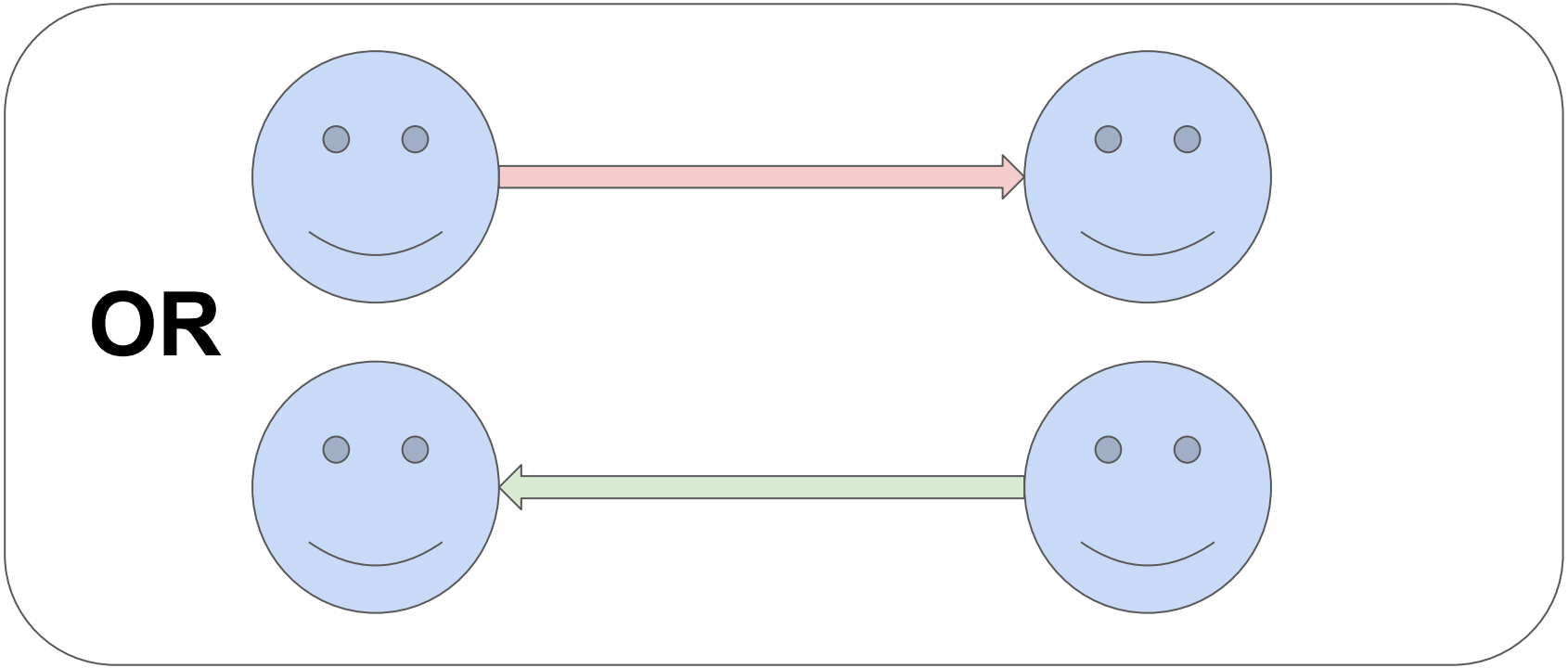


Simplex communication

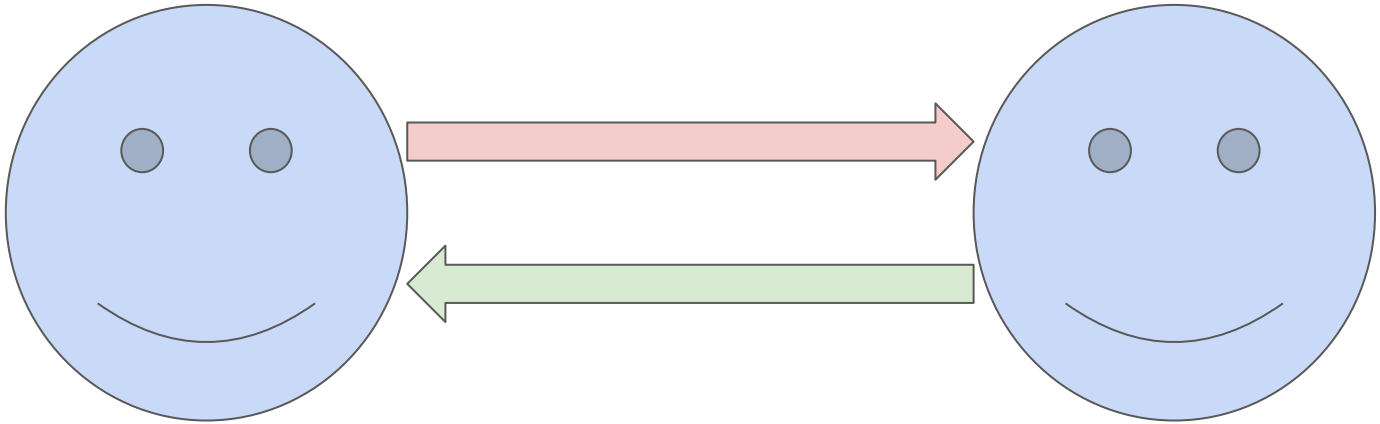
OR



Half-duplex communication

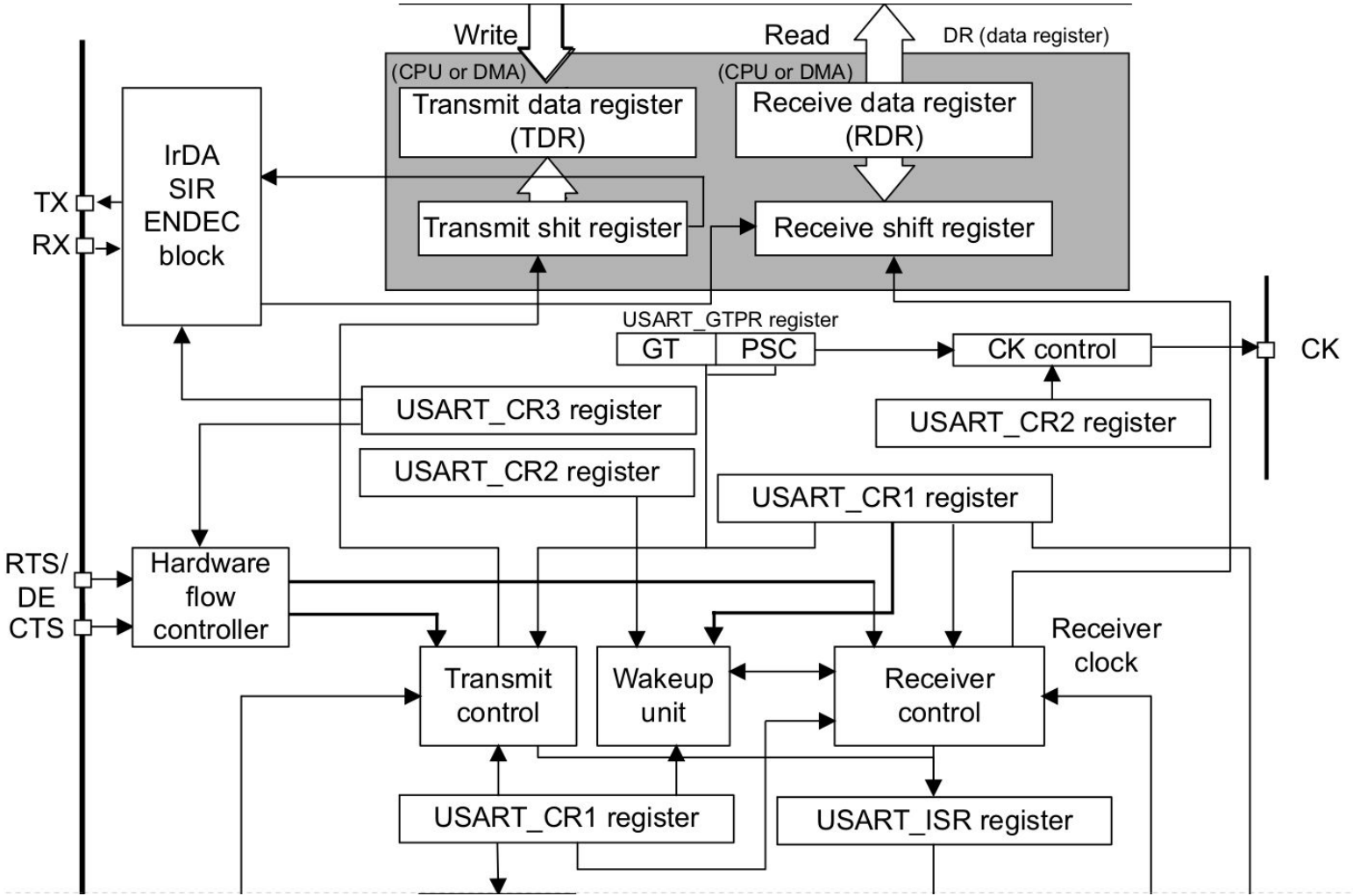


Full-duplex communication

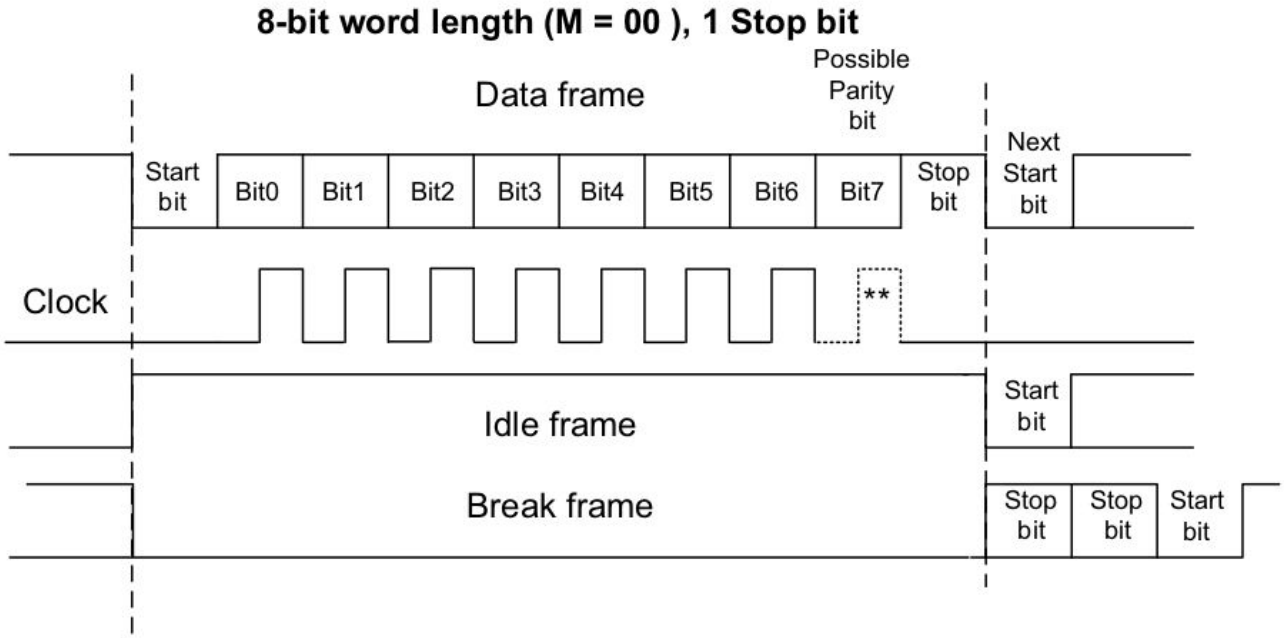


Universal synchronous asynchronous receiver transmitter (USART). Main features

- Full-duplex asynchronous communications
- Baud rate of up to 6 Mbit/s
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire Half-duplex communications
- Continuous communications using DMA
- Swappable Tx/Rx pin configuration


Write
Read
DR (data register)
(CPU or DMA)
(CPU or DMA)
Transmit data register (TDR)
Receive data register (RDR)
IrDA SIR ENDEC block
TX
RX
Transmit shit register
Receive shift register
USART_GTPR register
GT
PSC
CK control
CK
USART_CR3 register
USART_CR2 register
USART_CR2 register
USART_CR1 register
RTS/ DE
CTS
Hardware flow controller
Receiver clock
Transmit control
Wakeup unit
Receiver control
USART_CR1 register
USART_ISR register

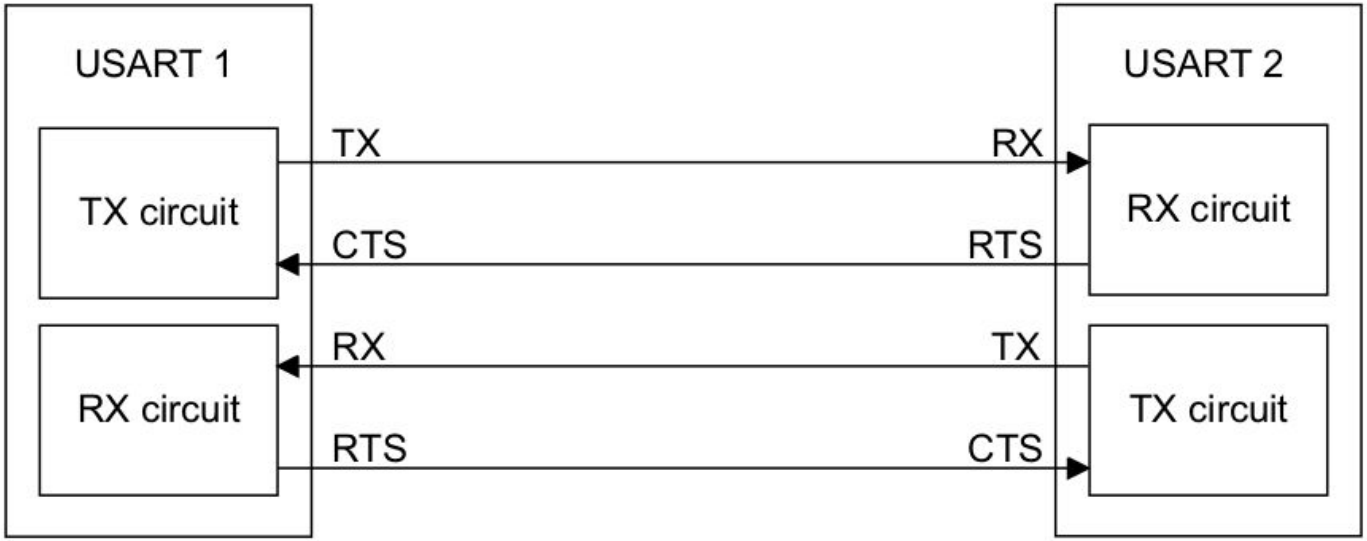
USART. Output waveform



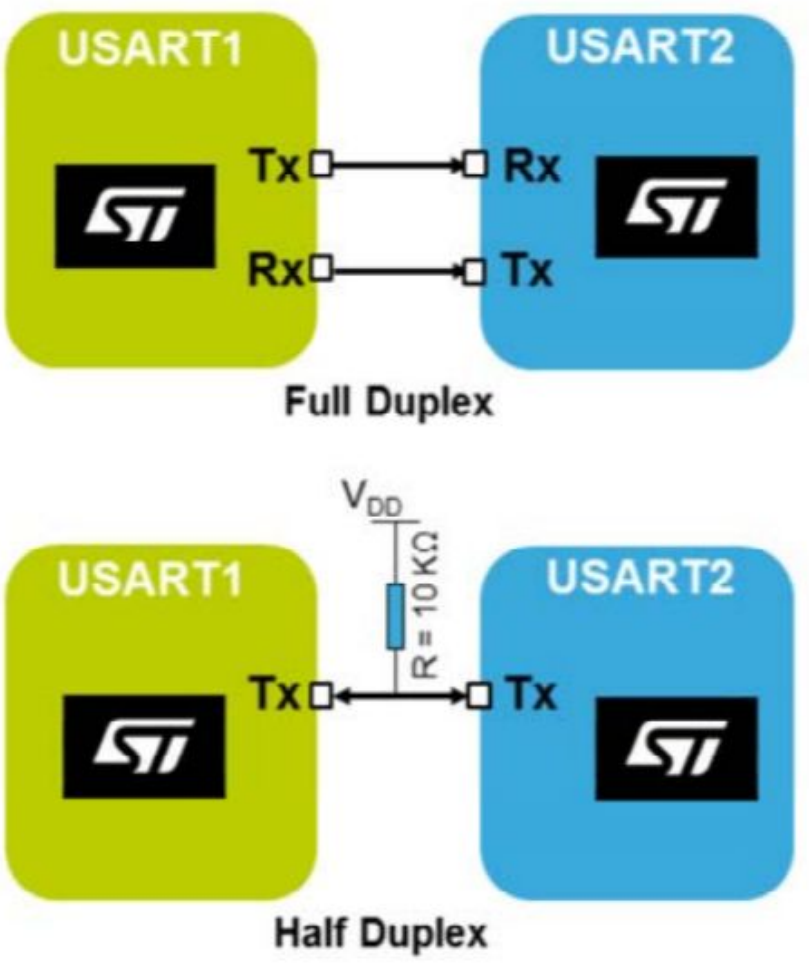
USART. Specific features

- Hardware flow control for modem
- Continuous communication using DMA
- Multiprocessor communication
- Synchronous mode
- Smartcard mode
- Single-wire Half-duplex communication
- IrDA SIR ENDEC block
- LIN mode
- Modbus communication
- Auto baud rate detection
- Receiver timeout interrupt

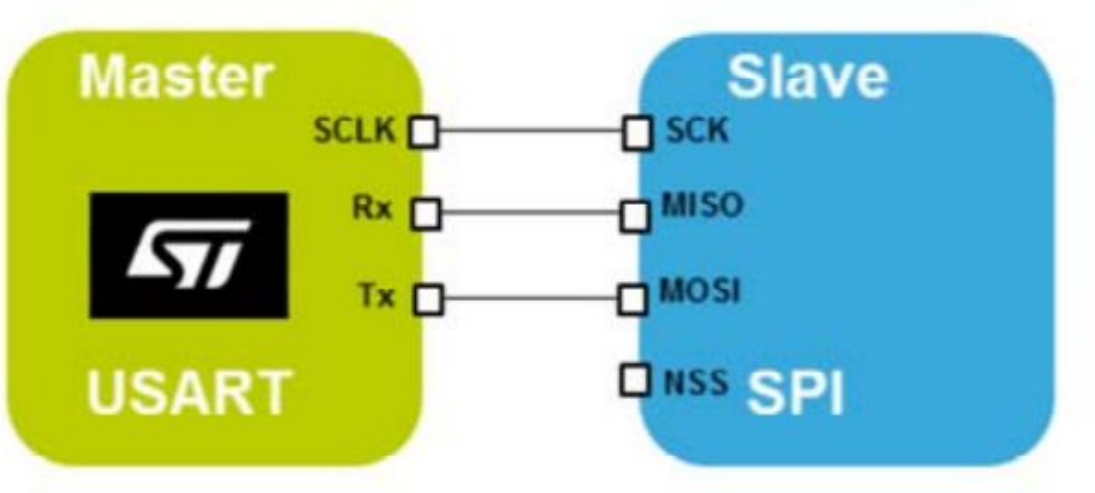
USART. Hardware flow control



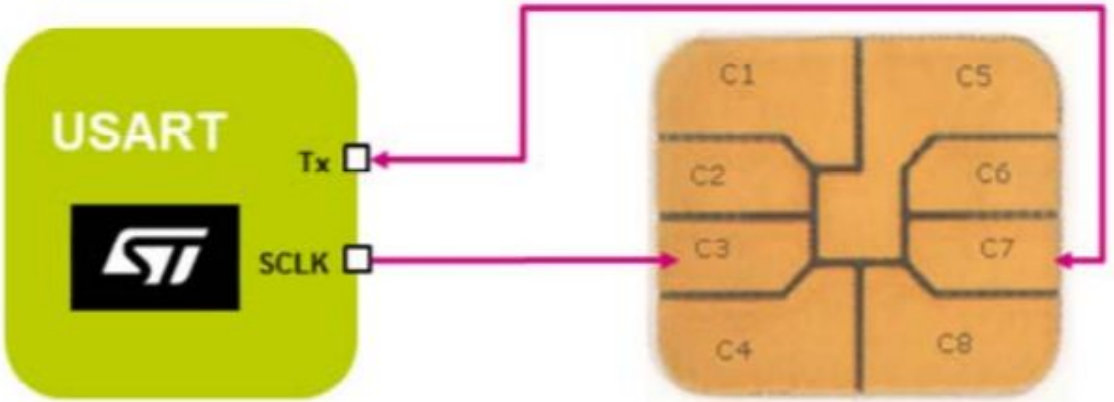
USART. Full/half duplex modes



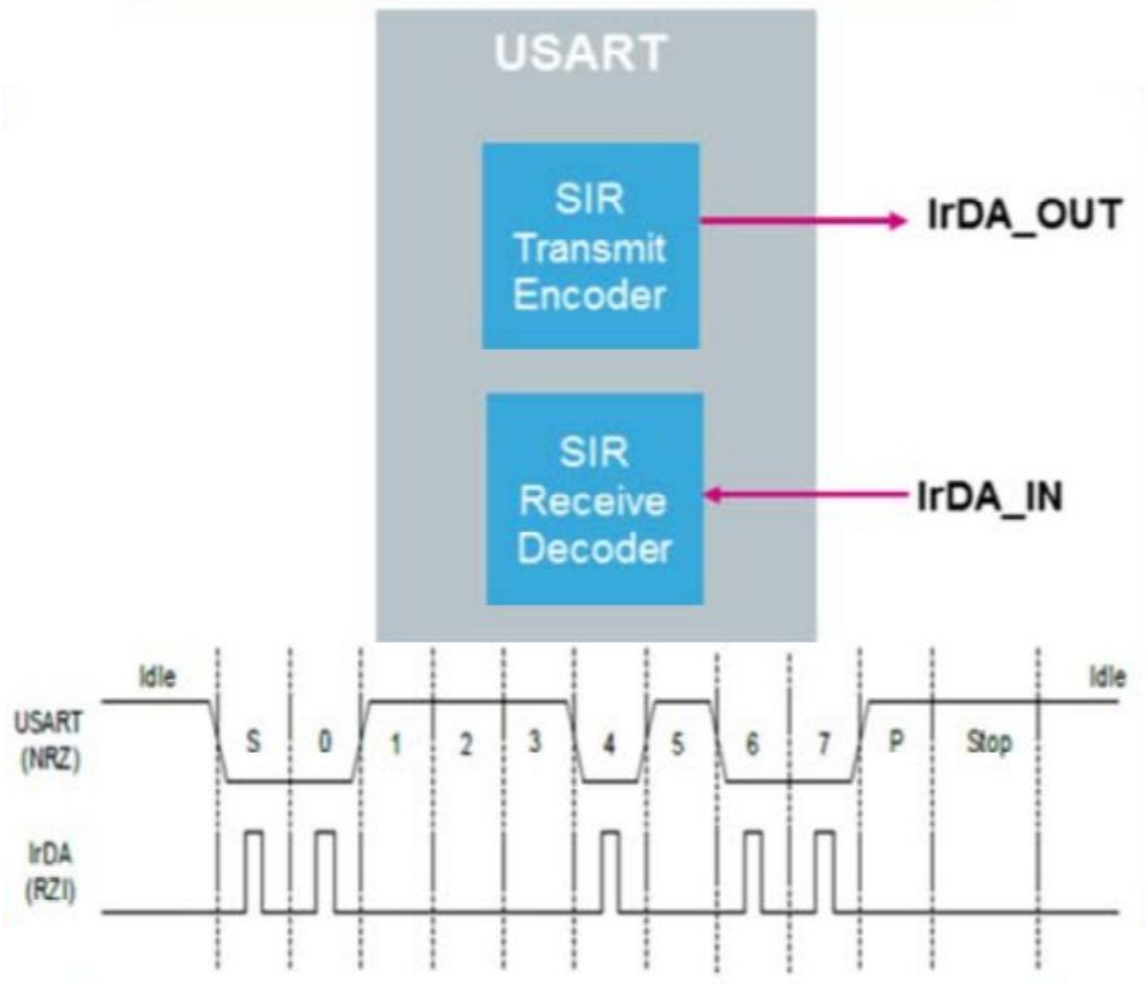
USART. SPI mode



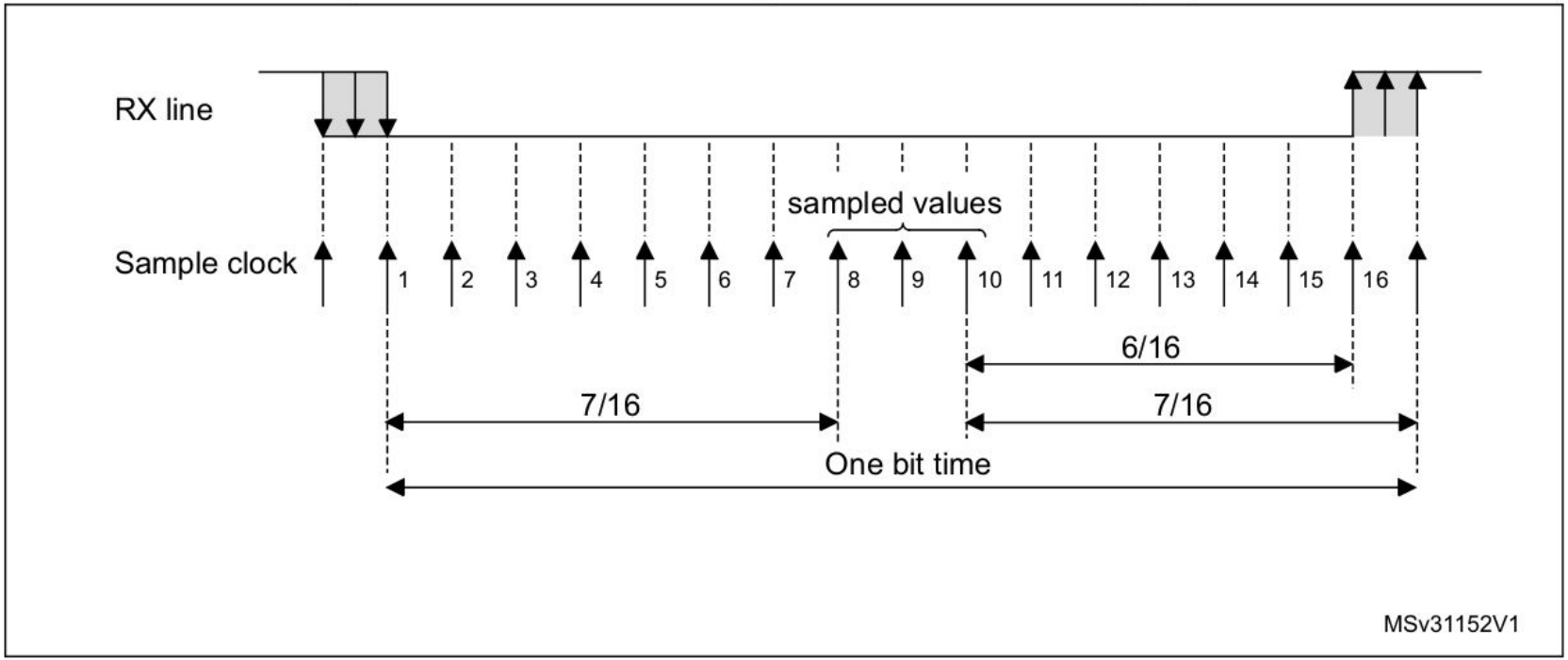
USART. Smartcards



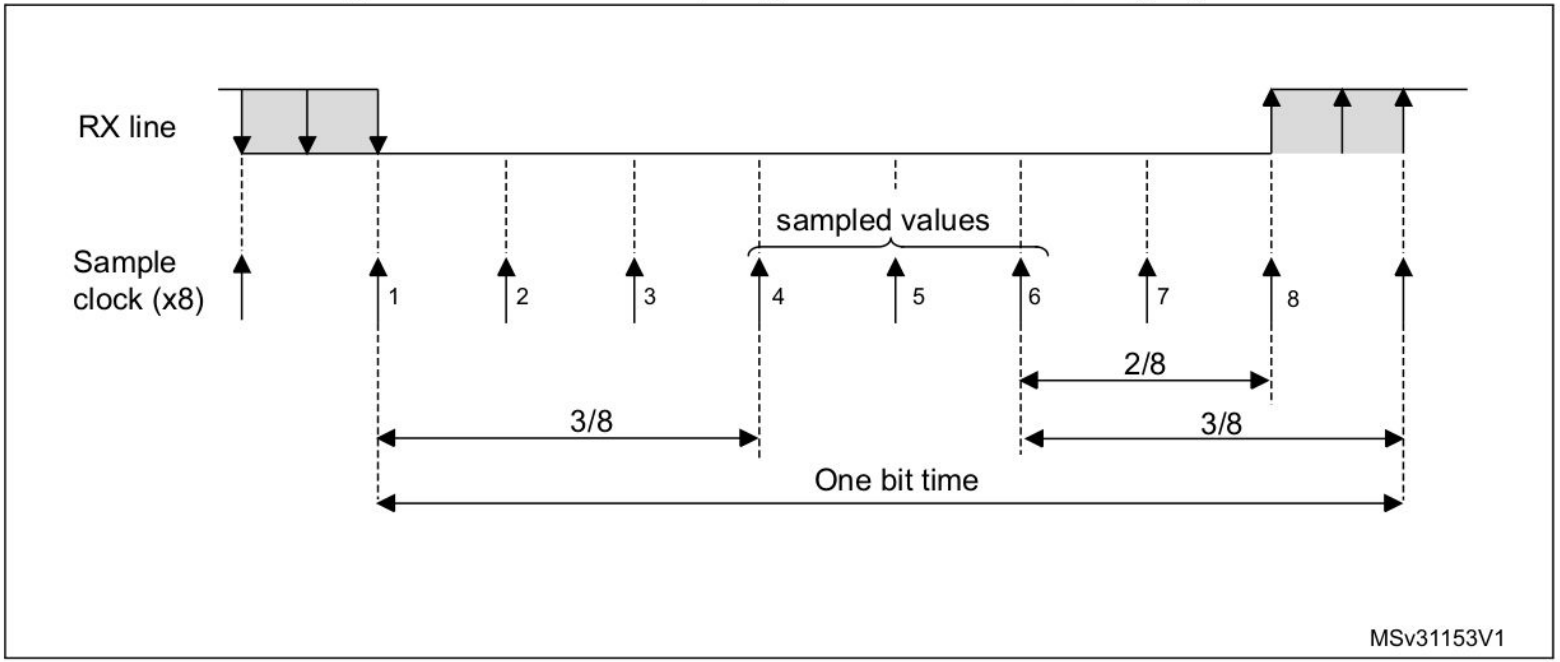
USART. IrDA


USART
SIR
Transmit
Encoder
IrDA_OUT
SIR
Receive
Decoder
IrDA_IN
Idle
USART
(NRZ)
S
0
1
2
3
4
5
6
7
P
Stop
Idle
IrDA
(RZI)

USART. Oversampling by 16



USART. Oversampling by 8



UART. Noise detection

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

USART. Baud rate generation

In case of oversampling by 16, the equation is:

$$\text{Tx/Rx baud} = \frac{f_{CK}}{\text{USARTDIV}}$$

In case of oversampling by 8, the equation is:

$$\text{Tx/Rx baud} = \frac{2 \times f_{CK}}{\text{USARTDIV}}$$

USART. Baud rate generation

To obtain 9600 baud with f CK = 8 MHz.

In case of oversampling by 16:

USARTDIV = 8,000,000/9,600

BRR = USARTDIV = 833d = **0x341**

In case of oversampling by 8:

USARTDIV = 2 * 8,000,000/9,600

USARTDIV = 1,666.66 (1,667d = 683h)

BRR[3:0] = 3h >> 1 = 1h

BRR = 0x681

USART. USART baud rate generation

Baud rate		Oversampling by 16 (OVER8 = 0)			Oversampling by 8 (OVER8 = 1)		
S.No	Desired	Actual	BRR	% Error = (Calculated - Desired)B.Rate / Desired B.Rate	Actual	BRR	% Error
2	2.4 KBps	2.4 KBps	0x4E20	0	2.4 KBps	0x9C40	0
3	9.6 KBps	9.6 KBps	0x1388	0	9.6 KBps	0x2710	0
4	19.2 KBps	19.2 KBps	0x9C4	0	19.2 KBps	0x1384	0
5	38.4 KBps	38.4 KBps	0x4E2	0	38.4 KBps	0x9C2	0
6	57.6 KBps	57.62 KBps	0x341	0.03	57.59 KBps	0x681	0.02
7	115.2 KBps	115.11 KBps	0x1A1	0.08	115.25 KBps	0x340	0.04
8	230.4 KBps	230.76KBps	0xD0	0.16	230.21 KBps	0x1A0	0.08
9	460.8 KBps	461.54KBps	0x68	0.16	461.54KBps	0xD0	0.16
10	921.6KBps	923.07KBps	0x34	0.16	923.07KBps	0x64	0.16
11	2 MBps	2 MBps	0x18	0	2 MBps	0x30	0
12	3 MBps	3 MBps	0x10	0	3 MBps	0x20	0
13	4MBps	N.A	N.A	N.A	4MBps	0x14	0
14	5MBps	N.A	N.A	N.A	5052.63KBps	0x11	1.05
15	6MBps	N.A	N.A	N.A	6MBps	0x10	0

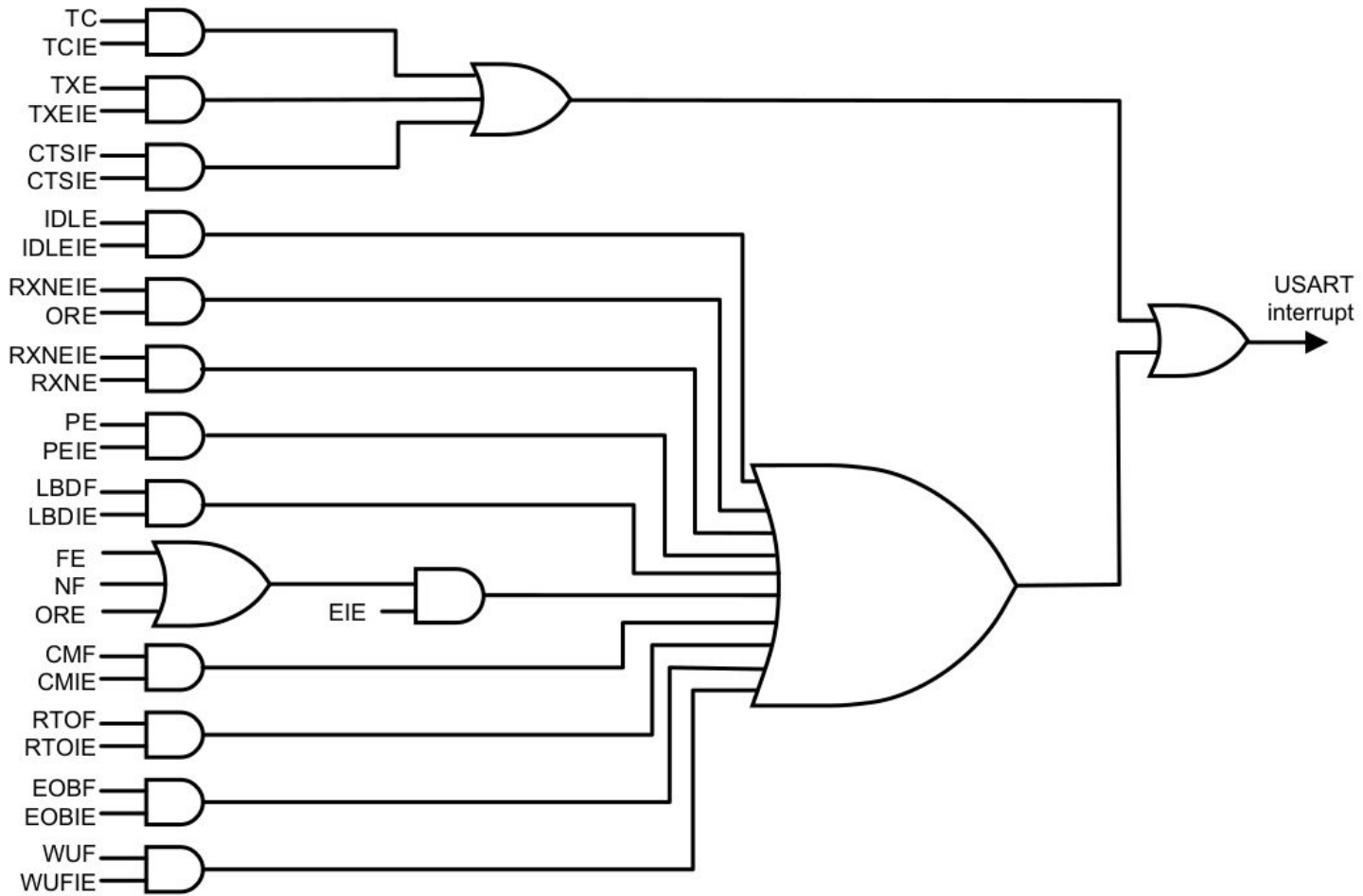
USART. Parity control

M bits	PCE bit	USART frame(1)
00	0	| SB | 8-bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data | PB | STB |
10	0	| SB | 7-bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (9th, 8th or 7th, depending on the M bits value).

USART. Interrupts

Interrupt event	Event flag	Enable Control bit
Transmission Complete	TC	TCIE
Receive data register not empty (data ready to be read)	RXNE	RXNEIE
Overrun error detected	ORE	
Idle line detected	IDLE	IDLEIE
Parity error	PE	PEIE
LIN break	LBDF	LBDIE
Noise Flag, Overrun error and Framing Error in multibuffer communication.	NF or ORE or FE	EIE
Character match	CMF	CMIE
Receiver timeout	RTOF	RTOIE
End of Block	EOBF	EOBIE
Wakeup from Stop mode	WUF(1)	WUFIE


TC
TCIE
TXE
TXEIE
CTSIF
CTSIE
IDLE
IDLEIE
RXNEIE
ORE
RXNEIE
RXNE
PE
PEIE
LBDF
LBDIE
FE
NF
ORE
EIE
CMF
CMIE
RTOF
RTOIE
EOBF
EOBIE
WUF
WUFIE
USART
interrupt

USART. Registers

- **USART_CR1**
- **USART_CR2**
- **USART_CR3**
- **USART_BRR - Baud rate register**
- **USART_GTPR - Guard time and prescaler register**
- **USART_RTOR - Receiver timeout register**
- **USART_RQR - Request register**
- **USART_ISR - Interrupt and status register**
- **USART_ICR - Interrupt flag clear register**
- ***USART_RDR* - Receive data register**
- ***USART_TDR* - Transmit data register**

Killer-feature of USART

Debug might become easier and PC-like

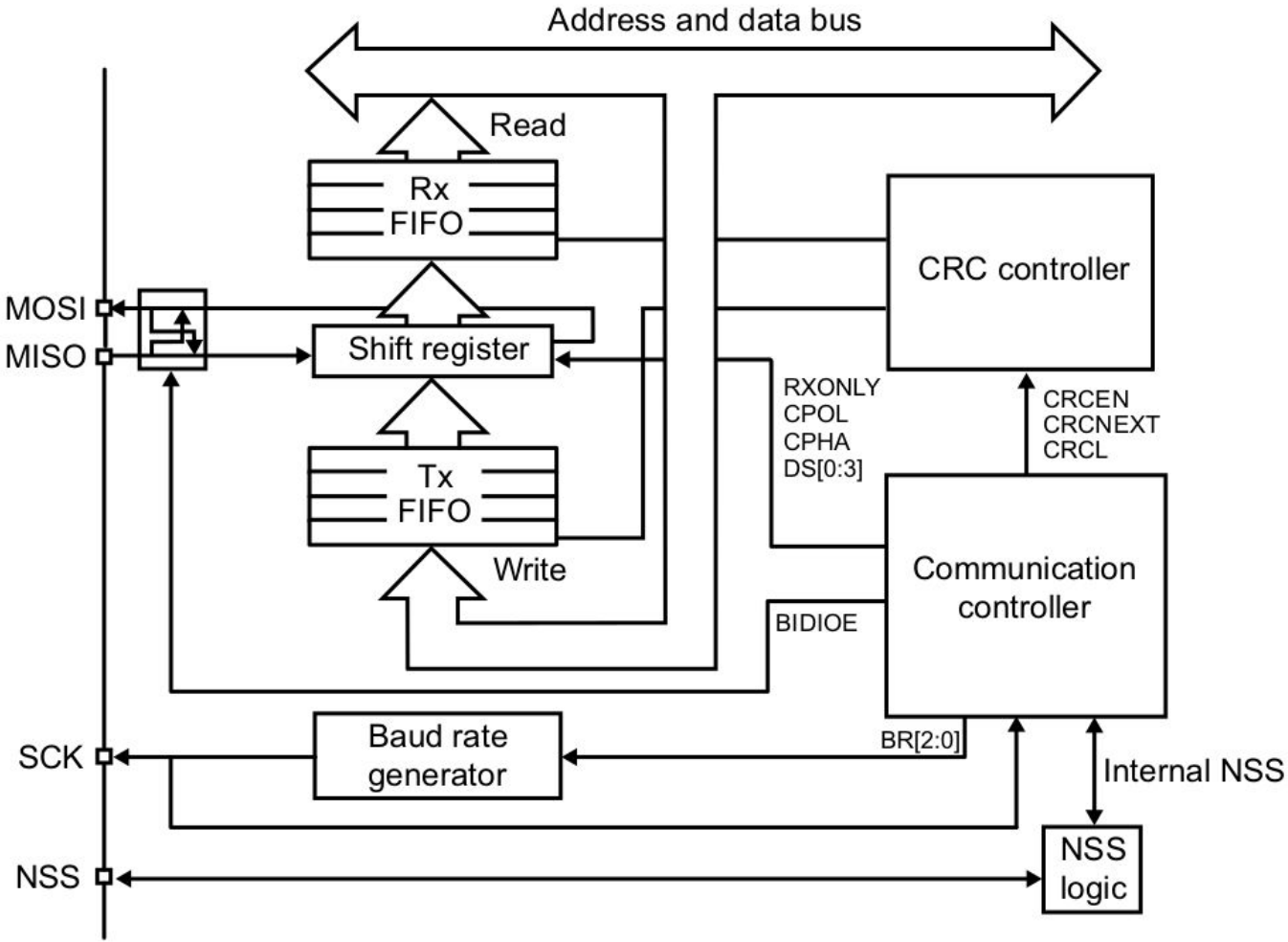
But we give no warranty

USART. Example

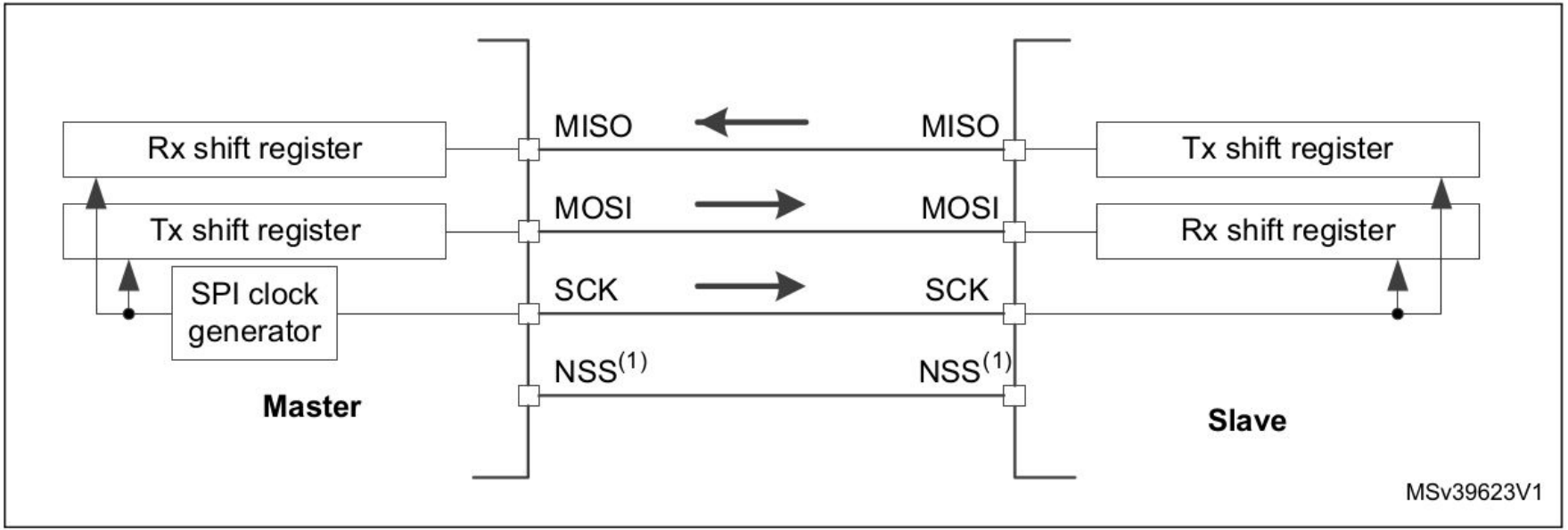
Switch to example!

SPI. Serial peripheral interface

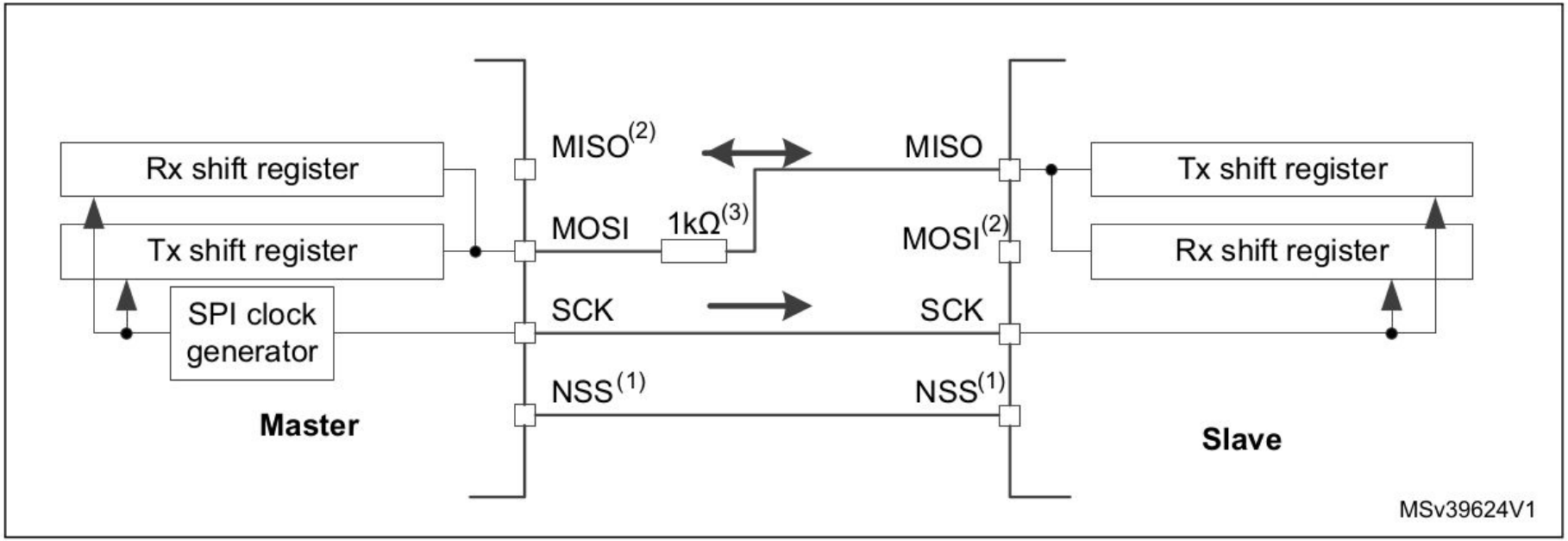
- Master or slave operation
- Full-duplex, half-duplex and simplex
- 4-bit to 16-bit data size selection
- Programmable clock polarity and phase
- NSS management by hardware or software


Address and data bus
Read
Rx
FIFO
CRC controller
MOSI
MISO
Shift register
RXONLY
CPOL
CPHA
DS[0:3]
CRCEN
CRCNEXT
CRCL
Tx
FIFO
Write
Communication
controller
BIDIOE
Baud rate
generator
BR[2:0]
SCK
Internal NSS
NSS
NSS
logic

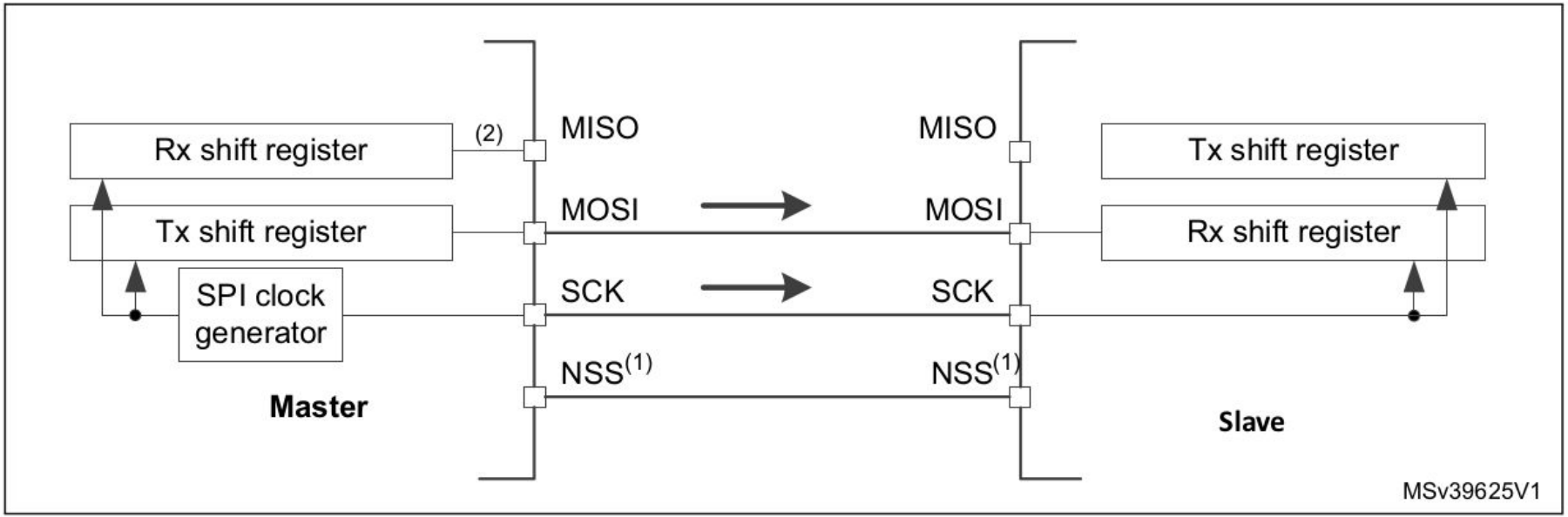
SPI. Full-duplex



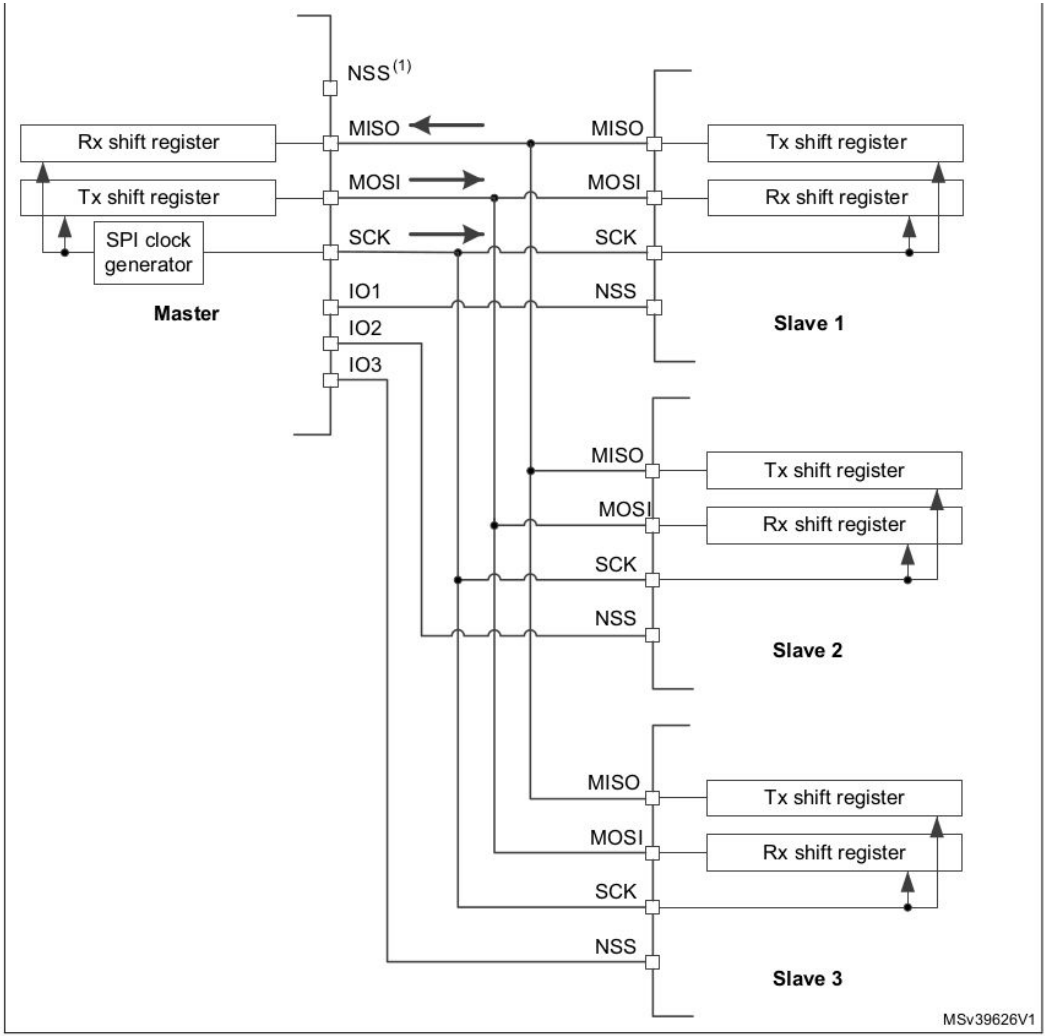
SPI. Half-duplex



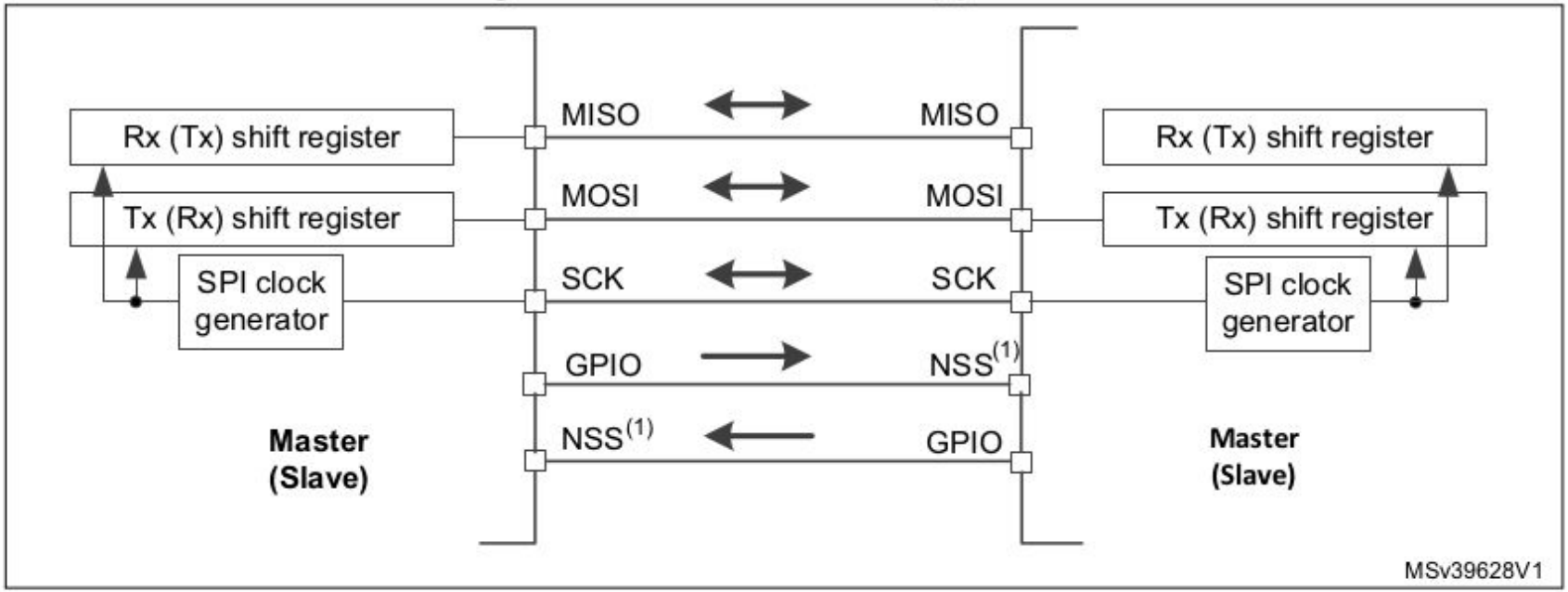
SPI. Simplex


Rx shift register
(2)
MISO
MISO
Tx shift register
Tx shift register
MOSI
MOSI
Rx shift register
SPI clock generator
SCK
SCK
NSS(1)
NSS(1)
Master
Slave
MSv39625V1

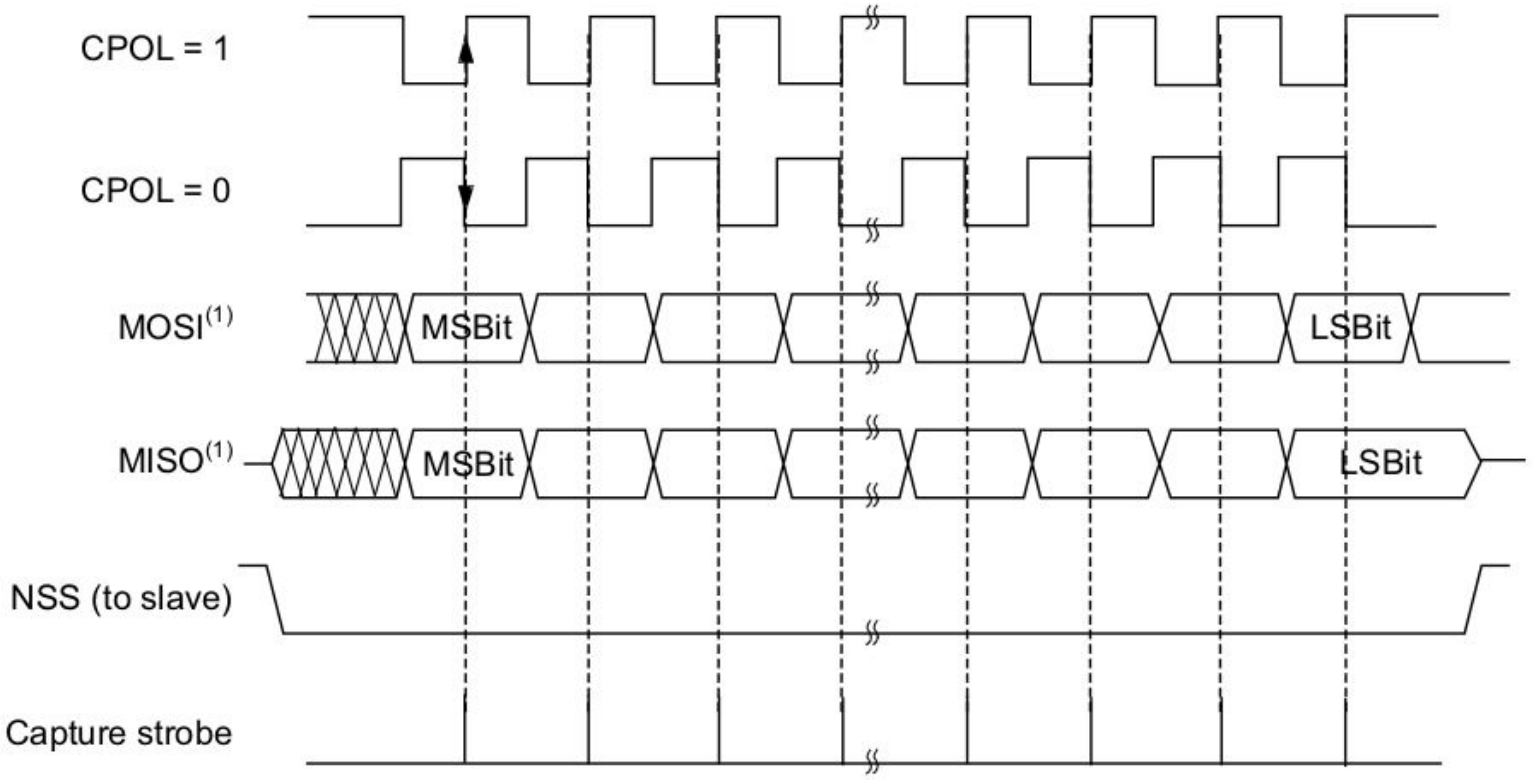
SPI. Multi slave mode



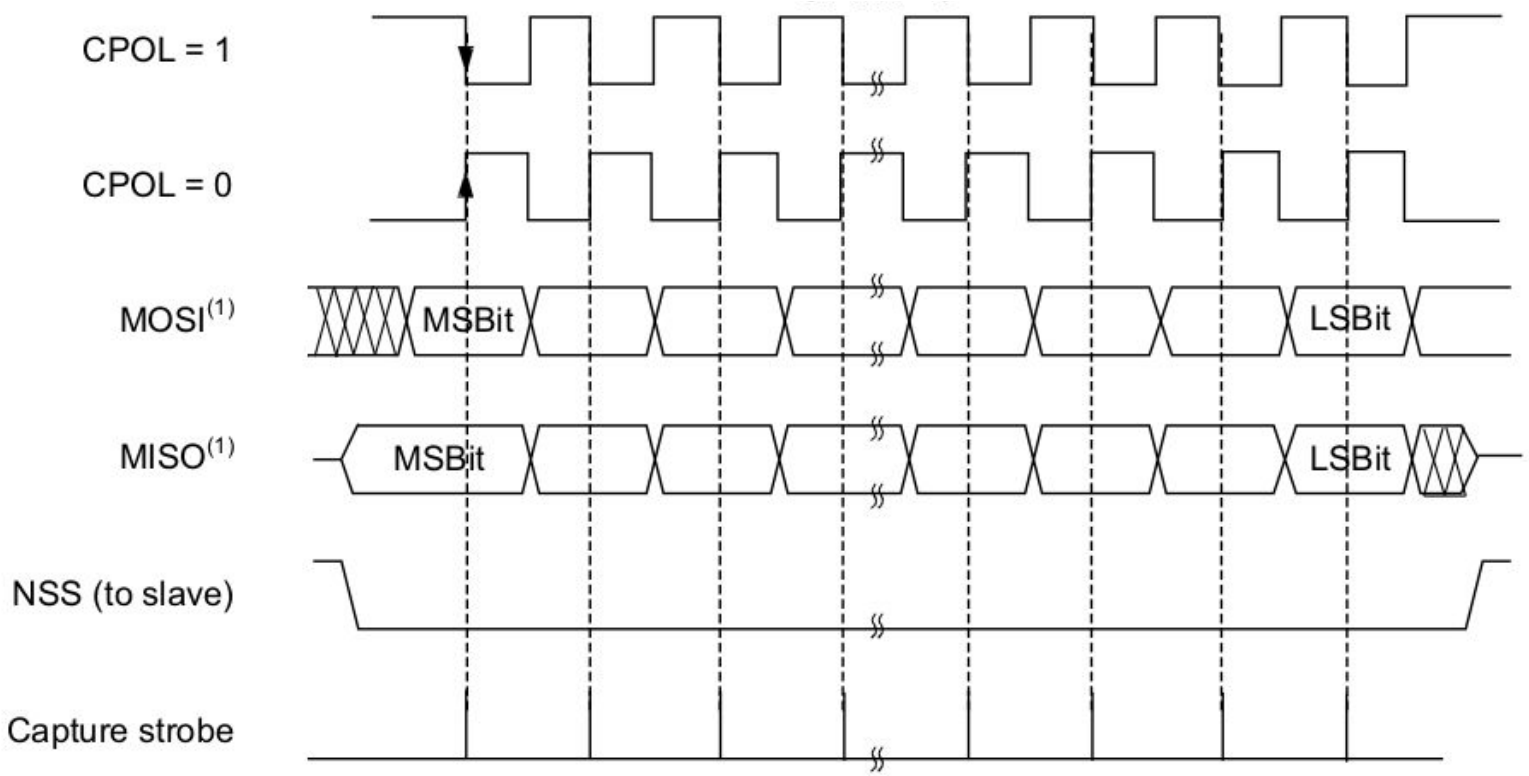
SPI. Multi master mode



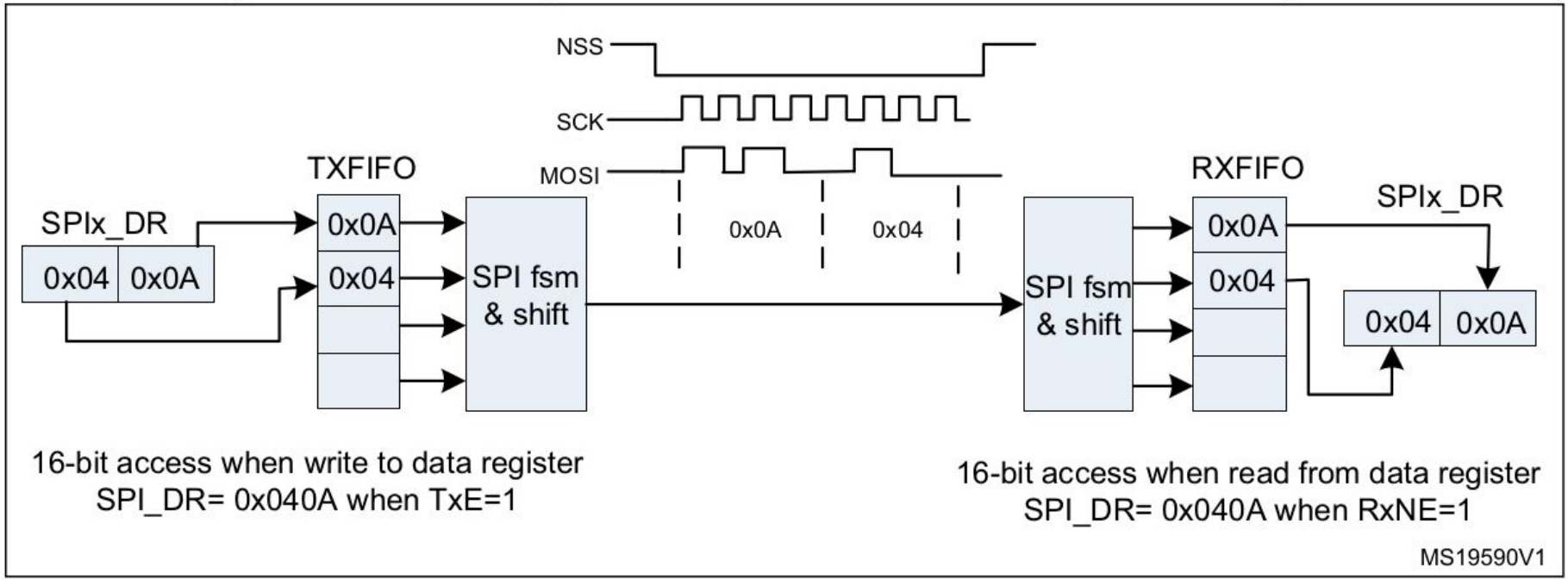
SPI. CPHA (clock phase) = 1



SPI. CPHA (clock phase) = 0



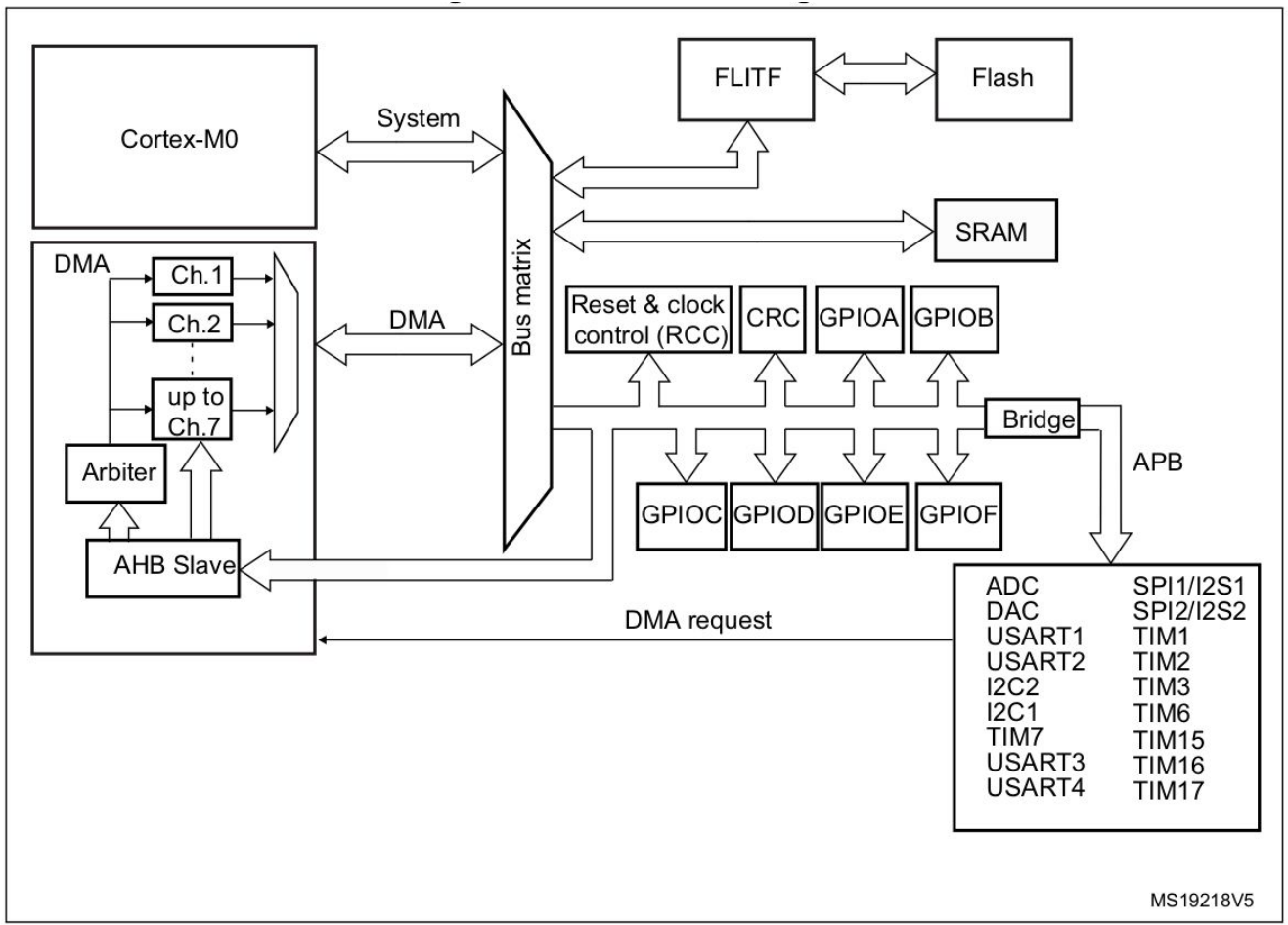
SPI. RX and TX FIFOs



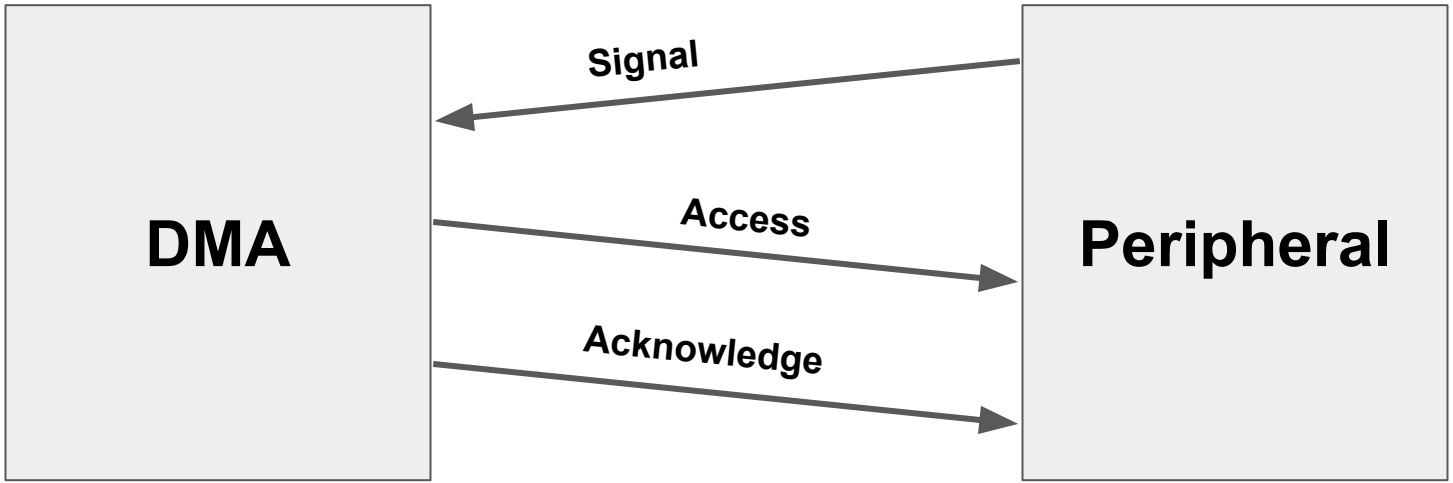
DMA. Main features

- Up to 7 independently configurable channels
- Priorities between requests from the DMA channels are software programmable
- Independent source and destination transfer size (byte, halfword, word)
- 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, APB and AHB peripherals as source and destination
- Programmable number of data to be transferred: up to 65535

DMA. Diagram


Cortex-M0
System
FLITF
Flash
SRAM
DMA
Ch.1
Ch.2
up to Ch.7
Arbiter
AHB Slave
DMA
Bus matrix
Reset & clock control (RCC)
CRC
GPIOA
GPIOB
Bridge
APB
GPIOC
GPIOD
GPIOE
GPIOF
DMA request
ADC
DAC
USART1
USART2
I2C2
I2C1
TIM7
USART3
USART4
SPI1/I2S1
SPI2/I2S2
TIM1
TIM2
TIM3
TIM6
TIM15
TIM16
TIM17
MS19218V5

DMA transactions



DMA_CPARx, DMA_CMARx and DMA_CNDTRx

Arbiter

- The arbiter manages the channel requests based on their priority and launches the peripheral/memory access sequences
- Software: each channel priority can be configured in the DMA_CCRx register. There are four levels:
  - Very high priority
  - High priority
  - Medium priority
  - Low priority
- Hardware: if 2 requests have the same software priority level, the channel with the lowest number will get priority versus the channel with the highest number.

DMA. Channel configuration procedure

- Set the peripheral register address in the DMA_CPARx register
- Set the memory address in the DMA_CMARx register
- Configure the total number of data to be transferred in the DMA_CNDTRx register
- Configure the channel priority using the PL[1:0] bits in the DMA_CCRx register
- Configure data transfer direction, circular mode, peripheral & memory incremented mode, peripheral & memory data size, and interrupt after half and/or full transfer in the DMA_CCRx register
- Activate the channel by setting the ENABLE bit in the DMA_CCRx register

DMA. Reminder of sizes

- When the DMA initiates an AHB byte or halfword write operation, the data are duplicated on the unused lanes of the HWDATA[31:0] bus
- To write the halfword “0xABCD”, the DMA sets the HWDATA bus to “0xABCDABCD” with HSIZE = HalfWord
- To write the byte “0xAB”, the DMA sets the HWDATA bus to “0xABABABAB” with HSIZE = Byte
- An AHB byte write operation of the data “0xB0” to 0x0 (or to 0x1, 0x2 or 0x3) will be converted to an APB word write operation of the data “0xB0B0B0B0” to 0x0
- An AHB halfword write operation of the data “0xB1B0” to 0x0 (or to 0x2) will be converted to an APB word write operation of the data “0xB1B0B1B0” to 0x0

DMA. Error management

- Error can be generated by reading from or writing to a reserved address space
- Only it happens, the faulty channel is automatically disabled through a hardware clear of its EN bit in the corresponding Channel configuration register (DMA_CCRx)
- The channel transfer error interrupt flag (TEIF) in the DMA_IFR register is set and an interrupt is generated if the transfer error interrupt enable bit (TEIE) in the DMA_CCRx register is set

DMA. Interrupts

Interrupt event	Event flag	Enable control bit
Half-transfer	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE

DMA. Be aware of limitations

The hardware requests from the peripherals (TIMx, ADC, DAC, SPI, I2C, and USARTx) are simply logically ORed before entering the DMA. This means that on one channel, only one request must be enabled at a time.

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
ADC	ADC(1)	ADC(2)	-	-	-
SPI	-	SPI1_RX	SPI1_TX	SPI2_RX	SPI2_TX
USART	-	USART1_TX(1)	USART1_RX(1)	USART1_TX(2) USART2_TX	USART1_RX(2) USART2_RX
I2C	-	I2C1_TX	I2C1_RX	I2C2_TX	I2C2_RX
TIM1	-	TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_CH3 TIM1_UP
TIM2	TIM2_CH3	TIM2_UP	TIM2_CH2	TIM2_CH4	TIM2_CH1
TIM3	-	TIM3_CH3	TIM3_CH4 TIM3_UP	TIM3_CH1 TIM3_TRIG	-
TIM6 / DAC	-	-	TIM6_UP DAC_Channel1	-	-
TIM15	-	-	-	-	TIM15_CH1 TIM15_UP TIM15_TRIG TIM15_COM
TIM16	-	-	TIM16_CH1(1) TIM16_UP(1)	TIM16_CH1(2) TIM16_UP(2)	-
TIM17	TIM17_CH1(1) TIM17_UP(1)	TIM17_CH1(2) TIM17_UP(2)	-	-	-

DMA. Example

Switch to example!