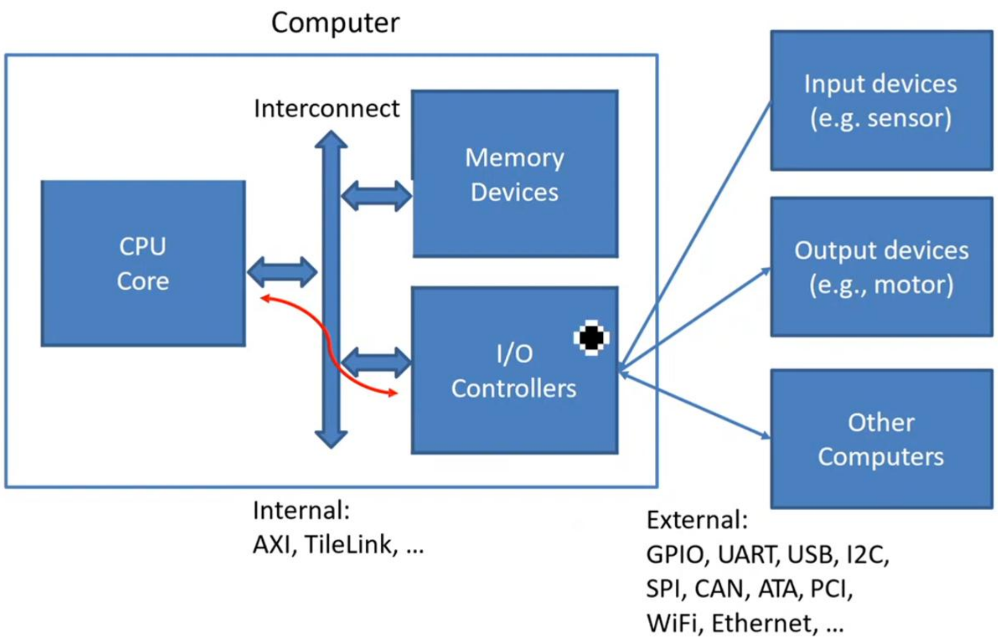
Introduction to Embedded Systems

Unit 1.5: External Communication Buses

Alexander Yemane
BCIT
INCS 3610
Fall 2025

How do we send data between chips and devices?

- Very often, a processor needs to exchange information with other processors or peripherals
- To satisfy various needs, there exists many different *communication protocols*
- *I/O controllers* act as communication intermediaries between the processor and peripherals

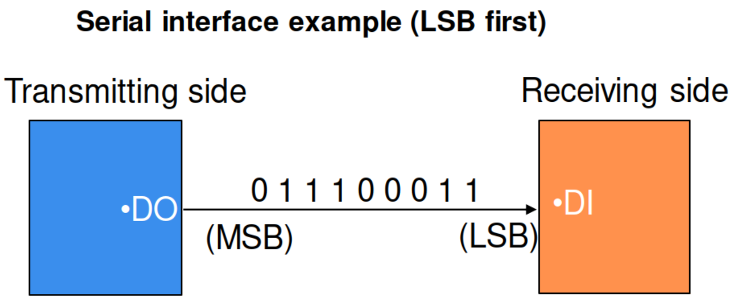


Communication bus design space

- Number of wires required?
- Asynchronous or synchronous?
- How fast can it transfer data?
- Can it support more than two endpoints?
- Can it support more than one master?
- How do we support flow control?
- How does it handle errors/noise?

Series and parallel communication

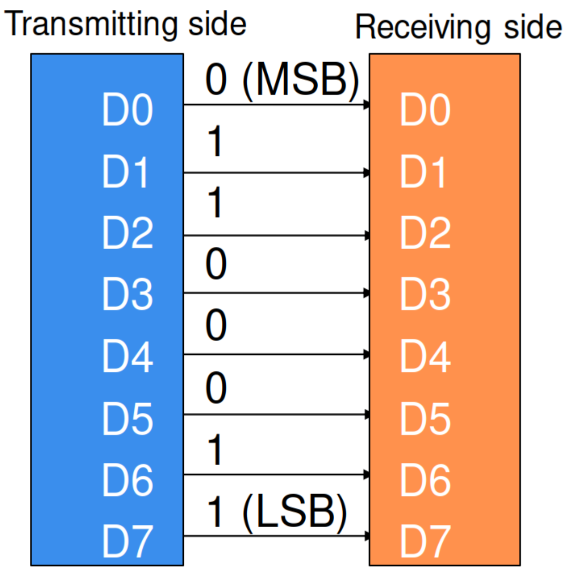
- Serial
  - Each bit of the message is sent in sequence, one at a time, through a single wire.
  - Slower but more cost-effective data transfer process
  - Commonly employed for longer distances



Series and parallel communication

- Parallel
  - Each bit in the data has its own dedicated path, allowing the entire message to be transmitted simultaneously.
  - Faster data transfer rate.
  - Suitable for short-distance applications where speed is critical

Parallel interface example



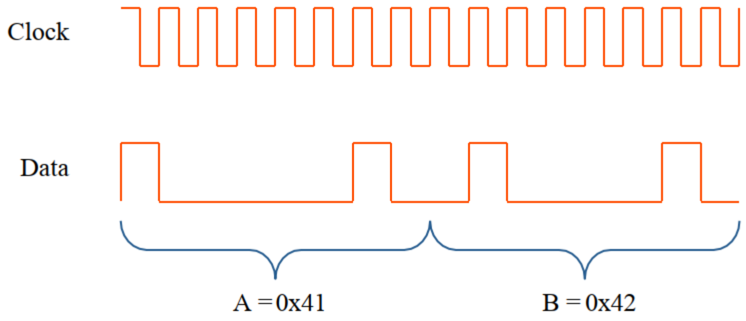
Series and parallel communication: use cases

- Why serial communication?
  - Is a pin-efficient way of sending and receiving bits of data
  - Parallel communication has issues with clock skew, crosstalk, interconnect density

- Serial interfaces are increasingly popular for external I/O
  - E.g. USB, SATA, SD cards
- Parallel interfaces are dominant* for internal interconnect within chips where clock synchronization is easier
  - E.g. AXI, TileLink, PCI (not PCIe)

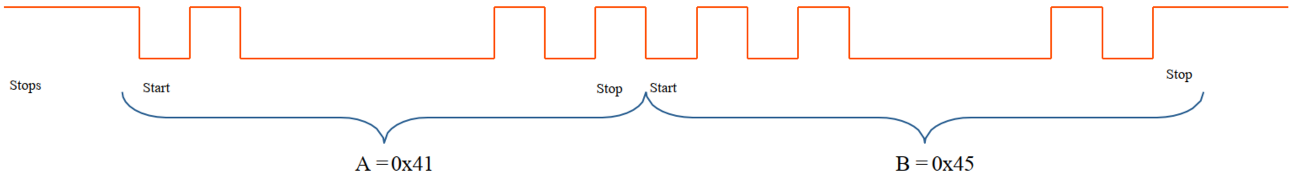
Synchronous and asynchronous communication

- Synchronous communication
  - Data transmitted as a steady stream at regular intervals
  - All transmitted bits are synchronized to a common clock signal
  - The two devices initially synchronize themselves to each other, and then continually send characters to stay synchronized



Synchronous and asynchronous communication

- Asynchronous communication
  - Data transmitted intermittently at irregular intervals
  - Each device uses its own internal clock resulting in bytes that are transferred at arbitrary times
  - Instead of using time as a way to synchronize the bits, the data format is used
  - Data transmission is synchronized using the start bit of the word, while one or more stop bits indicate the end of the word



Synchronous and asynchronous communication: differences

- Synchronous communication
  - Requires common clock
  - Whoever controls the clock controls communication speed
  - Faster

- Asynchronous communication
  - Has no clock
  - Speed must be agreed upon beforehand (accomplished via baud rate configuration)
  - More flexible

Communication modes

Simplex Mode
Transmission is possible only in one direction.

Half-duplex Mode
Data is transmitted in one direction at a time but the direction can be changed.

Full-duplex Mode
Data may be transmitted simultaneously in both directions.

Point-to-point and shared buses

- Point-to-point
  - 1:1 communication
- Bus
  - Shared among multiple devices
  - Master devices initiate a data transfer on the bus
  - Slave devices cooperate with the master
    - Roles/relationships are not permanent
  - Need an arbitration mechanism for multiple master systems to prevent data corruption if two or more masters simultaneously initiate data transfer

Newer device naming schemes:
- Controller/Leader
- Peripheral/Follower

Common external communication protocols

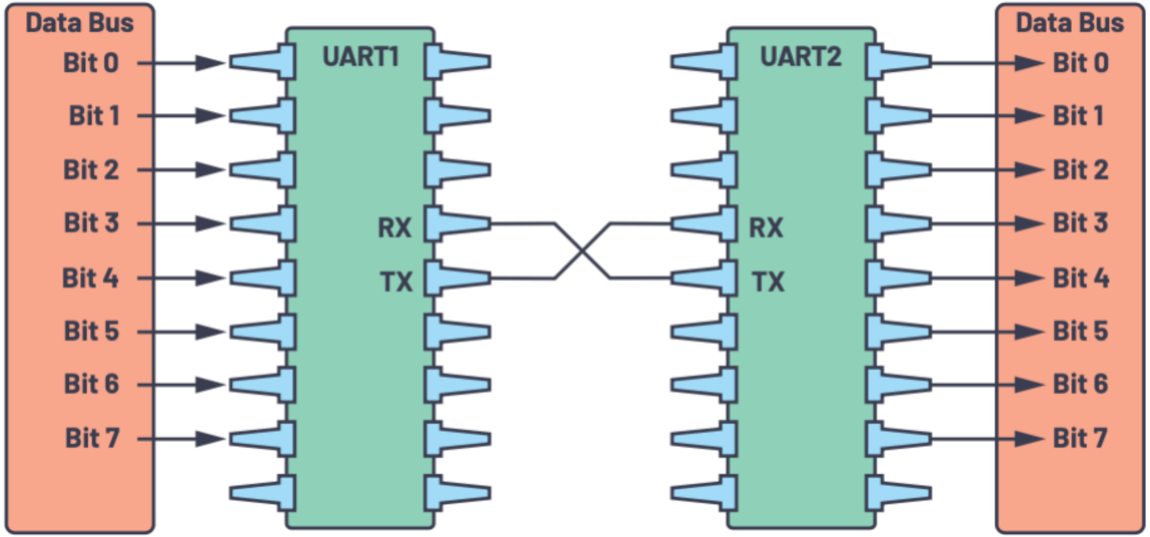
- Parallel (not so much anymore)
- UART (Universal Asynchronous Receiver Transmitter)
  still common in some communication and GPS devices
- USART (Universal Synchronous Asynchronous Receiver Transmitter)
  more advanced version
- SPI (Serial Peripheral Interface) very common
- I2C (Inter-Integrated Circuit) very common

UART

- Universal Asynchronous Receiver Transmitter
- PROTOCOL: asynchronous; point-to-point; full-duplex
- NUMBER OF WIRES: 2, one for transmission and one for reception
- SPEED: low-medium, configurable, generally between 9600bps and some Mbps
- TYPICAL APPLICATIONS:
  - point-to-point communication for debugging, generally over RS232 or USB physical channels
  - data transfer between microcontroller and PC
  - communication with low-speed modules

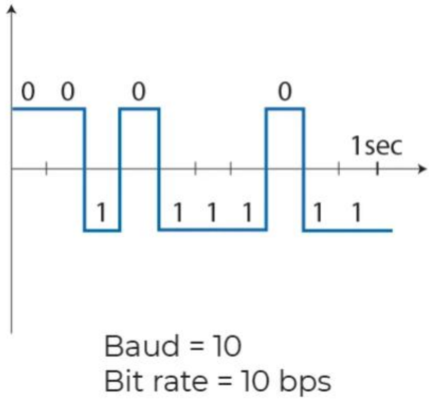
UART

- Connected to a bus which sends data in parallel. The UART than serializes them

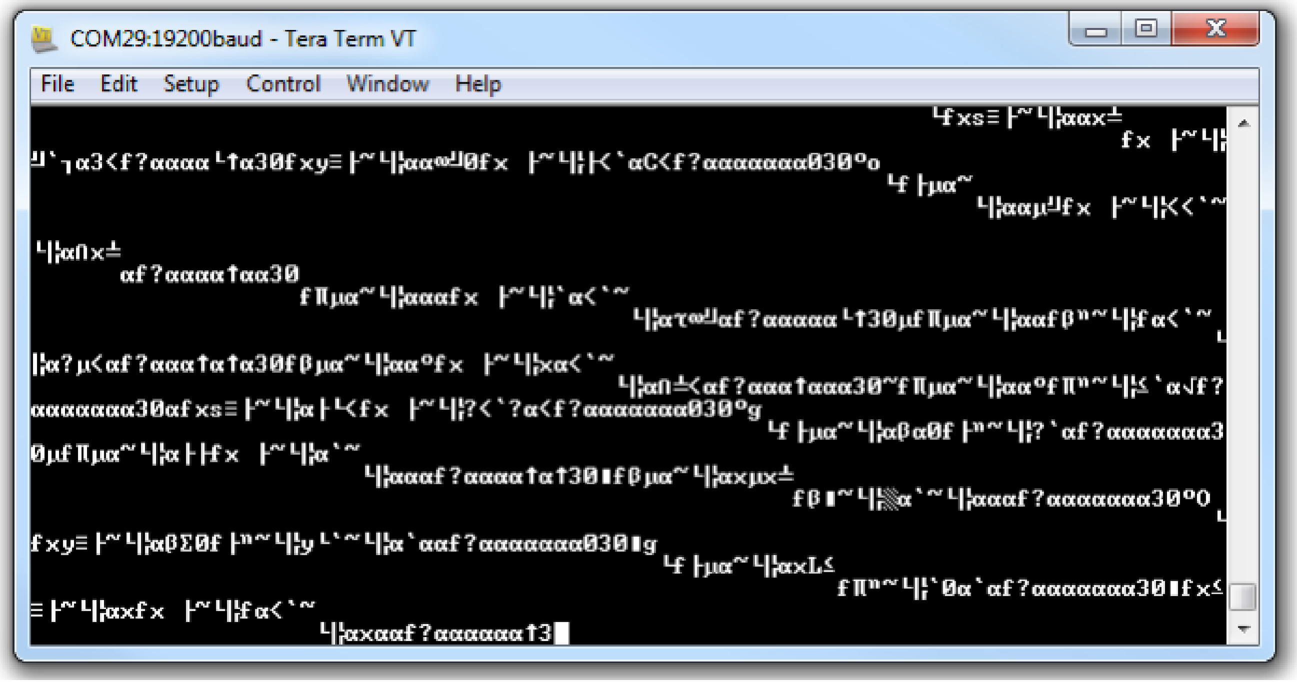


UART configuration: baud rate

- UART communication speed is defined by its symbol rate measured in baud
  - 1 baud (Bd) = 1 symbol per second
  - In UART, a symbol has two values (0/1), aka a single bit
  - This number includes both data payload and protocol bits, aka the physical or gross bit rate
    - Not to be confused with the effective or net data rate
- Typical values:
  - 9600 Bd
  - 115200 Bd
- Transmitter and receiver must use the same baud rate

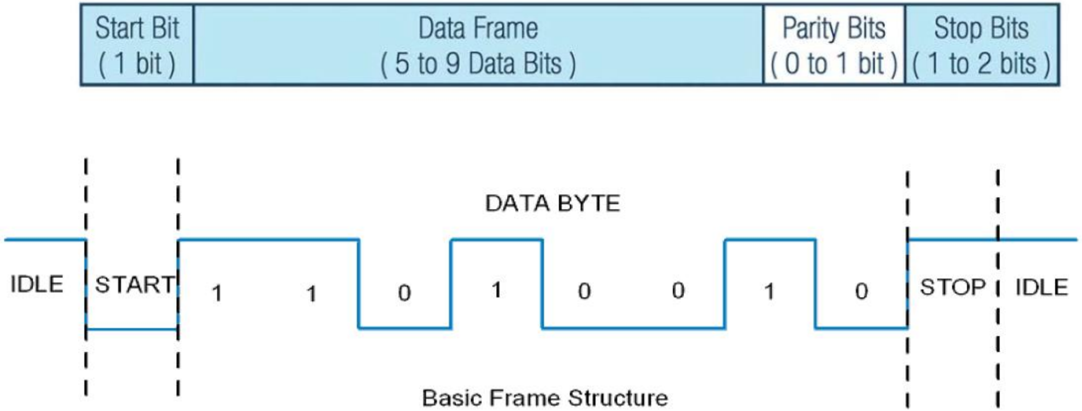


Baud rate mismatch



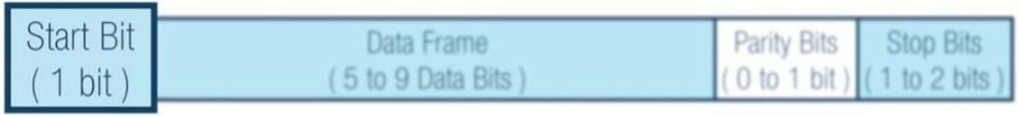
UART data transmission

- In UART, data transmission is done in form of packets
- Transmitter and receiver must be configured to transmit and receive the same data packet structure.



UART data transmission

- START BIT
  - When no transmission, the line is set to high (1)
  - When receiver detects a high (1) to low (0) transition, it starts sampling the incoming data

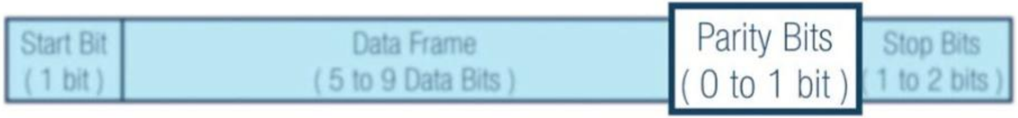


UART data transmission

- DATA FRAME
  - Minimum of 5 bits
  - Maximum of 8 bits if parity bit is used
  - Maximum of 9 bits if no parity bit

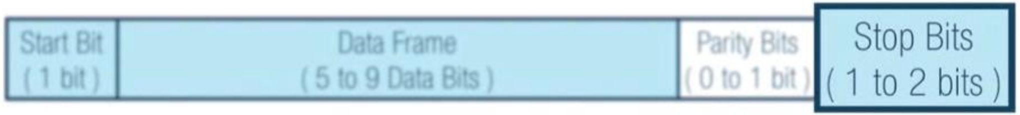
UART data transmission

- PARITY BIT
  - Used to say if the number of 1s in the data frame is even (parity bit = 0) or odd (parity bit = 1)
  - The receiver counts the number of received 1s and compares it with the parity bit. If they differ one bit changed during transmission -> ERROR
  - Good for single bit flips during transmission



UART data transmission

- STOP BIT
  - Low to high transition
  - Can last 1 or 2 cycles

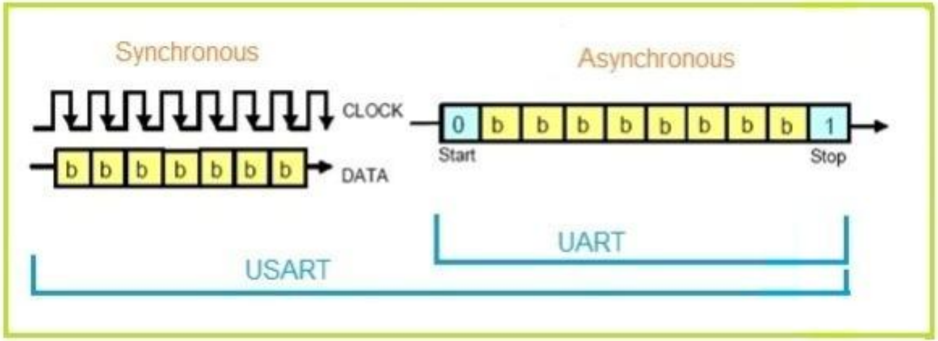


UART

- Pros
  - Only uses two wires
  - No clock signal is necessary
  - Has a parity bit to allow for error checking
  - The structure of the data packet can be changed as long as both sides are set up for it
  - Well documented and widely used method
- Cons
  - The size of the data frame is limited to a maximum of 9 bits
  - Doesn't support multiple slave or multiple master systems
  - The baud rates of each UART must be within 10% of each other

USART

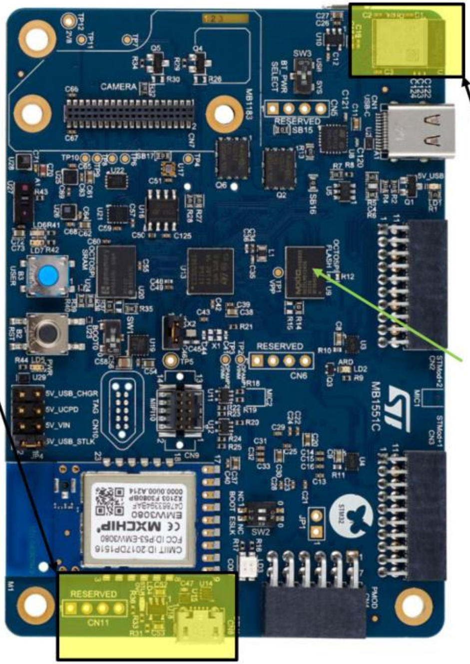
- Universal Synchronous Asynchronous Receiver Transmitter
- Two transmission lines: one for clock and one for data
- No need for start and stop bits, since receiver and transmitter are synchronized



UART connected modules in TI MSP432

DEBUGING CONNECTION

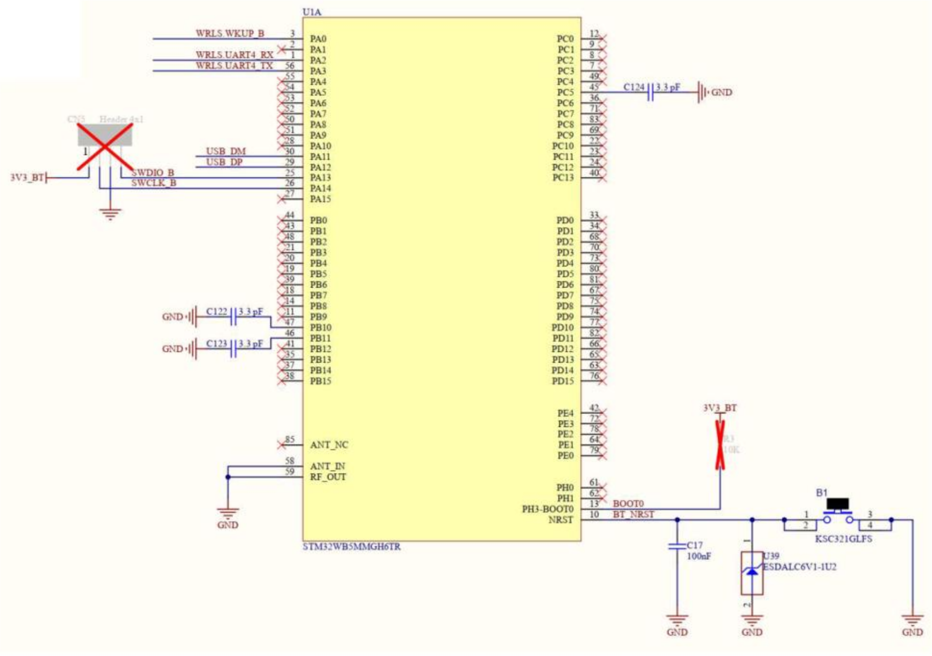
- ST-LINK
- UART1

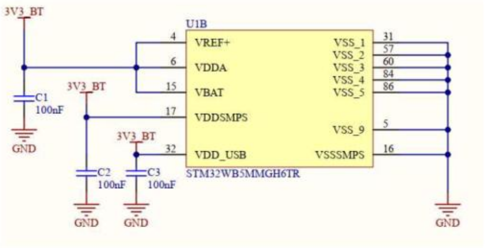


Bluetooth Module

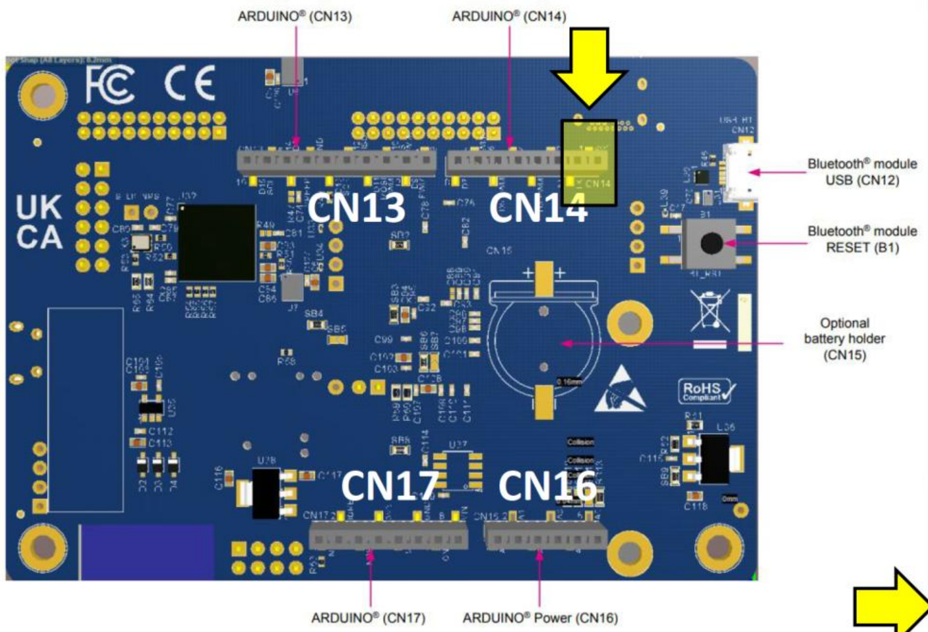
- STM32WB5MMGH6TR
- UART4

UART connected modules in TI MSP432: schematics





UART external connections



https://www.st.com/en/evaluation-tools/b-u585i-iot02a.html

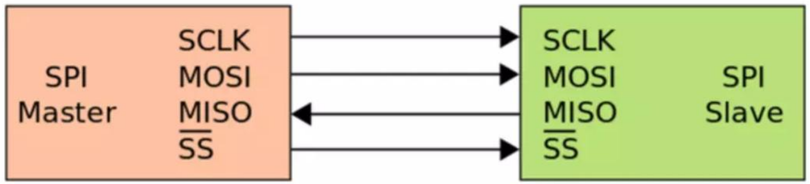
Connector	Pin number	Pin name	Signal name	STM32L 4+ pin	Function
CN17	1	NC	-	-	-
	2	IOREF	-	-	3.3 V reference
	3	NRST	NRST	NRST	Reset
	4	3V3	-	-	3.3 V I/O
	5	5V	-	-	5 V
	6	GND	-	-	GND
	7	GND	-	-	GND
	8	VIN	-	-	Power input
CN16	1	A0	ARD_ADC_A0	PC0	ADC
	2	A1	ARD-ADC_A1	PC2	ADC
	3	A2	ARD-ADC_A2	PC4	ADC
	4	A3	ARD-ADC_A3	PC5	ADC
	5	A4	ARD-ADC_A4	PA7	ADC / I2C1_SDA
	6	A5	ARD-ADC_A5	PB0	ADC / I2C1_SCL
CN13	10	SCL/D15	I2C1_SCL	PB8	I2C1_SCL
	9	SDA/D14	I2C1_SDA	PB9	I2C1_SDA
	8	VREFP	VREFP	-	VREFP
	7	GND	GND	-	GND
	6	SCK/D13	ARD.D13_SPI1_SCK	PE13	SPI1_SCK / LD2
	5	MISO/D12	ARD.D12_SPI1_MISO	PE14	SPI1_MISO
	4	PWM/MOSI/D11	ARD.D11_SPI1_MOSI	PE15	SPI1_MOSI / TIM1_CH4N
	3	PWM/CS/D10	ARD.D10_TIM_SPI1_NSS	PE12	SPI1_NSS / TIM1_CH3N
	2	PWM/D9	ARD.D9_TIM	PA8	TIM1_CH1
	1	D8	ARD.D8_IO	PC1	GPIO
CN14	8	D7	ARD.D7_IO	PF13	GPIO
	7	PWM/D6	ARD.D6_TIM	PB6	TIM4_CH1
	6	PWM/D5	ARD.D5_TIM	PE0	TIM16_CH1
	5	D4	ARD.D4_INT	PE7	GPIO
	4	PWM/D3	ARD.D3_TIM	PB2	TIM8_CH4N
	3	D2	ARD.D2_IO	PD15	GPIO
	2	TX/D1	ARD.D1_TX	PD8	UART3_TX
	1	RX/D0	ARD.D0_RX	PD9	UART3_RX

SPI

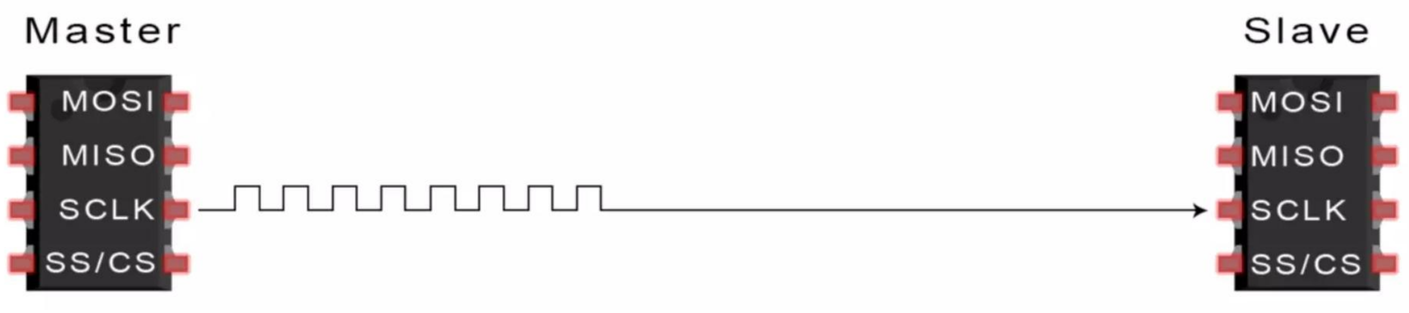
- Serial Peripheral Interface
- PROTOCOL: synchronous; single master/multi-slave; full-duplex
- NUMBER OF WIRES: 4 typically, two for data and two for clock; 6 maximum
- SPEED: fast, up to tens of Mbps
- TYPICAL APPLICATIONS:
    - High-speed communication with flash memory
    - High-speed communication with sensors, Analog to Digital and Digital to Analog Converters
    - High-speed communication with displays

SPI 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 – Unique line to select a slave
- SCLK – Master produced clock to synchronize data transfer
- Note: names are not standard

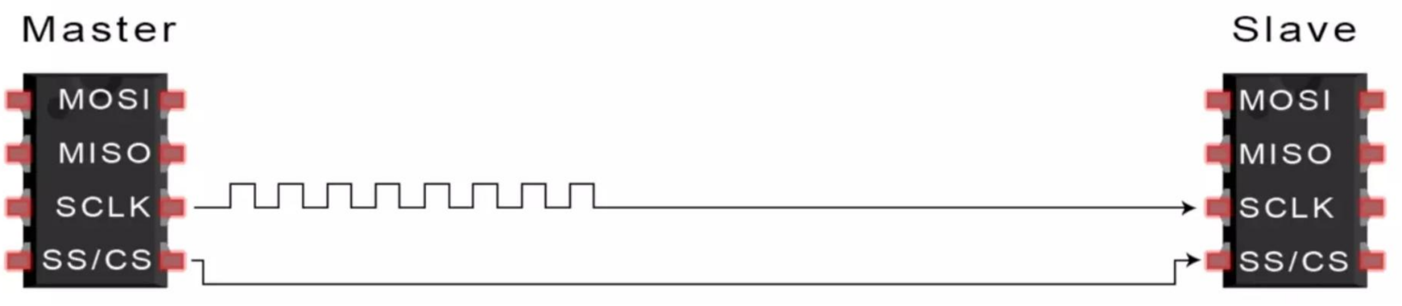


SPI data transmission



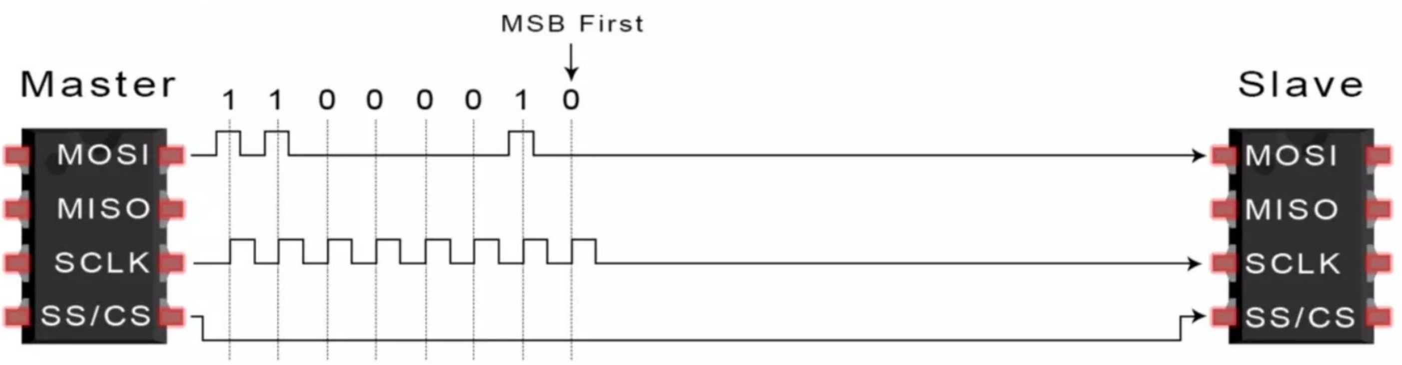
- Master outputs clock signal

SPI data transmission



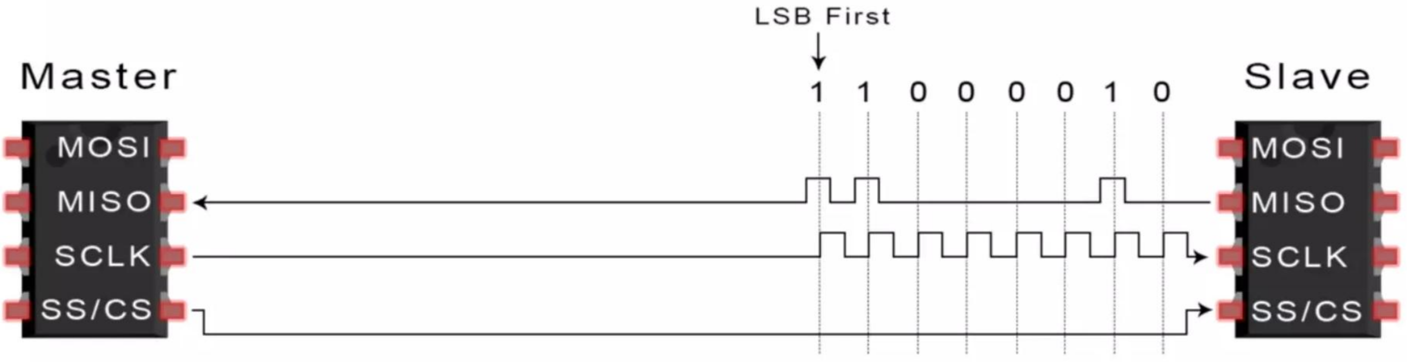
- Master switches the SS pin to low voltage state, which activates Slave

SPI data transmission



- Master sends data one bit at a time to Slave along MOSI line
- Slave reads the bits as they are being received

SPI data transmission

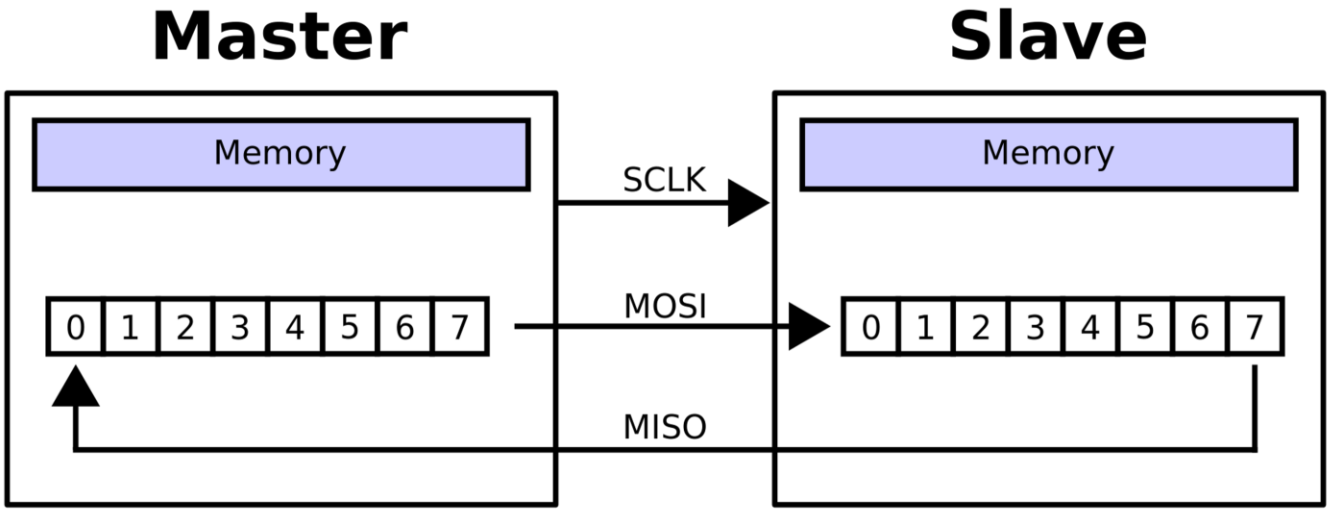


- If a response is needed, Slave returns data one bit at a time to Master along MISO line
- Master reads the bits as they are being received

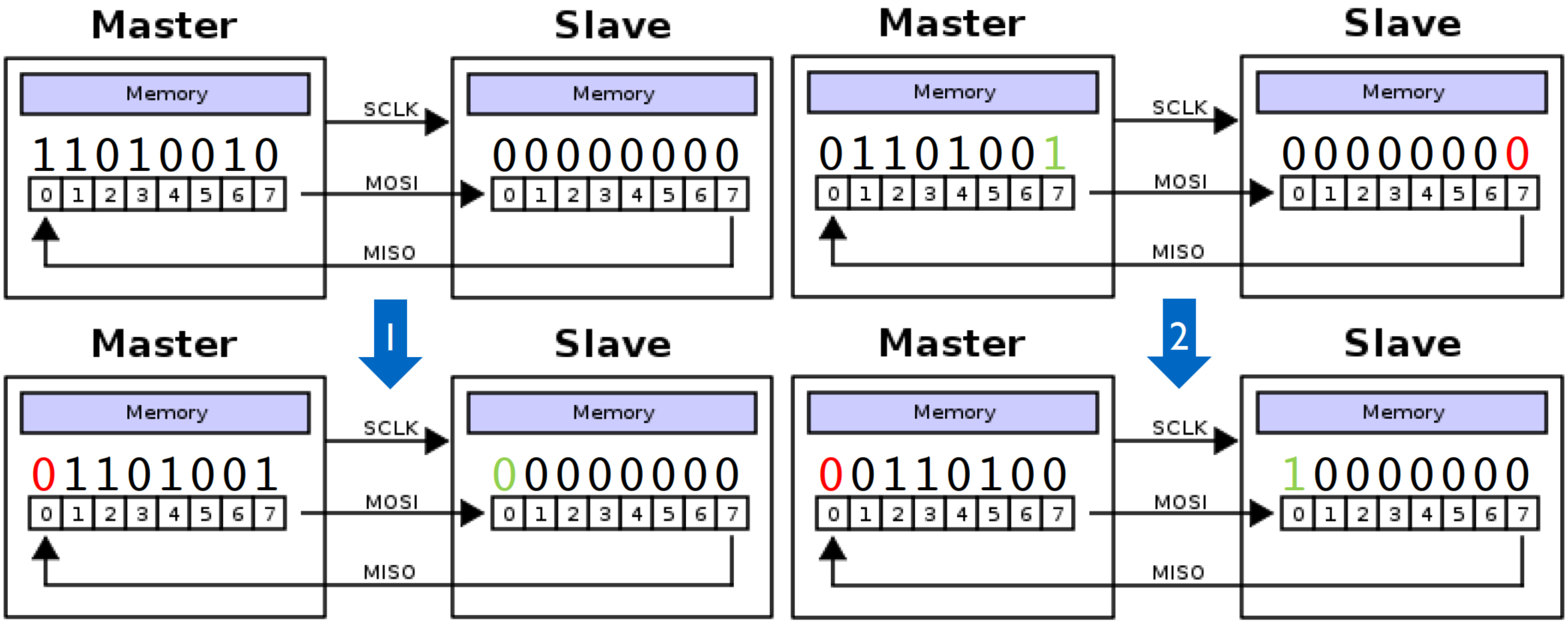
SPI data transmission in an oscilloscope

SPI basic data loop

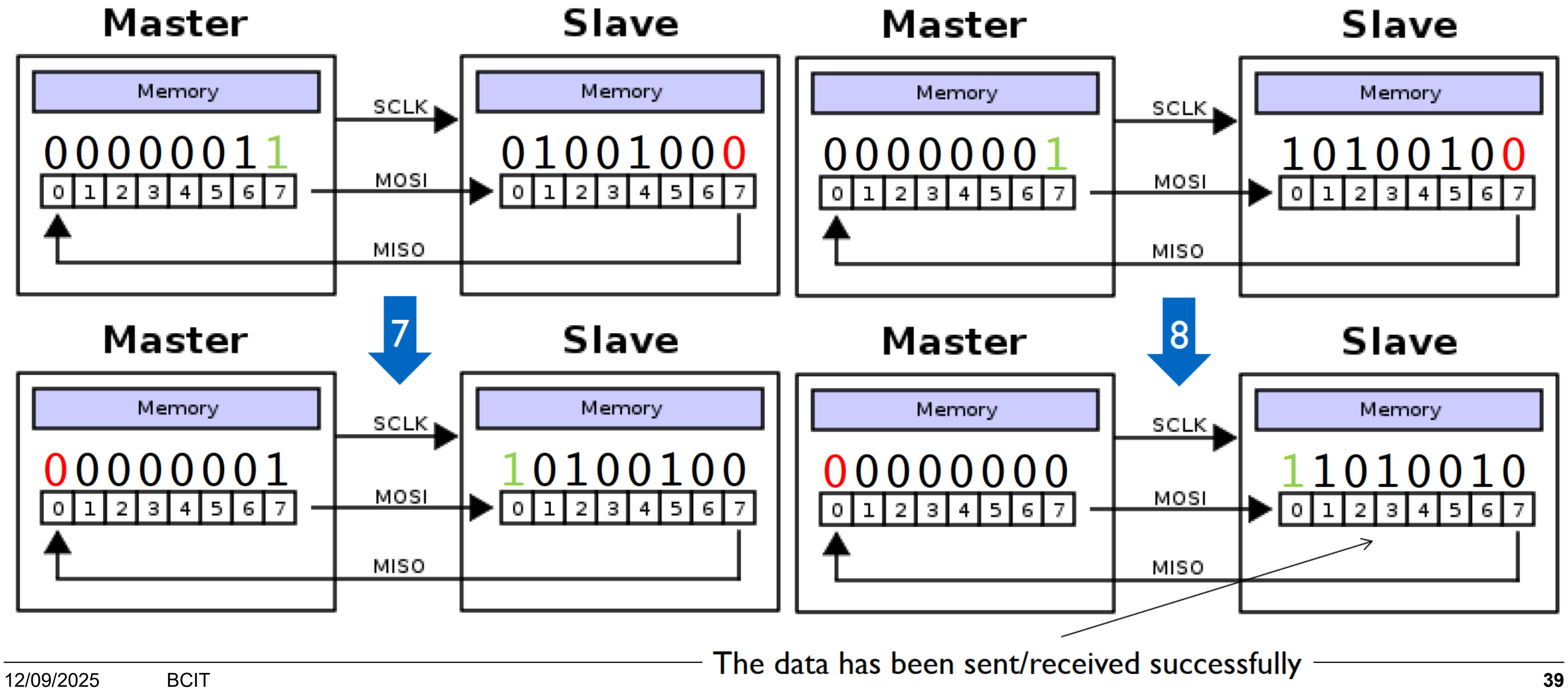
- Master/Slave shift register stores 4 to 16 bits



SPI basic data loop

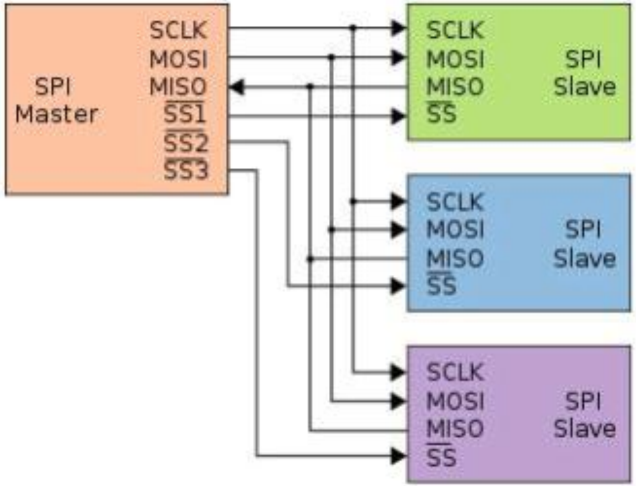


SPI basic data loop

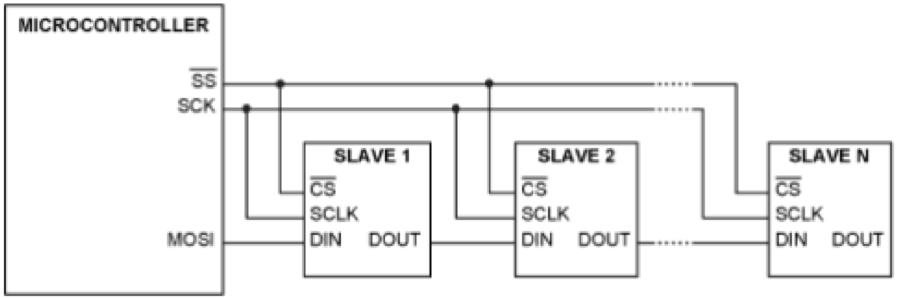


SPI multi-slave configurations

- Master and multiple independent slaves



- Master and daisy-chained slaves



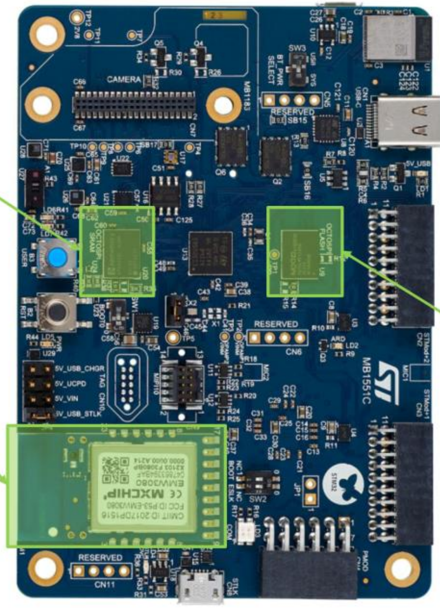
SPI

- Pros:
  - Fast for point-to-point connections (almost 2x I2C data transfer speeds)
  - No start or stop bits, so easily allows streaming without interruption
  - No complicated slave addressing so less overhead
  - Everyone supports it
- Cons:
  - Uses four wires
  - SS makes multiple slaves very complicated
  - No acknowledgement that the data has been successfully received
  - No form of error checking

SPI connected modules in TI MSP432

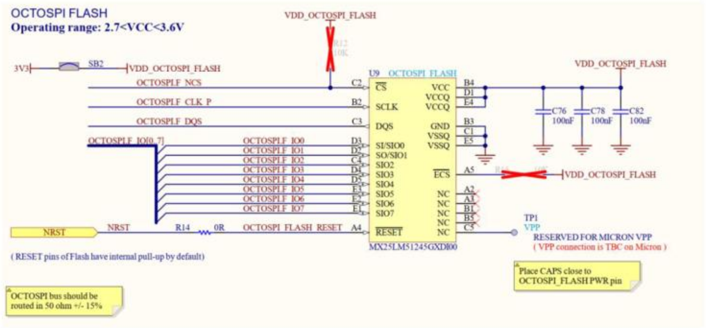
12-Mbit Octo-SPI Flash
- MX25LM51245GXDI005
- OCTOSPI2

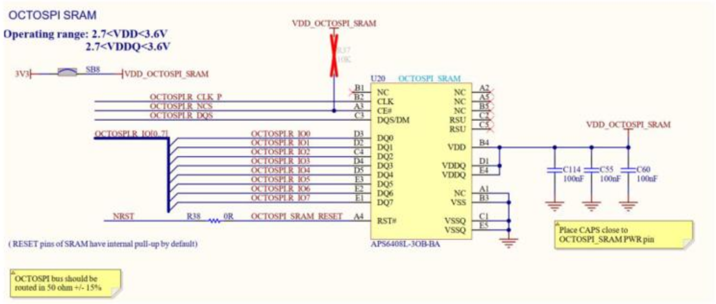
Wifi Module
- EMW3080
- SPI2

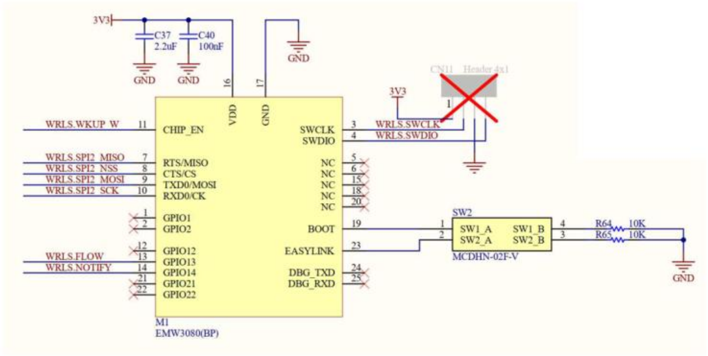


12-Mbit Octo-SPI RAM
- APS6408L-3OB-BA
- Octo-SPI

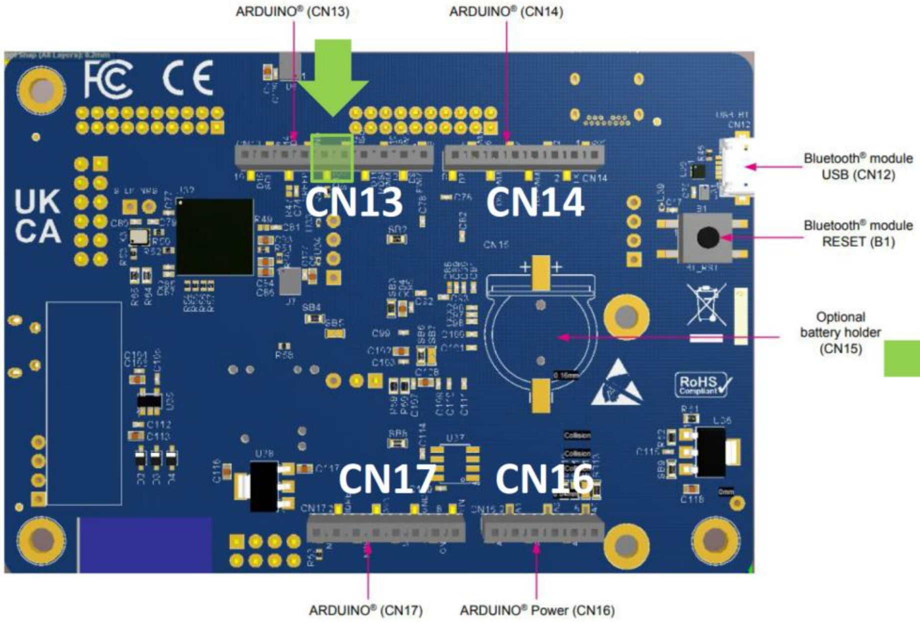
SPI connected modules in TI MSP432: schematics







SPI external connections



https://www.st.com/en/evaluation-tools/b-u585i-iot02a.html

Connector	Pin number	Pin name	Signal name	STM32L 4+ pin	Function
CN17	1	NC	-	-	-
	2	IOREF	-	-	3.3 V reference
	3	NRST	NRST	NRST	Reset
	4	3V3	-	-	3.3 V I/O
	5	5V	-	-	5 V
	6	GND	-	-	GND
	7	GND	-	-	GND
	8	VIN	-	-	Power input
CN16	1	A0	ARD_ADC_A0	PC0	ADC
	2	A1	ARD-ADC_A1	PC2	ADC
	3	A2	ARD-ADC_A2	PC4	ADC
	4	A3	ARD-ADC_A3	PC5	ADC
	5	A4	ARD-ADC_A4	PA7	ADC / I2C1_SDA
	6	A5	ARD-ADC_A5	PB0	ADC / I2C1_SCL
CN13	10	SCL/D15	I2C1_SCL	PB8	I2C1_SCL
	9	SDA/D14	I2C1_SDA	PB9	I2C1_SDA
	8	VREFP	VREFP	-	VREFP
	7	GND	GND	-	GND
	6	SCK/D13	ARD.D13_SPI1_SCK	PE13	SPI1_SCK / LD2
	5	MISO/D12	ARD.D12_SPI1_MISO	PE14	SPI1_MISO
	4	PWM/MOSI/D11	ARD.D11_SPI1_MOSI	PE15	SPI1_MOSI / TIM1_CH4N
	3	PWM/CS/D10	ARD.D10_TIM_SPI1_NSS	PE12	SPI1_NSS / TIM1_CH3N
	2	PWM/D9	ARD.D9_TIM	PA8	TIM1_CH1
	1	D8	ARD.D8_IO	PC1	GPIO
CN14	8	D7	ARD.D7_IO	PF13	GPIO
	7	PWM/D6	ARD.D6_TIM	PB6	TIM4_CH1
	6	PWM/D5	ARD.D5_TIM	PE0	TIM16_CH1
	5	D4	ARD.D4_INT	PE7	GPIO
	4	PWM/D3	ARD.D3_TIM	PB2	TIM8_CH4N
	3	D2	ARD.D2_IO	PD15	GPIO
	2	TX/D1	ARD.D1_TX	PD8	UART3_TX
	1	RX/D0	ARD.D0_RX	PD9	UART3_RX

I2C

- Inter-Integrated Circuit
- PROTOCOL: synchronous; multi-master/multi-slave; half-duplex
- NUMBER OF WIRES: 2, one for data and one for clock
- SPEED: from 100kbps to 5 Mbps
- TYPICAL APPLICATIONS:
  - Low-speed communication between multiple devices
  - Connection of sensors, displays and flash memory to microcontrollers

I2C addressing

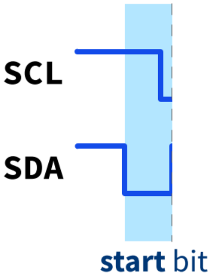
- Devices come with their own address
  - Originally 7 bits long
  - More modern devices have 10-bit addresses
- Allows potentially up to 2^7 devices (or 2^{10}) on a bus

- Addresses starting with 0000 or 1111 have special functions:
  - 0000000 is a General call broadcast – to address every device on the bus

I2C data transmission

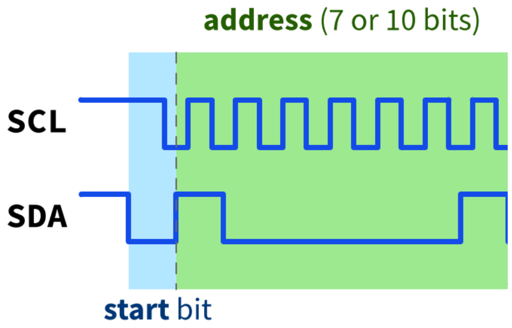
- In idle, both the Serial Clock (SCL) and Serial Data (SDA) lines are pulled-up to 1
  - SCL— the line that carries the clock signal
  - SDA — the line for the master and slave to send and receive data.

I2C data transmission



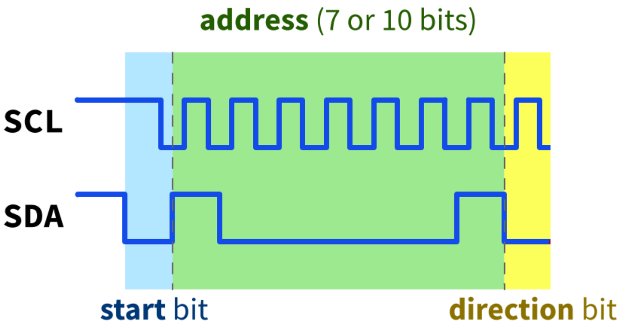
- To start the communication, the master:
    - asserts the start bit (SDA 1->0 transition while SCL is still 1)
    - then, it starts generating the SCL clock
    - except for the start and stop bits, SDA transitions only when SCL is 0

I2C data transmission



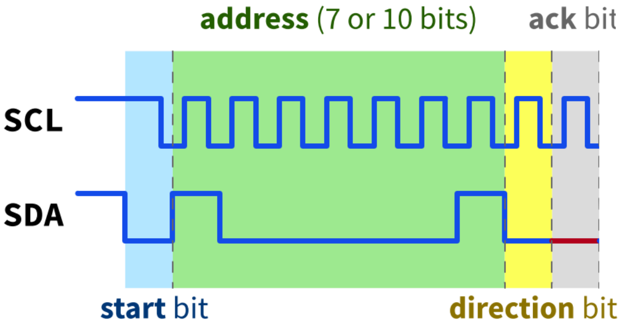
- The master transmits the slave address:
  - broadcasted to all devices on the I2C bus
  - used to select the target slave
  - all other devices will ignore until STOP signal appears later on

I2C data transmission



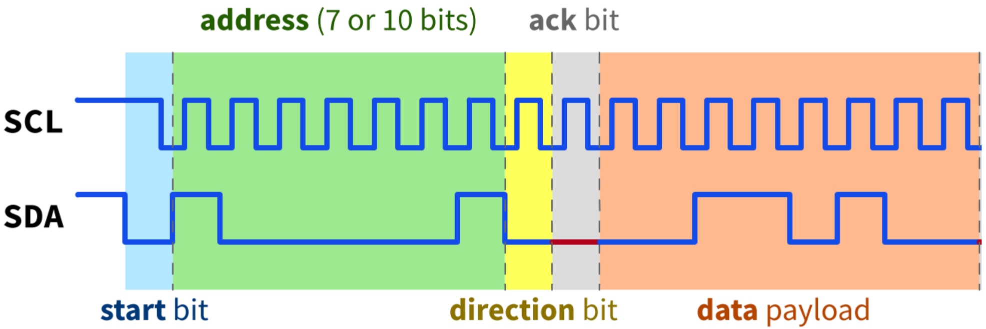
- The master transmits a direction bit:
  - a 0 for master->slave (write) transfer
  - a 1 for slave->master (read) transfer

I2C data transmission



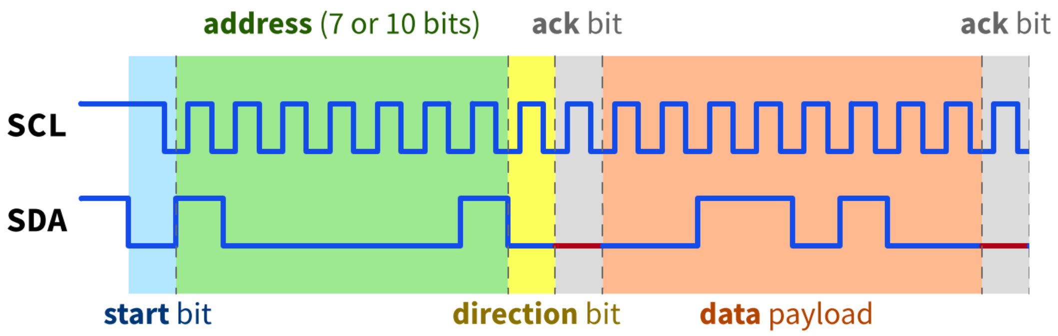
- The slave then acknowledges reception:
  - by driving SDA to 0
  - if not acknowledged, the transaction must be repeated by the master

I2C data transmission



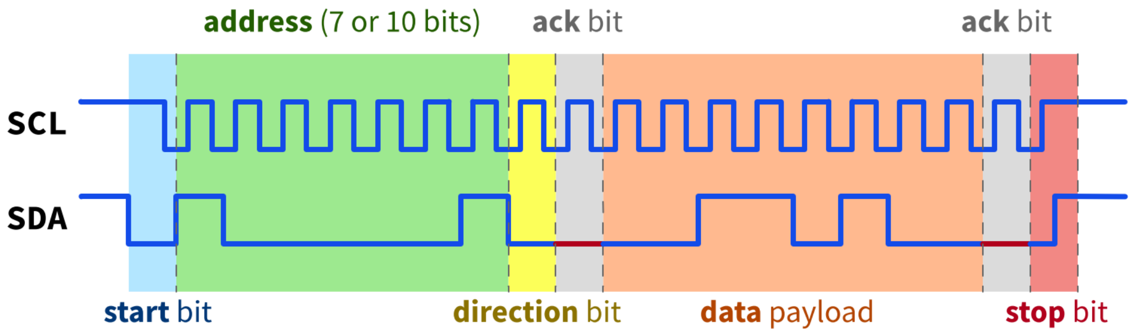
- The master transmits its data payload:
  - each payload packet is 8 bits
  - there might be more than one packet, depending on application

I2C data transmission



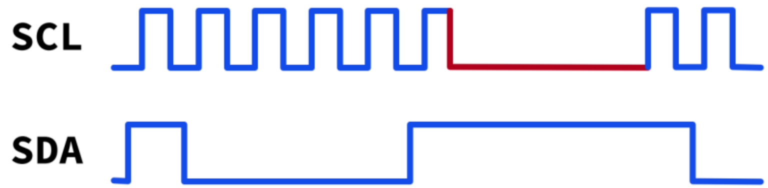
- The slave acknowledges reception of the data packet:
  - 1 ack bit every 8 payload bits
  - slave must acknowledge each packet

I2C data transmission



- At the end of the transfer, the master transmits a stop bit:
  - first, it sets SDA to 0
  - then it releases SCL (i.e. it lets it go to 1)
  - finally, it releases SDA which also goes to 1
- Reads work similarly, but data transfer – ack roles are reversed

I2C data transmission: clock stretching

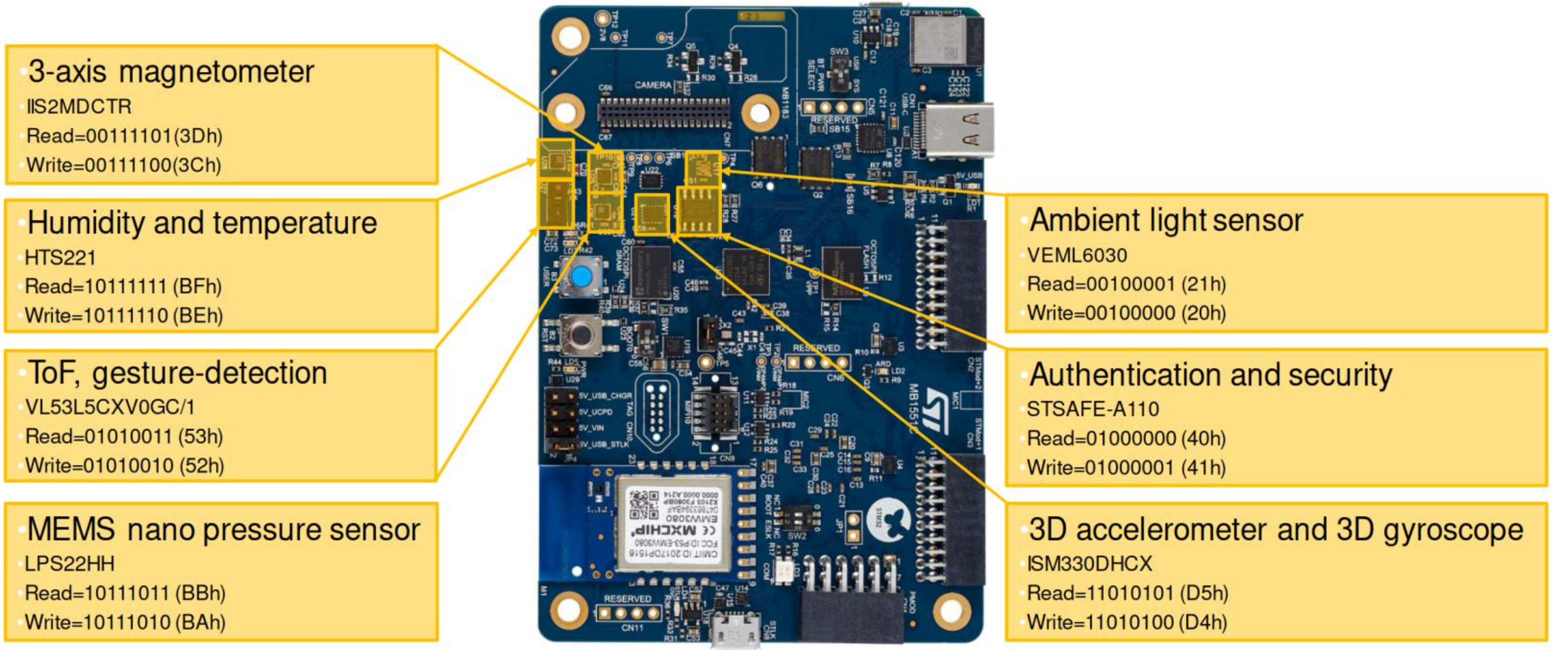


- Slave can ask for more time to process a bit by clock stretching:
  - drive SCL to 0 if in need of more processing time
  - form of flow control

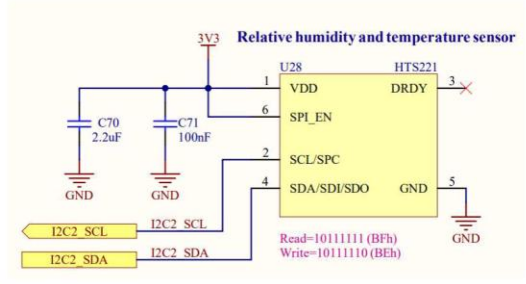
I2C

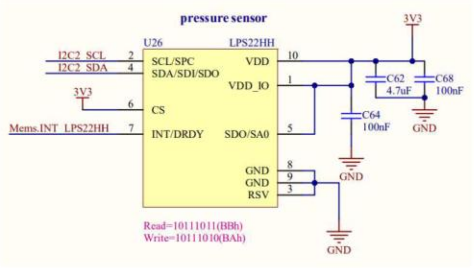
- Pros
  - Only uses two wires
  - Supports multiple masters and multiple slaves
  - Acknowledgement bit gives confirmation that each frame is transferred successfully
  - Hardware is less complicated than with UARTs
  - Well known and widely used protocol
- Cons
  - Slower data transfer rate than SPI
  - The size of the data frame is limited to 8 bits
  - More complicated hardware needed to implement than SPI

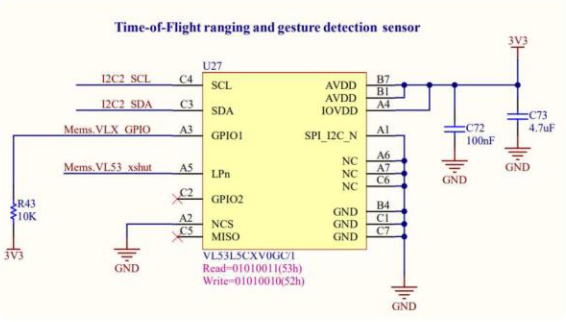
I2C connected sensors in TI MSP432

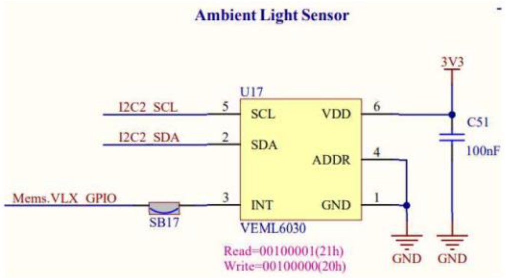


I2C connected sensors in TI MSP432: schematics

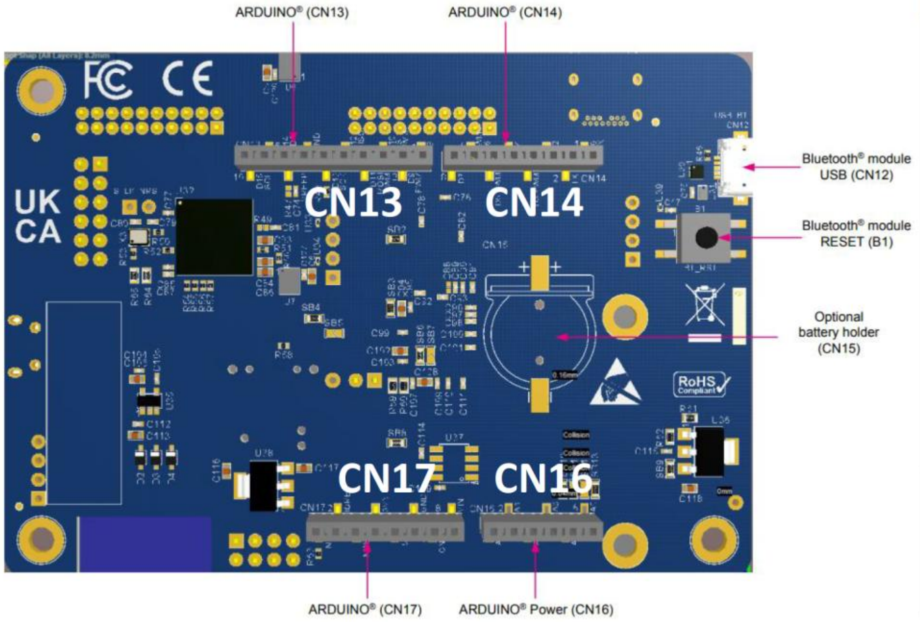








I2C external connections



Connector	Pin number	Pin name	Signal name	STM32L 4+ pin	Function
CN17	1	NC	-	-	-
	2	IOREF	-	-	3.3 V reference
	3	NRST	NRST	NRST	Reset
	4	3V3	-	-	3.3 V I/O
	5	5V	-	-	5 V
	6	GND	-	-	GND
	7	GND	-	-	GND
	8	VIN	-	-	Power input
CN16	1	A0	ARD_ADC_A0	PC0	ADC
	2	A1	ARD-ADC_A1	PC2	ADC
	3	A2	ARD-ADC_A2	PC4	ADC
	4	A3	ARD-ADC_A3	PC5	ADC
	5	A4	ARD-ADC_A4	PA7	ADC / I2C1_SDA
	6	A5	ARD-ADC_A5	PB0	ADC / I2C1_SCL
CN13	10	SCL/D15	I2C1_SCL	PB8	I2C1_SCL
	9	SDA/D14	I2C1_SDA	PB9	I2C1_SDA
	8	VREFP	VREFP	-	VREFP
	7	GND	GND	-	GND
	6	SCK/D13	ARD.D13_SPI1_SCK	PE13	SPI1_SCK / LD2
	5	MISO/D12	ARD.D12_SPI1_MISO	PE14	SPI1_MISO
	4	PWM/MOSI/D11	ARD.D11_SPI1_MOSI	PE15	SPI1_MOSI / TIM1_CH4N
	3	PWM/CS/D10	ARD.D10_TIM_SPI1_NSS	PE12	SPI1_NSS / TIM1_CH3N
	2	PWM/D9	ARD.D9_TIM	PA8	TIM1_CH1
	1	D8	ARD.D8_IO	PC1	GPIO
CN14	8	D7	ARD.D7_IO	PF13	GPIO
	7	PWM/D6	ARD.D6_TIM	PB6	TIM4_CH1
	6	PWM/D5	ARD.D5_TIM	PE0	TIM16_CH1
	5	D4	ARD.D4_INT	PE7	GPIO
	4	PWM/D3	ARD.D3_TIM	PB2	TIM8_CH4N
	3	D2	ARD.D2_IO	PD15	GPIO
	2	TX/D1	ARD.D1_TX	PD8	UART3_TX
	1	RX/D0	ARD.D0_RX	PD9	UART3_RX

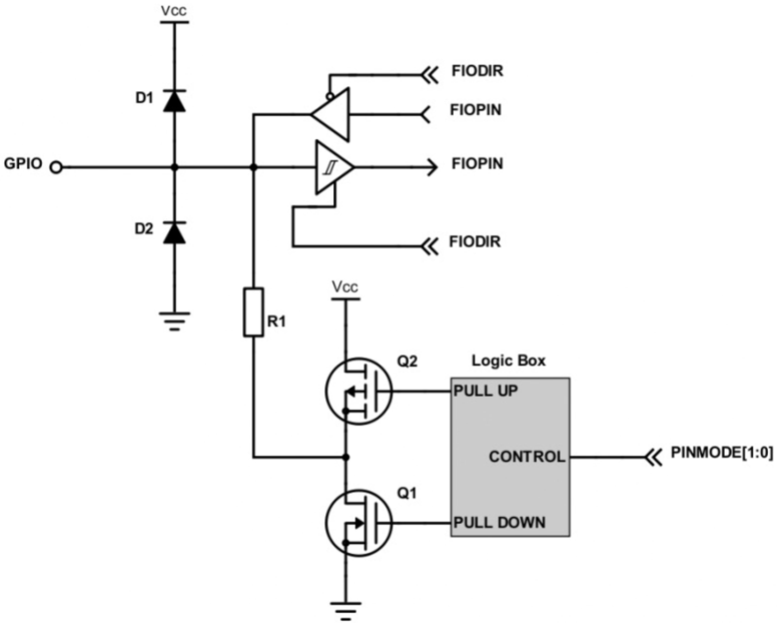
https://www.st.com/en/evaluation-tools/b-u585i-iot02a.html

GPIO

- General Purpose Input-Output
- PROTOCOL: digital I/0 controllable by software
    - 5V = logic 1 (for 5V CPUs)
    - 0V = logic 0
- DIRECTION: configurable as input or output
- TYPICAL APPLICATIONS:
    - Controlling LEDs, reading button states, driving relays
    - Sending control signals, or used as physical pins for one of the previous protocols
    - Interfacing with simple digital sensors or switches

GPIO

- Essential building block to interface with external components
- Several building blocks to manage different electrical conditions
- Generally designed to work with small currents
- Supports features like:
    - Connecting to UART, SPI, or I2C peripherals
    - Electrostatic discharge protection
    - Hardware debouncing



USB

- Universal Serial Bus
- PROTOCOL: serial; synchronous; single master, multi-slave
- NUMBER OF WIRES: 2, D+ and D-, differential signal for noise immunity
- SPEED: 1.5 Mbps to 80 Gbps
- TYPICAL APPLICATIONS:
  - Printers
  - Keyboards
  - Game-pads / joysticks
  - Mass external storage

CAN

- Controller Area Network
- PROTOCOL: serial; asynchronous; multi-master
- NUMBER OF WIRES: 2, CAN high and CAN low, differential signal for noise immunity
- SPEED: up to Mbps
- TYPICAL APPLICATIONS:
  - Automotive and industrial systems
  - Reliable, noise resistant data transfer
  - Communication between Electronic Control Units (ECUs)