COEN-4720 Embedded Systems

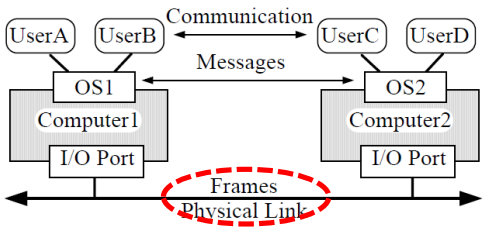
Lecture 6
UART, SPI

Cris Ababei
Dept. of Electrical and Computer Engineering
MARQUETTE UNIVERSITY
BE THE DIFFERENCE.

1

Communication Systems – the Layered View

1) Address information field
   - physical address specifying the destination/source computers
   - logical address specifying the destination/source processes (e.g., users)
2) Synchronization or handshake field
   - Physical synchronization like shared clock, start and stop bits
   - OS synchronization like request connection or acknowledge
   - Process synchronization like semaphores
3) Data field
   - ASCII text (raw or compressed)
   - Binary (raw or compressed)
4) Error detection and correction field
   - Vertical and horizontal parity
   - Checksum
   - Logical redundancy check (LRC)
   - Block correction codes (BCC)



2

Outline

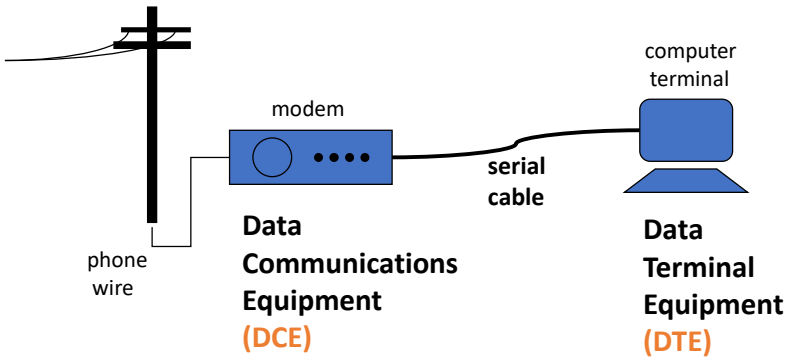
- UART (Serial)
- SPI

3

3

DTE and DCE

- DTE: Data Terminal Equipment
- DCE: Data Communications Equipment
- PC with a modem:

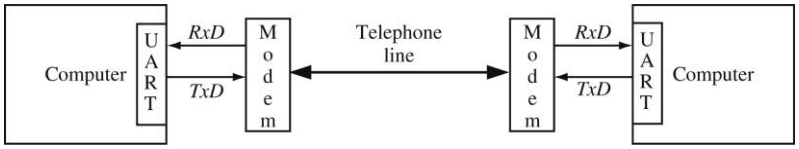


4

4

UART

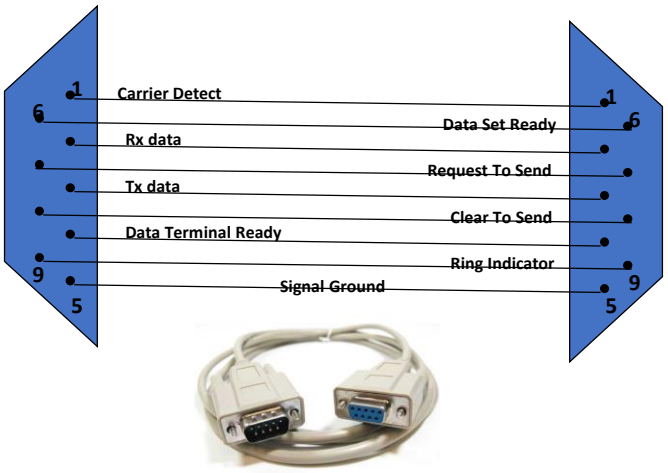
- Original purpose of the UART was for PCs to communicate via the telephone network
- Telephones were for voice communication (analog signals) whereas computers need so exchange discrete data (digital signals)
- Special 'communication equipment' was needed for doing the signal conversions (i.e., a modulator/demodulator, or modem)



5

5

Normal 9-wire Serial Cable



6

6

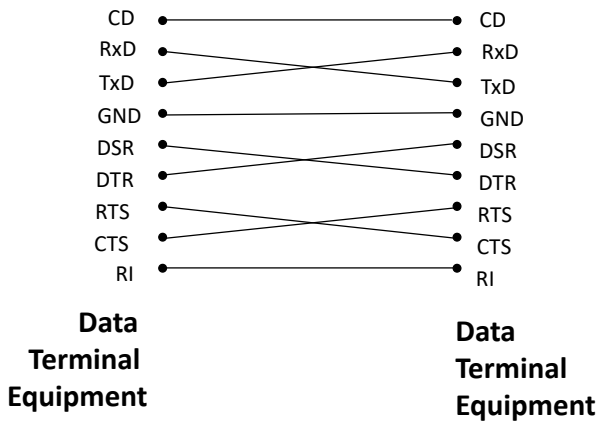
Signal Functions

- CD: **Carrier Detect** The modem asserts this signal to indicate that it successfully made its connection to a remote device
- RI: **Ring Indicator** The modem asserts this signal to indicate that the phone is ringing at the other end of its connection
- DSR: **Data Set Ready** Modem to PC
- DTR: **Data Terminal Ready** PC to Modem
- RTS: **Request To Send** PC is ready for the modem to relay some received data
- CLS: **Clear To Send** Modem is ready for the PC to begin transmitting some data

7

7

9-wire Null-Modem Cable



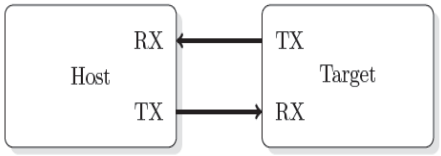
no modems

8

8

Basics

- The most basic method for communication with an embedded processor is **asynchronous serial**.
- It is implemented over a symmetric pair of wires connecting two devices (referred as host and target, though these terms are arbitrary).
- Whenever the host has data to send to the target, it does so by sending an encoded bit stream over its transmit (TX) wire. This data is received by the target over its receive (RX) wire.
- The communication is similar in the opposite direction.



9

9

Basics

- This mode of communications is called **asynchronous** because the host and target share no time reference (no clock signal).
- Instead, temporal properties are encoded in the bit stream by the transmitter and must be decoded by the receiver.
- A commonly used **device** for encoding and decoding such asynchronous bit streams is a **Universal Asynchronous Receiver/Transmitter (UART).**

10

10

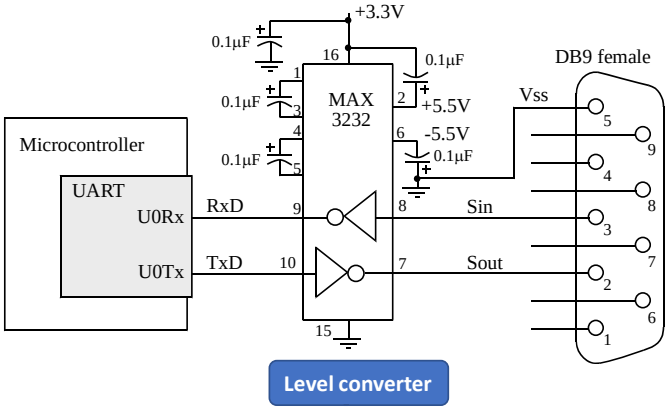
UART

- UART is a circuit that sends parallel data through a serial line.
- UARTs are frequently used in conjunction with the RS-232 standard (or specification), which specifies the electrical, mechanical, functional, and procedural characteristics of two data communication equipment.
- Other used standards: EIA, RS-422, RS-485
- A UART includes a transmitter and a receiver.
  - The transmitter is a special shift register that loads data in parallel and then shifts it out bit by bit at a specific rate.
  - The receiver shifts in data bit by bit and reassembles the data.

11

11

UART



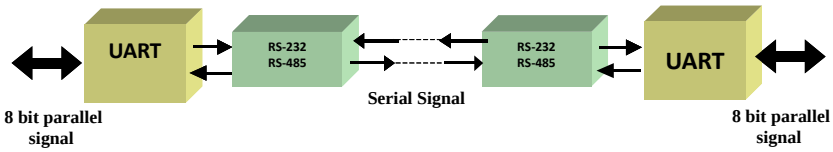
DB25 Pin	RS232 Name	DB9 Pin	EIA-574 Name	Signal	Description	True	DTE	DCE
2	BA	3	103	TxD	Transmit Data	-5.5V	out	in
3	BB	2	104	RxD	Receive Data	-5.5V	in	out
7	AB	5	102	SG	Signal Ground			

12

12

UART System Block Diagram

- UARTs are useful in applications where a lower cost connection is desired.
- A UART takes a parallel data stream and funnels it down to a serial data stream at the transceiver end and then returns the data stream to a parallel signal at the receiver end.
- This lowers the cost of connection by
  - Decreasing the number of transceivers that are necessary.
  - Enabling the connecting cable to be less costly and less bulky.



13

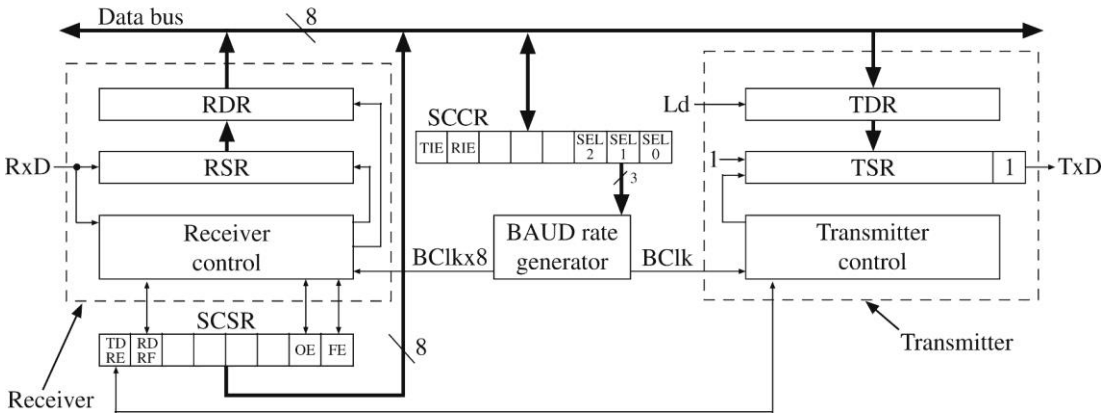
Notations

- UART Registers
  - RSR: Receive Shift Register
  - RDR: Receive Data Register
  - TDR: Transmit Data Register
  - TSR: Transmit Shift Register
- SCCR: Serial Communications Control Register
- SCSR: Serial Communications Status Register
- UART Flags
  - TDRE: Transmit Data Register Empty
  - RDRF: Receive Data Register Full

14

14

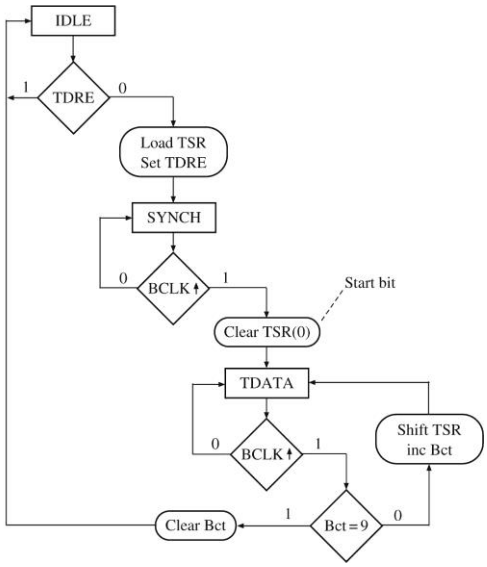
UART Block Diagram



Transmitter Operation

- Microcontroller waits until TDRE = '1'
  - Loads data into TDR
  - Clears TDRE
- UART transfers data from TDR to TSR
  - Sets TDRE
- UART outputs start bit ('0') then shifts TSR right eight times followed by a stop bit ('1')

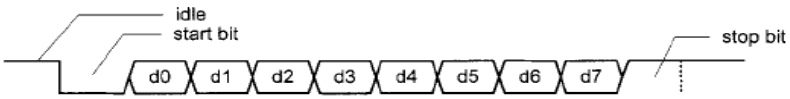
Transmitter SM Chart



17

UART Transmission Details

- The serial line is '1' when it is idle.
- The transmission starts with a start-bit, which is '0', followed by data-bits and an optional parity-bit, and ends with stop-bits, which are '1'.
- The number of data-bits can be 6, 7, or 8.
- The optional parity bit is used for error detection.
  - For odd parity, it is set to '0' when the data bits have an odd number of '1's.
  - For even parity, it is set to '0' when the data-bits have an even number of '1's.
- The number of stop-bits can be 1, 1.5, or 2.



18

18

UART "Agreement"

- No clock information is conveyed through the serial line.
- Before the transmission starts, the transmitter and receiver must agree on a set of parameters in advance:
  - the baud-rate (i.e., number of bits per second),
  - the number of data bits and stop bits
  - use of parity bit

19

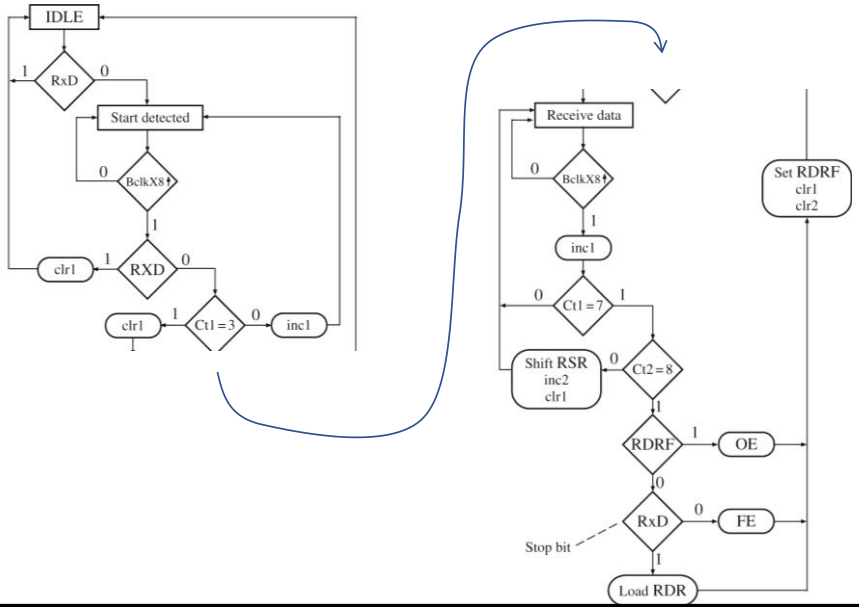
19

Receiver Operation

- UART waits for start bit
  - Shifts bits into RSR
- When all data bits and stop bit are received
  - RSR loaded into RDR
  - Set RDRF
- Microcontroller waits until RDRF is set
  - Read RDR
  - Clear RDRF

20

Receiver SM Chart

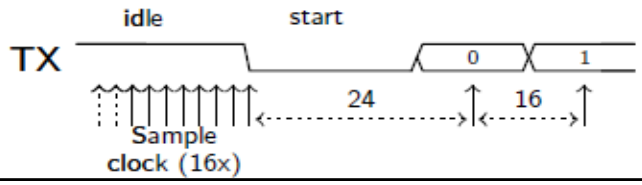


21

21

Data Extraction Details

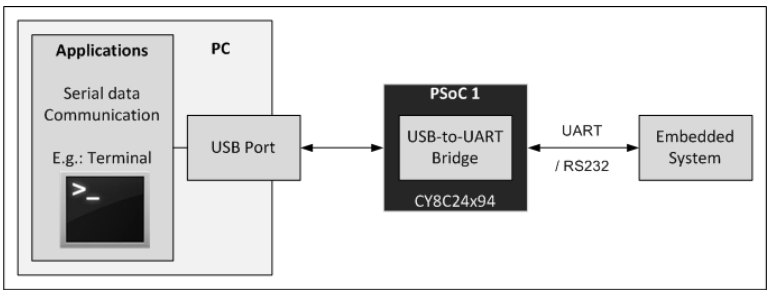
- Assume the UART's receiver has a clock running at a multiple of the baud rate (e.g., 16x).
- Starting in the idle state, the receiver “samples” its RX signal until it detects a high-low transition.
- Then, it waits 1.5 bit periods (24 clock periods) to sample its RX signal at what it estimates to be the center of data bit 0.
- The receiver then samples RX at bit-period intervals (16 clock periods) until it has read the remaining 7 data bits and the stop bit.
- From that point this process is repeated.
- Successful extraction of the data from a frame requires that, over 10.5 bit periods, the drift of the receiver clock relative to the transmitter clock be less than 0.5 periods in order to correctly detect the stop bit.



22

UART Use Example

- UARTs can be used to interface to a wide variety of other peripherals
  - Widely available GSM/GPRS cell phone modems
  - Bluetooth modems can be interfaced to a microcontroller UART
  - GPS receivers frequently support UART interfaces



23

23

Outline

- UART
- SPI

24

24

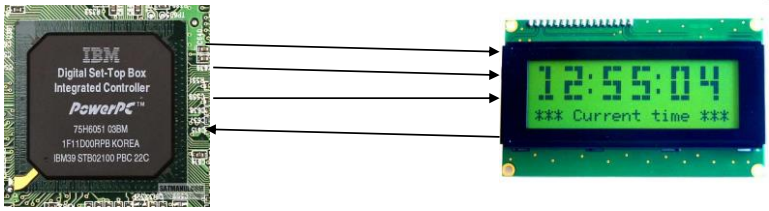
SPI Basics

- Serial Peripheral Interface (SPI) is a simple serial communication method/protocol using 4 wires
  - Also known as a 4-wire bus
- Used to communicate across small distances
- Multiple Slaves, Single Master
- Synchronized

25

SPI

- Used to connect devices such as printers, cameras, scanners, etc. to a desktop computer; but it has largely been replaced by USB
- SPI can still be a useful communication tool for some applications



26

SPI

- Fast, Easy to use, Simple
- Everyone supports it
- Has some advantages over I2C
  - SPI can communicate at much higher data rates than I2C
  - Also, when multiple slaves are present, SPI requires no addressing to differentiate between these slaves

27

27

Capabilities of SPI

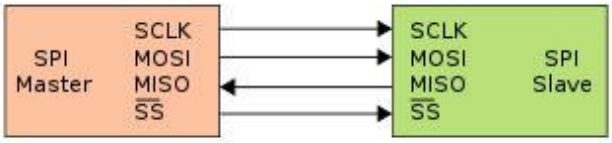
- Always Full Duplex
  - Communicating in two directions at the same time
- Multiple Mbps transmission speed
- Transfers data in 4..16 bit characters
- Multiple slaves
  - Daisy-chaining possible (a wiring scheme in which multiple devices are wired together in sequence or in a ring)

28

28

Communication Method

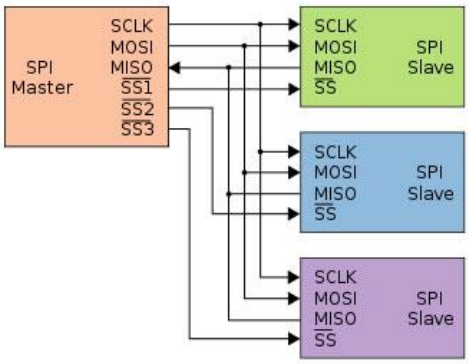
- SPI runs using a master/slave set-up and can run in full duplex mode (i.e., signals can be transmitted between the master and the slave simultaneously).



29

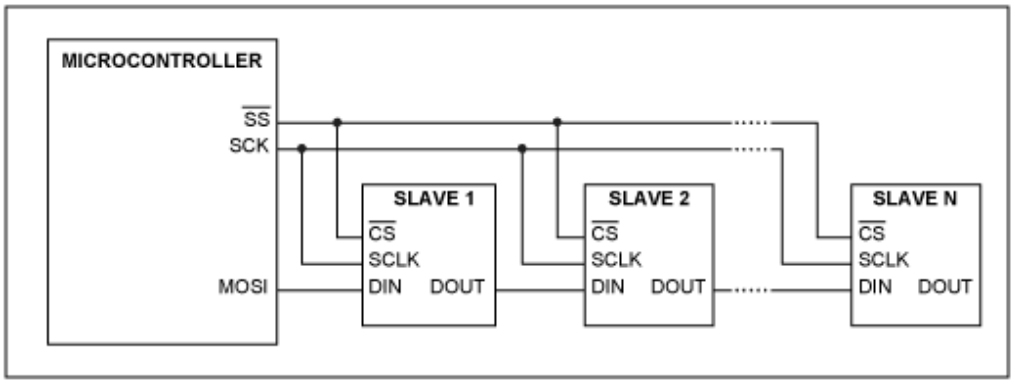
29

Master and Multiple Independent Slaves



30

Master and Multiple Daisy-chained Slaves

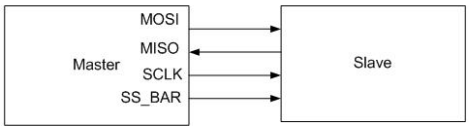


31

31

Protocol

- Wires:
  - Master Out Slave In (MOSI)
  - Master In Slave Out (MISO)
  - System Clock (SCLK)
  - Slave Select 1…N
- Master Set Slave Select low
- Master Generates Clock
- Shift registers shift in and out data

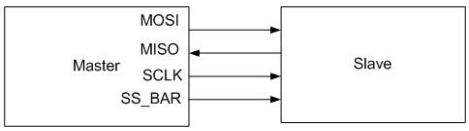


32

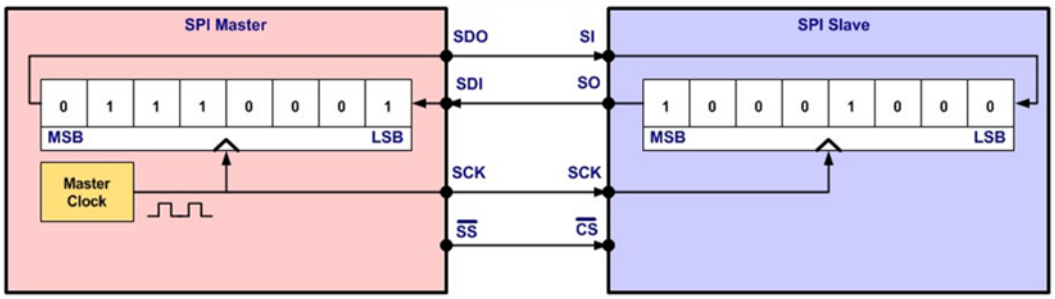
32

Wires in Detail

- MOSI – Carries data out of Master to Slave
- MISO – Carries data from Slave to Master
  - Both signals happen for every transmission
- SS_BAR – Unique line to **select a slave**
- SCLK – Master produced clock to synchronize data transfer



Shifting Detail



Clock Phase (advanced)

- Two phases and two polarities of clock
- Four modes
- Master and selected slave must be in same mode
- Master must change polarity and phase to communicate with slaves of different numbers

35

35

Pros and Cons

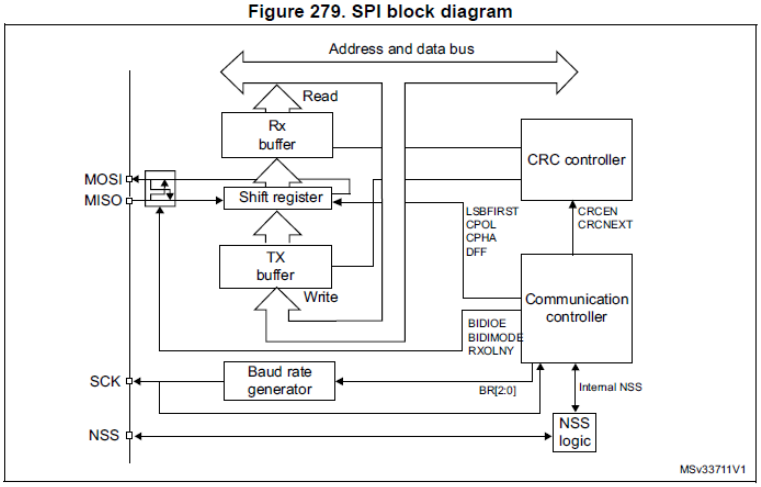
- Pros:
  - Fast and easy
    - Fast for point-to-point connections
    - Easily allows streaming/Constant data inflow
    - No addressing/Simple to implement
  - Everyone supports it
- Cons:
  - SS makes multiple slaves very complicated
  - No acknowledgement ability
  - No inherent arbitration
  - No flow control

36

36

SPI Block Diagram – STM32L0x3 MCU

Figure 279. SPI block diagram



Source: Datasheet

SPI Interrupts

31.5 SPI interrupts

During SPI communication an interrupts can be generated by the following events:

- Transmit Tx buffer ready to be loaded
- Data received in Rx buffer
- Master mode fault
- Overrun error
- TI frame format error

Interrupts can be enabled and disabled separately.

Table 157. SPI interrupt requests

Interrupt event	Event flag	Enable Control bit
Transmit Tx buffer ready to be loaded	TXE	TXEIE
Data received in Rx buffer	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
CRC error	CRCERR	
TI frame format error	FRE	

For code example, refer to *A.19.6: SPI interrupt code example.*

Source: Datasheet

Summary

Comm. method	Shares clock	Num. of wires	Speed	Dist	Pros	Cons
UART	No	2	115Kbits/sec max	Medium, long	Simple; Widely supported; Large range of physical standard interfaces (TTL, RS-232, RS-422, RS-485);	It's asynchronous; Requires reasonable clock accuracy at both ends;
CAN	No	3	1 Mbits/sec	Long: 40m (1Mbit/sec) up to 10km (5Kbits/sec)	Highly reliable; Reduces amount of wiring; Multi-master capability;	Complex;
I2C	Yes	2	100Kbits/sec 400Kbits/sec fast mode	Short, medium (< 6")	Simple; Multi-master capability; Only 2 wires to support multiple devices; Robust in noisy or power-up/down situations;	More complex protocol than SPI; Harder to level-shift or opto-isolate due to bidirectional lines; Need for pull-up resistors can reduce power efficiency in some cases;
SPI	Yes	4	10-20Mbits/sec	Short	Fast, easy, simple; A lot of support; Self clocking; Flexible data word sizes;	Multiple devices need multiple select lines; No acknowledgement ability; No inherent arbitration; No flow control; Single master only;

39

39

Credits and References

- [1] Carmine Noviello, Mastering STM32, Second Edition, 2022. (Required, Book 1). Available to purchase online.
- [2] Joseph Yiu, The Definitive Guide to ARM Cortex-M0 and Cortex-M0+ Processors, 2nd Ed., 2015. (Book 2). Can be found online.
- http://www.ee.nmt.edu/~teare/ee308l/datasheets/S12SPIV3.pdf
- https://www.eecs.umich.edu/courses/eecs373/refs.html
- Jonathan W. Valvano, Embedded Systems: Introduction to Arm Cortex-M3 Microcontrollers, 2012. (Chapter 8)

40

40