Lecture 15: Serial Communication

Vikram Iyer

Adapted from material by Blake Hannaford

Announcements

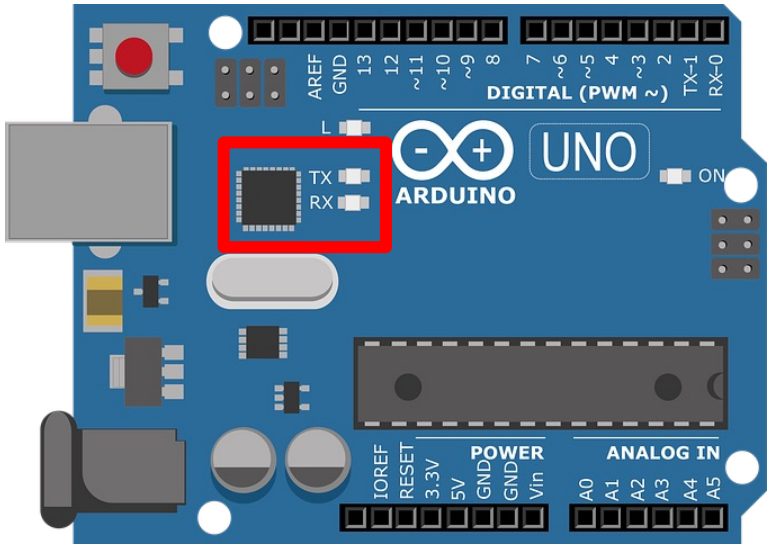
- Lab 3 due next Friday 5/26
- Lab 4 will be posted this weekend, due at end of finals week
- Pick up midterms in OH

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



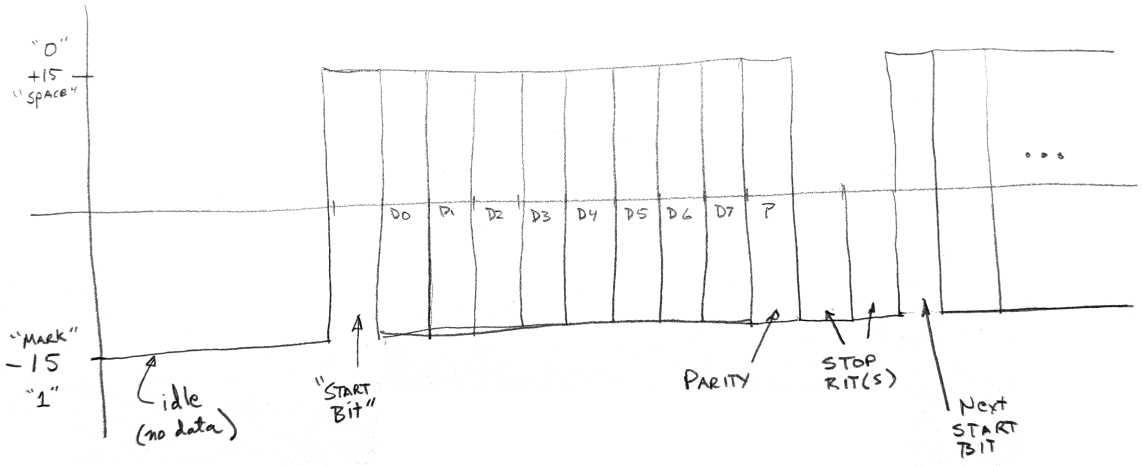
Electrical Signaling

+2.5V — +15V	logic 0	“space”
-2.5V — -15V	logic 1	“mark”

- When first designed the high voltage margin gave more immunity to noise
- These days it’s a hassle since many systems don’t otherwise need +15V, -15V supplies, modern processes can’t support high voltages
- Requires special driver chips for +- voltages.
- Arduino and many other systems use
  +5V → logic 0
  0V → logic 1
- Lower voltage saves power supply costs

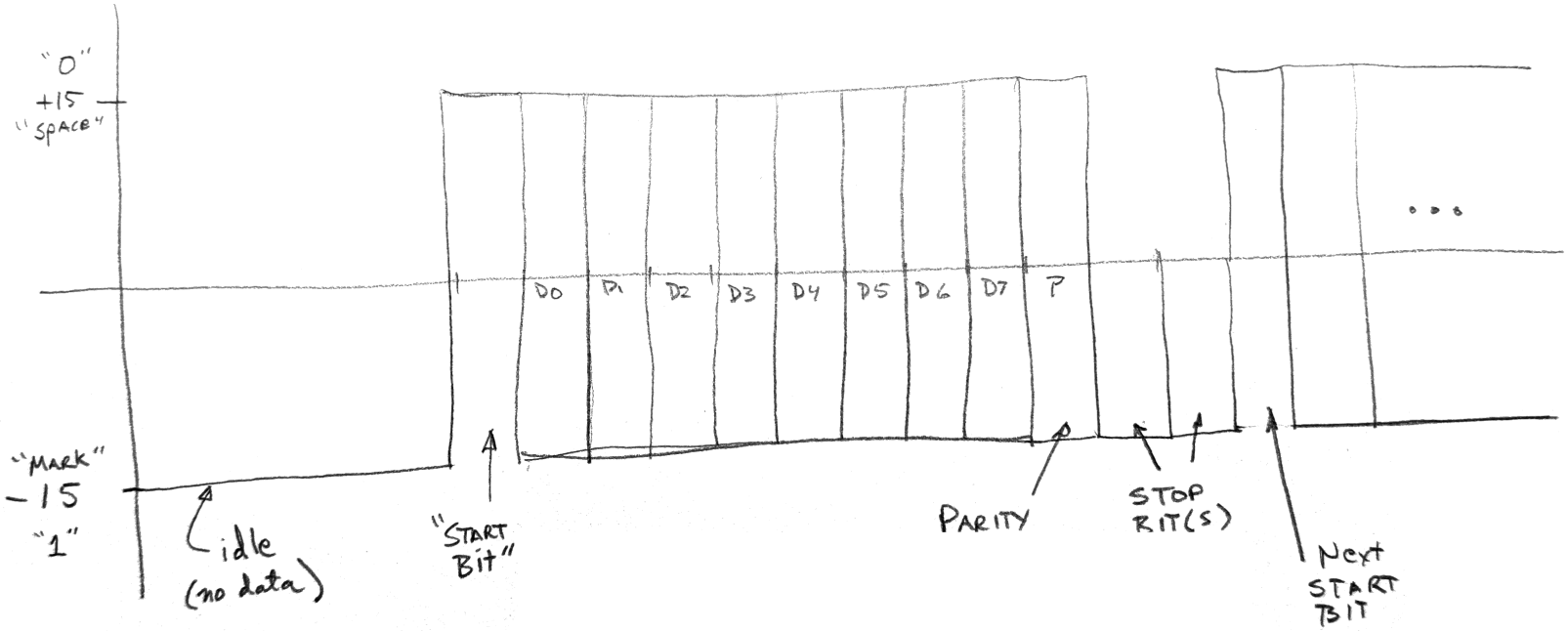
Serial Signaling

Pos.	Value	Function
0	space	Start Bit
1	X	Data 0 (lsb)
2	X	Data 1
...	X	...
8	X	Data 7 (msb)
9	X	Parity*
10	mark	Stop Bit 1
11	mark	Stop Bit 2*



Idle line in “mark” state (logic 1, -15V)
* Note: Parity and Stop Bit 2 are optional, can have 1.5 Stop Bits.

Serial Signaling



Parity

All communication systems can suffer from errors

Parity bit provides a simple error checking mechanism

Compute by taking the XOR of all data bits:

```
P = D0 ^ D1 ^ D2 ... ^ D7
```

Interpretation of XOR(bits) is
”1” if number of ones is ODD
”0” if number of ones is EVEN

Parity Eample

D0	D1	D2	XOR(D0,D1,D2)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Speed and Timing

Baud is short for Baudot, 19th century French telegrapher who invented first fixed length alphanumeric code.

Baud or **Baud rate** means number of signal changes on the line per second. Commonly equal to bits per second

Example: Arduino serial port default: 9600 Baud

```
Serial.begin(9600);      //Baud rate = 9600
Serial.begin(115200);    //Baud rate = 115200
```

Synchronization

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

`A1`: Synchronus mode — rarely used. 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.

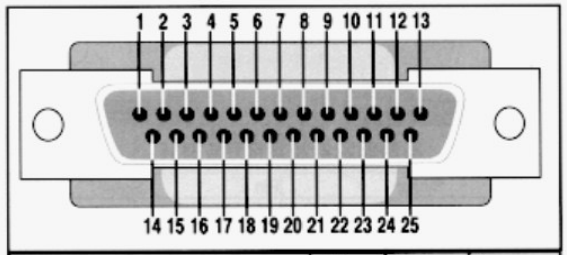
Multiplexing

Sending information in both directions. (Stringing 2 telegraph wires was very expensive!)
Terms:
Simplex One sender and one receiver.
Half Duplex One side can send at a time.
Full Duplex Both sides can send at same time. Typical serial communication today is Full Duplex.

RS232 Wiring

Minimum 3 wires:
TX (transmit)
RX (receive)
gnd (V=0 reference)

Source(https://www.aggsoft.com/rs232-pi



Pin	Description	EIA CKT	From DCE	To DCE
1	Frame Ground	AA		
2	Transmitted Data	BA		D (Data)
3	Received Data	BB	D	
4	Request to Send	CA		C (Control)
5	Clear to Send	CB	C	
6	Data Set Ready	CC	C	
7	Signal Gnd/Common Return	AB		
8	Rcvd. Line Signal Detector	CF	C	
11	Undefined			
12	Secondary Rcvd. Line Sig. Detector	SCF	C	
13	Secondary Clear to Send	SCB	C	
14	Secondary Transmitted Data	SBA		D
15	Transmitter Sig. Element Timing	DB	T (Timing)	
16	Secondary Received Data	SBB	D	
17	Receiver Sig. Element Timing	DD	T	
18	Undefined			
19	Secondary Request to Send	SCA		C
20	Data Terminal Ready	CD		C
21	Sig. Quality Detector	CG		C
22	Ring Indicator	CE	C	
23	Data Sig. Rate Selector (DCE)	CI	C	
23	Data Sig. Rate Selector (DTE)	CH		C
24	Transmitter Sig. Element Timing	DA		T
25	Undefined			

Fig.6. RS232 DB25 connector

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.

DTE, DCE are ancient names from pre-PC days!!. Typical 1990’s:
PC == DTE
modem, printer, etc, == DCE

Multi-drop (Bus)

Can also have more than 2 systems interconnected by a serial bus. In that case all users transmit and listen to same wire.
Must implement protocol to prevent two or more users from talking simultaneously. (like half-duplex)

Null modem
What if you want to connect a DTE to another DTE (such as PC-PC)?
Use a special cable which connects all lines “straight through”
except
pin 2 → pin 3 and
pin 3 → pin 2

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

Note: If you use Hardware Flow control, then “null modem” cable must also cross the RTS and DTS wires (pins 7,8 on 9-pin IBM standard).

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.

Typical 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

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.