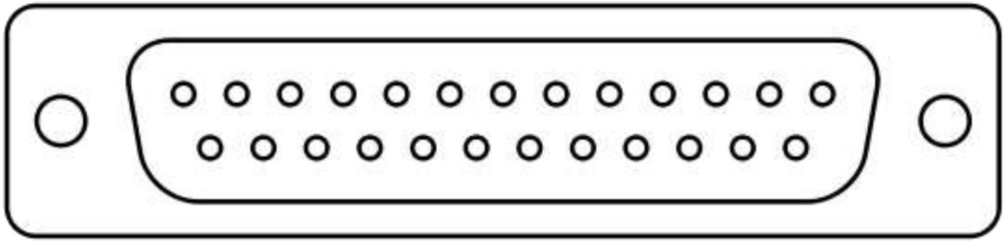
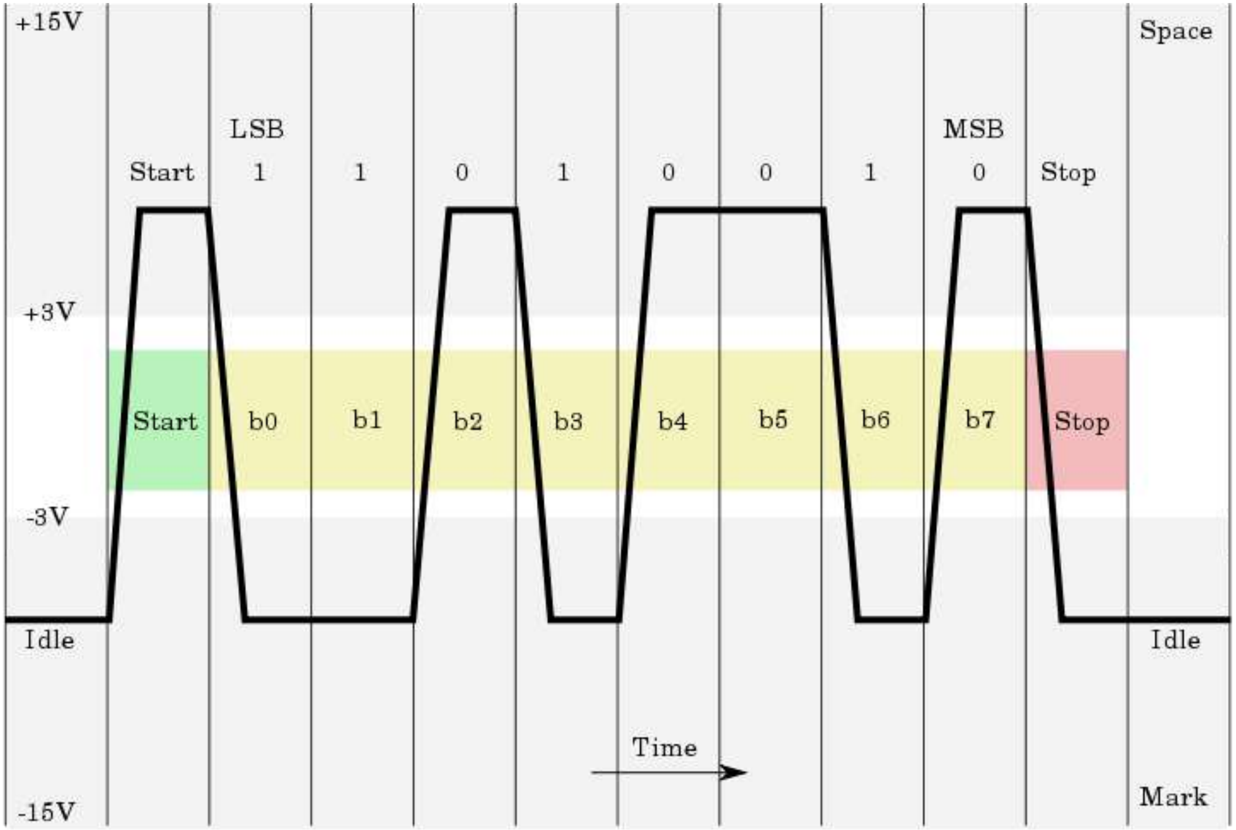
Serial communication





Overview

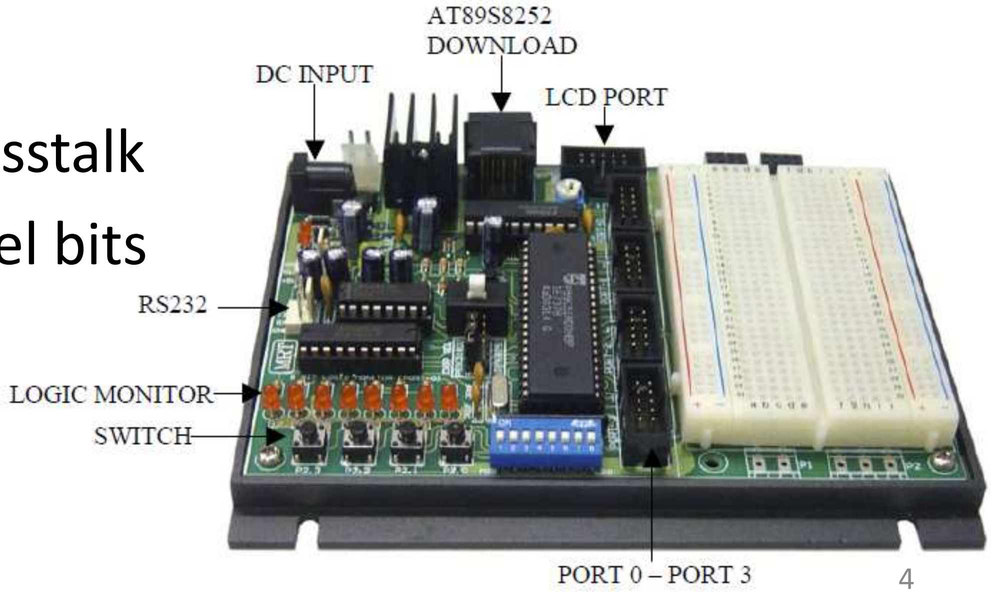
- Serial communication
  - Terminology
  - RS-232 protocol
  - Baud rates
  - Flow control
- Example
  - Develop functions to send / receive text
    - Counter (8052 pushing data stream)
    - Query/response (accept input, calculate output)
  - Talk to a VT 220 terminal using 2 wires + ground

Serial communication

- Simple way to talk over 3 wires
  - Transmit wire
  - Receive wire
  - Ground
- Why?
  - Program/upgrade an embedded device
  - Send data from sensor to a PC
  - Debug embedded device with no input/output device
    - Hook to a PC, device sends debug output to serial terminal

Parallel vs. Serial

- Parallel communication
  - Send several signal at the same time
  - One wire for each bit
- Serial communication
  - Send data one bit at a time
  - Smaller cables
  - Cheaper
  - Less opportunity for crosstalk
  - No need to synch parallel bits

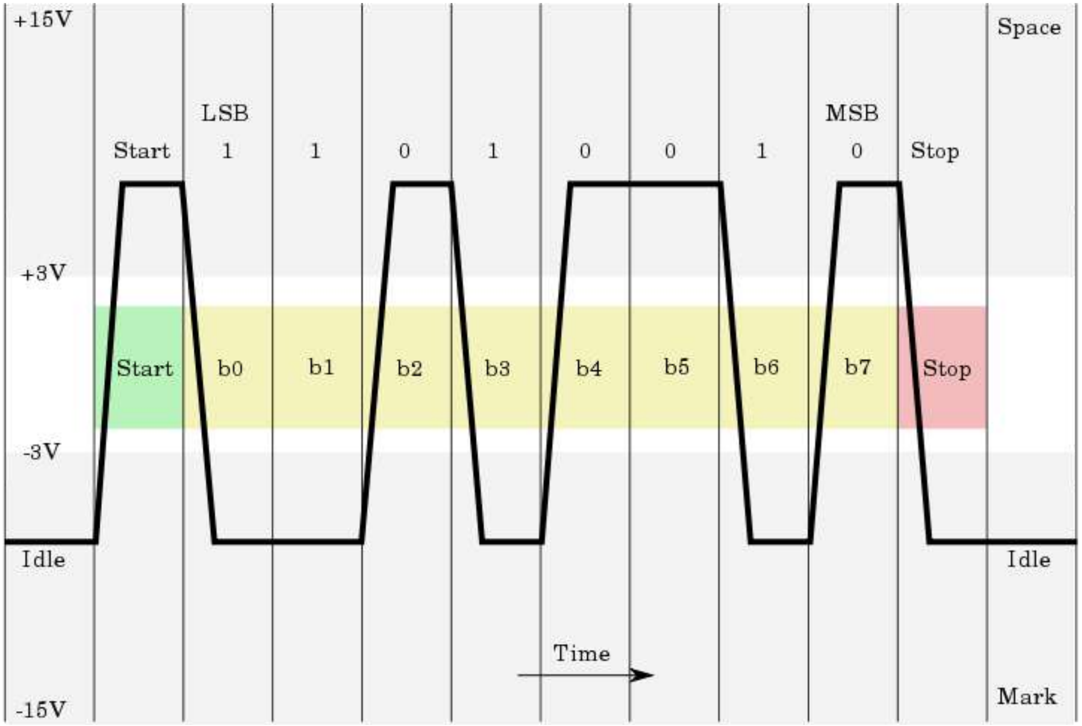


Terminology

- Full duplex
  - Can send and receive at the same time
  - Most microcontrollers with wired connections
- Half duplex
  - Cannot send and receive simultaneous
  - Wireless serial connections
- UART - Universal Asynchronous Receiver/Transmitter
  - Handles converting parallel information (a byte) into a sequences of serial information (bits)

RS-232

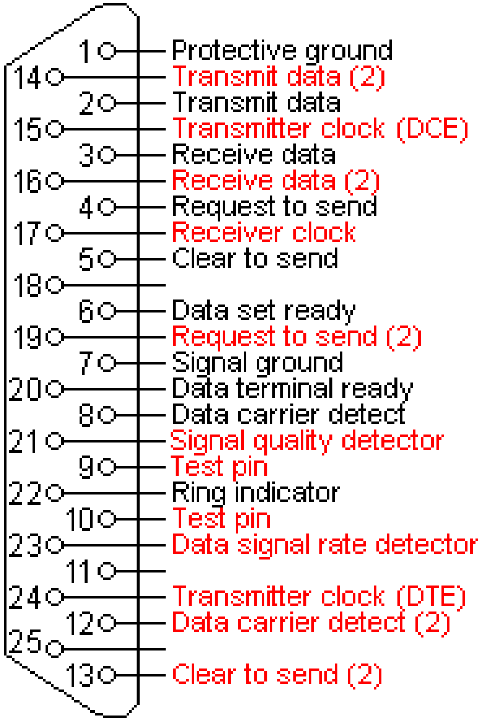
- RS-232 protocol
  - Byte-oriented protocol
    - Send a start bit, 0
    - Send the byte, one bit at a time
    - Optional parity bit
    - Send a stop bit, 1
  - No data, line at 1
  - 1-0 signals start of transmission



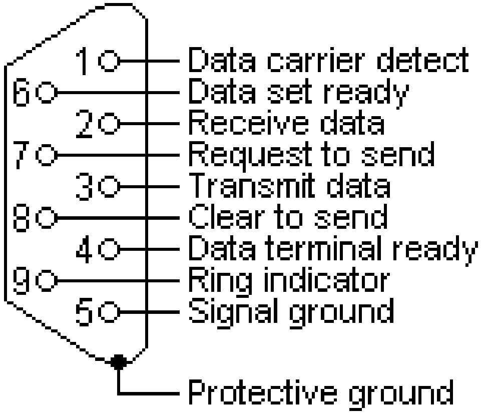
Physical connectivity

- DB25
  - 25-pin serial connector

- DB9
  - 9-pin serial connector



1 Protective ground
14 Transmit data (2)
2 Transmit data
15 Transmitter clock (DCE)
3 Receive data
16 Receive data (2)
4 Request to send
17 Receiver clock
5 Clear to send
18
6 Data set ready
19 Request to send (2)
7 Signal ground
20 Data terminal ready
8 Data carrier detect
21 Signal quality detector
9 Test pin
22 Ring indicator
10 Test pin
23 Data signal rate detector
11
24 Transmitter clock (DTE)
12 Data carrier detect (2)
25
13 Clear to send (2)



1 Data carrier detect
6 Data set ready
2 Receive data
7 Request to send
3 Transmit data
8 Clear to send
4 Data terminal ready
9 Ring indicator
5 Signal ground
Protective ground

Parity bit

- Parity bit
  - Simple form of error detection
  - Detects single bit errors
  - Add an extra bit to the 8-bits of data
    - Odd parity, count of 1's is odd
    - Even parity, count of 1's is even

Parity	Data	Data + parity
None	0100 0010	0100 0010
Odd	0100 0010	0100 0010 1
Even	0100 0010	0100 0010 0

Data transmission

- RS-232 is asynchronous
  - Bytes sent erratically in time
    - Whenever an event (e.g. key press) occurs
  - No clock signal sent with the data
    - This would require an additional wire
  - Both sides have an internal clock
    - Running at same rate
  - Clocks synched on transmission/reception of start bit
    - Sender clocks out byte one bit at a time
    - Receiver clocks in byte one bit at a time

Baud rate

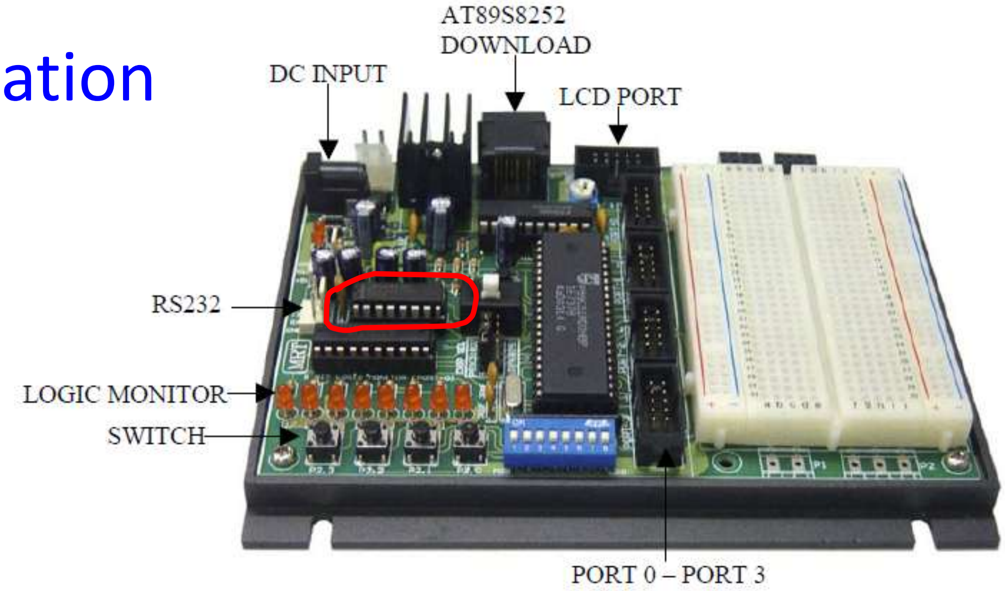
- How fast can we talk?
  - Sender and receiver agree on a clock rate
  - Baud rate
    - Bits per second
    - RS-232 One of a restricted set
      - 75, 110, 300, 1200, 2400, 4800,9600, 14400, 19200, 28800, 33600, 56000, 115000

Flow control

- Flow control
  - Preventing one side from overwhelming the other
    - e.g. Limited buffer space, slow printer
  - Hardware based, add extra wires
    - RTS (Request to Send)
    - CST (Clear to Send)
  - Software based
    - In-band signaling on the transmit line
    - XOFF (transmit off) - ctrl+s, ASCII 19
    - XON (transmit on) - ctrl+q, ASCII 17

8052 UART

- 8052 serial communication
  - 3 wire connector
  - 4 different modes
    - 1 synchronous
      - half-duplex
    - 3 asynchronous
      - full-duplex
  - Baud rate controlled by:
    - Timer1/2 overflow
    - Fixed to oscillator frequency



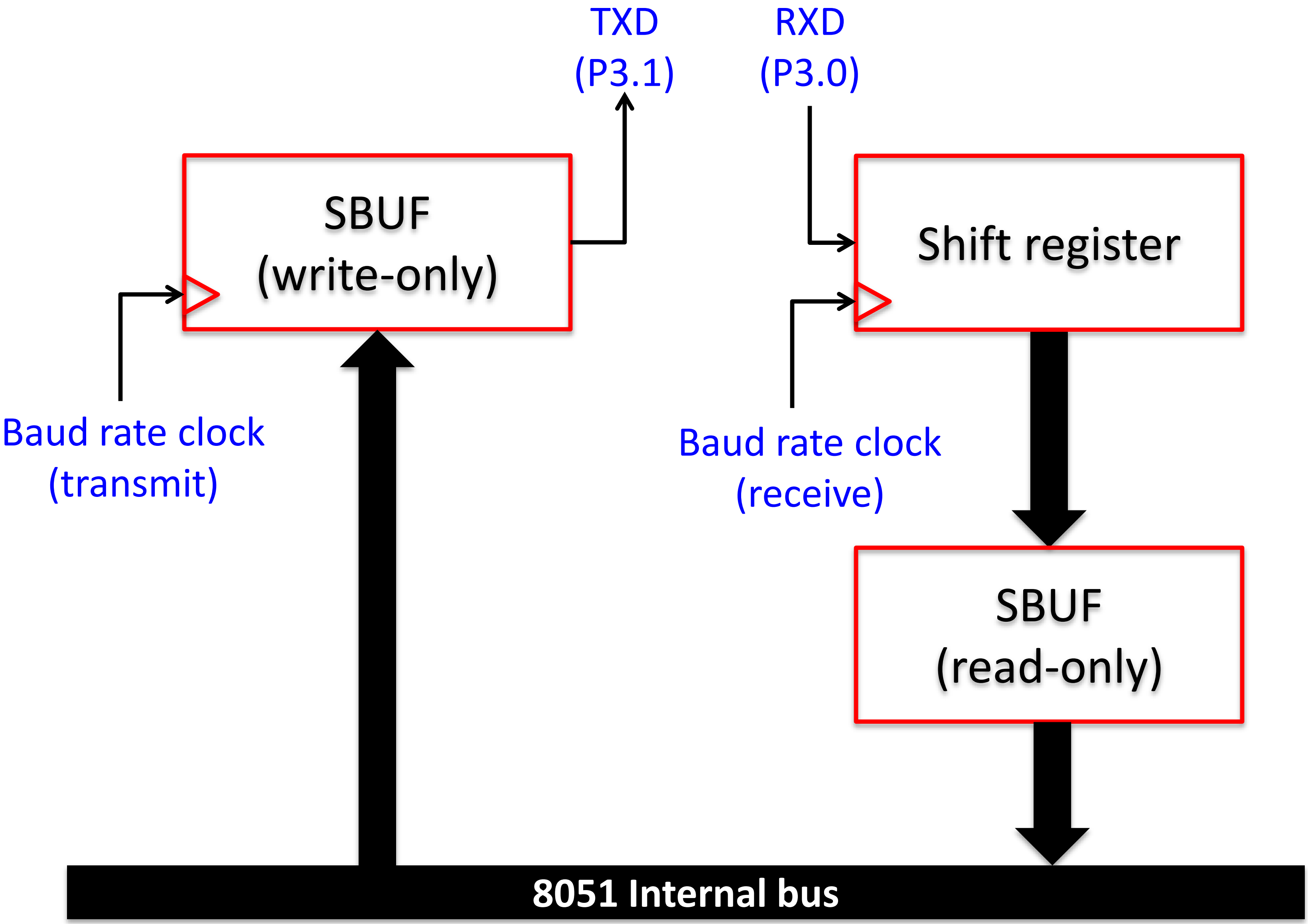
Serial control SFRs

- SCON
  - Controls the mode (SM0, SM1)
  - Flags for when byte transmitted (TI) or received (RI)
  - Bit for sending or receiving parity

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Name	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

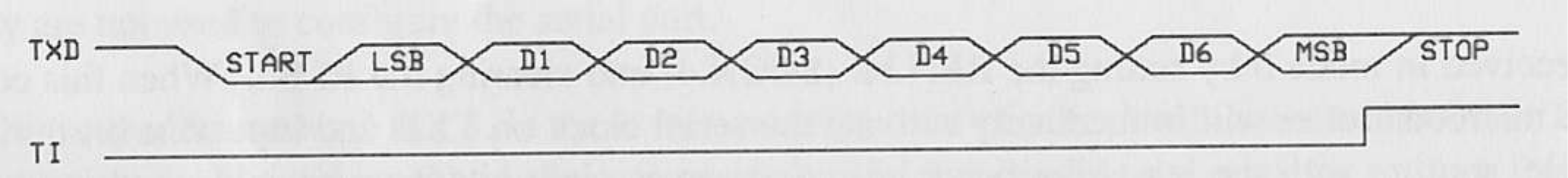
- SBUF
  - Holds byte result of send
  - Holds byte result of receive

Serial port buffer (SBUF)



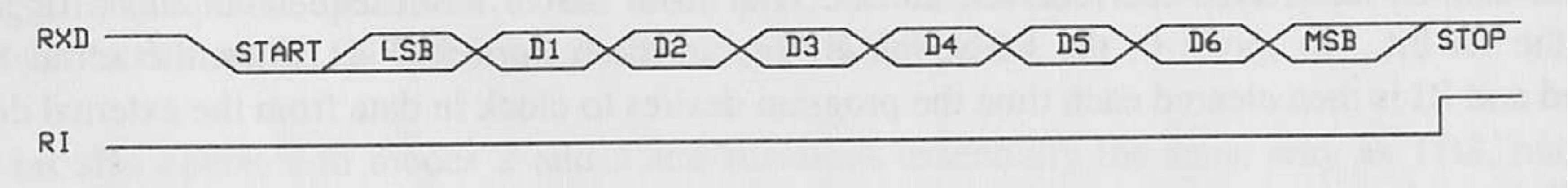
Serial mode 1

- 8-bit UART
  - Most common 8052 serial mode
  - SM0 = 0, SM1 = 1
  - Variable baud rate based on timer overflow
- Transmit:
  - Put a byte in SBUF SFR
  - Ten bits sent = start bit (0) + 8 data bits + stop bit (1)
  - TI bit set when last bit sent



Serial mode 1

- Receive:
  - Recognize incoming start bit on RXD
    - Stop bit (1), start bit (0)
    - Wait for 1-0 transition
  - Ten bits received = start bit (0) + 8 data bits + stop bit (1)
  - RI bit set when last bit received



Mode 1 procedure

- Step 1: Set mode bits in SCON
- Step 2: Start timer1 based on baud rate
- Step 3: Read/write byte
- Step 4: Wait for RI/TI bit to be set

```
SCON = 0x50;  // 8-bit UART, timer1 as baud rate generator
TMOD = 0x20;  // Set timer1 to 8-bit auto-reload
TH1  = 253;   // Reload value of 253 (9600 baud)
TR1  = 1;     // Start timer1

while (1)
{
   while (!RI) {}  // Wait to receive a byte
   ch = SBUF;      // Read character from serial port
   RI = 0;         // Clear the receive flag
   TI = 0;         // Clear the transmit flag
   SBUF = ch;      // Echo the character back to serial port
   while (!TI) {}  // Wait for byte to be transmitted
}
```

Timer setup

- How to determine reload value for timer?
  - Timer1 8-bit auto-reload
  - Set TH1, high byte of reload value
    - The closer to 0xFF, the faster the baud rate
  - Optionally set SMOD bit to double baud rate

$$baud\ rate = \frac{2^{SMOD} \cdot Frequencyo_{scillator}}{32 \cdot Instructionsc_{ycle} \cdot (256 \ - TH1)}$$

$$9600 = \frac{2^{0} \cdot 11059200}{32 \cdot 12 \cdot (256 \ - TH1)}$$

$$TH1 = 3$$

Why 11.0592 Mhz?

- Why the strange 8052 clock frequency
  - Using TH1 and SMOD, can obtain exact baud rates
  - Need to stay ±2.5% of ideal

$$baud\ rate = \frac{2^{SMOD} \cdot Frequencyo_{scillator}}{32 \cdot Instructionsc_{ycle} \cdot (256 \ - TH1)}$$

$$9600 = \frac{2^0 \cdot 11059200}{32 \cdot 12 \cdot (256 \ - TH1)}$$

$$TH1 = 3$$

Exactly 9600 baud

$$9600 = \frac{2^0 \cdot 12000000}{32 \cdot 12 \cdot (256 \ - TH1)}$$

$$TH1 = 3.255208333333$$

About 10417 baud

Sample reload values

Baud rate	Clock frequency	SMOD	TH1 reload value	Actual baud rate	Error
9600	12 Mhz	1	-7 (F9h)	8923	7%
2400	12 Mhz	0	-13 (F3h)	2404	0.16%
1200	12 Mhz	0	-26 (E6h)	1202	0.16%
19200	11.0592 Mhz	1	-3 (FDh)	19200	0%
9600	11.0592 Mhz	0	-3 (F4h)	9600	0%
2400	11.0592 Mhz	0	-12 (F4h)	2400	0%
1200	11.0592 Mhz	0	-24 (E8h)	1200	0%

Summary

- Serial communication
  - Communication over a few wires
  - RS-232 protocol
    - Start, stop, parity bit
    - Flow control
  - UART
    - Converts parallel data (byte) to serial data (bit signals on a wire)
  - 8052 serial communication
    - Setup SCON and timer
      - Timer1 auto-reload
      - Many exact baud rates possible with 11.0592 Mhz crystal
    - Send / receive using SBUF
    - TI / RI flags when byte done sent / received