Lecture 14: Communication protocols

Vikram Iyer

Adapted from material by Blake Hannaford

Announcements and reminders

Lab 3 is due 11/21

- ~~Deadline extended to Wednesday~~
- Tip: for scheduling, start with simple tasks

Plan for today

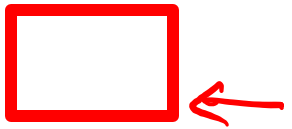
- Go over exam

- Finish UART, I2C and SPI

Serial Communication: RS232C

Goal: Send bits from point A to point B one at a time

RS232C: An international standard for serial communication (1960s). Very old but still influential in today's systems.



- This is how your Arduino communicates with the computer and why we do Serial.println
- FTDI chip converts between Serial and USB
- Some microcontrollers have native USB support
- Notation: RX = receive, TX = Transmit

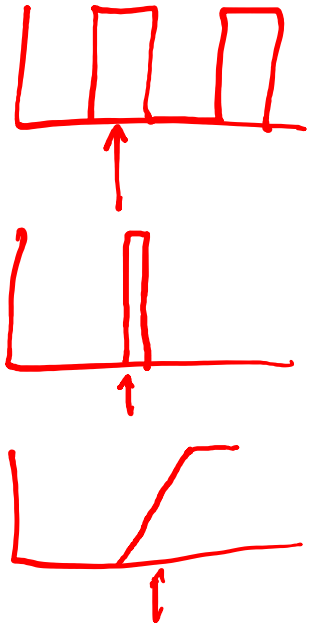
Synchronization

There is no common clock between sender and receiver. How does receiver know when to test levels?

`A1`: Synchronus mode — Continuous stream of characters.

`A2`: Asynchronus mode — start bit is known to be logic 0.

Falling (rising?) edge triggers a local clock in the receiver.



Multiplexing

Sending information in both directions.

TX = transmit RX = receiver

Terms:

Simplex One sender and one receiver. TX ⟶ RX

Half Duplex One side can send at a time. TX ⟷ RX

Full Duplex Both sides can send at same time.

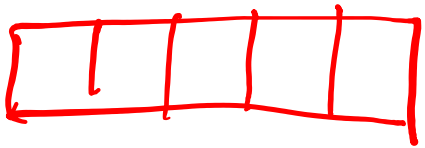
Connections

Must have:
TX → RX
RX ← TX
Cables are easier to make if pins are “straight through”. i.e. pin n connects to pin n.

23 pin (ancient, un-needed) 9 pin (IBM PC standard)
Two connector wiring types
DTE: pin 2 = RX, pin 3 = TX “Data Terminal Equipment”
DCE: pin 2 = TX, pin 3 = RX “Data Communications Equipment”
Thus, DCE can talk to DTE with a “straight through” cable.

Flow Control

Sometimes a slow device cannot handle data as fast as sender (even at same baud rate).
Receiver needs a way to stop and start sender.



Hardware solution: Additional wires
RTS “Request to Send”. DTE sets this to mark to allow data to be sent to it by DCE.
CTS “Clear to Send”. DCE sets this to mark to allow data to be sent to it by DTE.

Hardware flow control can be enabled/disabled.

Software solution: Special Control Characters
XON (ctl Q) Start Transmitting
XOFF(ctl S) Stop Transmitting

Error Control

Unfortunately, serial communication is subject to errors. Sources:

- Electrical interference
- Long wire lengths
- Ground potential differences between communicating systems.
- Timing errors between sender and receiver clocks (should be within 1%).

Solutions

Character Parity	9th data bit. Very weak and bandwidth hog.
Checksum	Add up a group of bytes. Send sum at end of packet.
Compare Parity	Send parity of some group of bits (rows or colums) in the packet.
CRC	Use theory of binary polynomials. Send a code so that combined message is divisible by some known polynomial.

Packets

Grouping bytes into packets gives structure to the serial communication. Packet structure must be same between software both sides.

0xAA
0101...

Example Packet:

Start	Type	Length	...	Checksum

Start A character which the software can recognize as the start of a packet. Ideally should not appear anywhere else in packet.

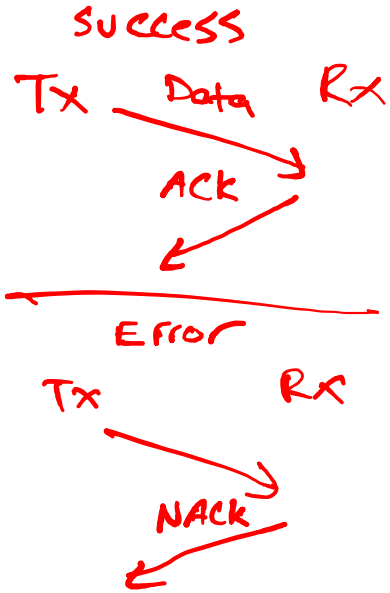
Type Kind of packet.

Length value which tells how long the packet will be.

... some number of data bytes (the payload).

Checksum Checksum computed on previous packet.

Packet Error control



What if a packet is received and the Checksum is wrong?

Normal:
Sender sends the packet and saves the packet.
Receiver checks Checksum and acknowledges receipt of packet.
Sender discards packet and sends next packet.

Error:
Sender sends and saves the packet.
Receiver detects error and sends negative acknowledge.
goto step 1.

ACK/NAK Special short packets
ACK Acknowledge NAK Negative Acknowledge

Start	ACK/NAK	Checksum

Protocol Issues

What Happens If:
Transmission is so unreliable that Checksum fails every time?

ACK/NAK packets are subject to errors?

Do we have to send ACK/NAK on receipt of an ACK/NAK?

Answers to all these questions must be worked out in a protocol spec.