NMK322 - Microcontroller

Lecture 06 – 8051 Serial Interface

Serial Communications

- Parallel vs serial
  - Why serial?
- Communication modes
  - Signal ‘directions’
- Synchronous/Asynchronous
  - Timing options
- Transfer rate / signaling protocol
  - Switching speed
  - Framing & electrical specifications
- For NMK322, 2 words: UART & RS232

Why serial?

- Disadvantage
  - less data per clock cycle
- Advantage
  - higher clock rate
    - less space → better isolation
    - less ‘crosstalk’ issue ← less capacitive effect?
  - lower cost
    - less wires
    - less pins/ports
- Your pick?

Communication Mode

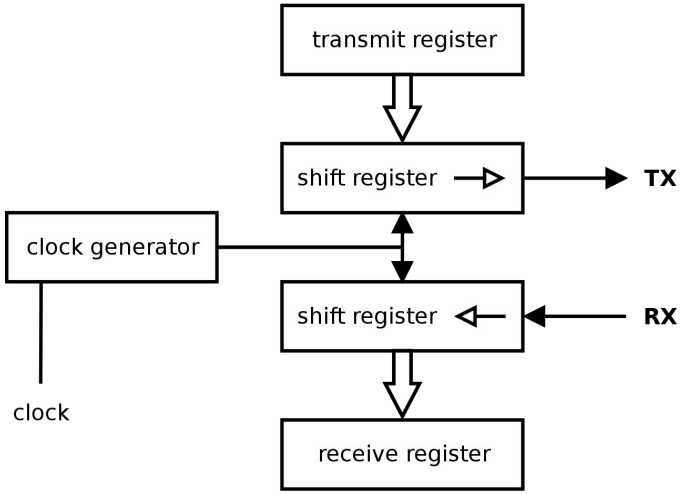
- Simplex
  - single transmission line , single direction
  - device transmit OR receive ONLY
- Half Duplex
  - single transmission line , both directions
  - taking turns transmitting and receiving
- Full Duplex
  - dual transmission line , both directions
  - device can transmit AND receive simultaneously

{S,As}ynchronous Transfer

- Synchronous
  - clock signal is part of the interface
  - single time reference for both devices
  - usually higher transmission rate
- Asynchronous
  - clock signal NOT in the interface
  - both devices need ‘agree’ on a clock rate
  - need a method to keep in-phase

Asynchronous Serial Communication

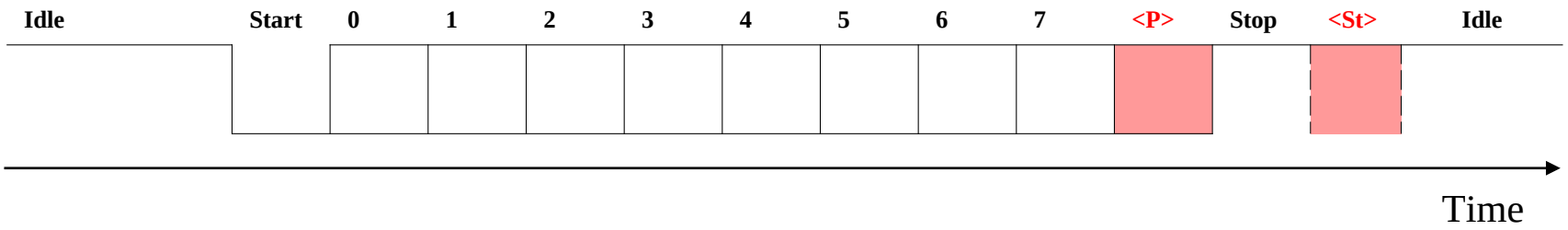
- Universal Asynchronous Receiver-Transmitter (UART)
  - generic hardware design consensus
  - parallel ↔ serial logic conversion
  - configurable clock generator
  - dual lines: transmit (TX) and receive (RX)



UART: Data Framing

- 2 issues in asynchronous serial transmission
  - keep device clock in-phase
  - know when data begins/ends
- needs meta-signal → frame!
  - Idle [1] (logic high @mark)
  - Start [1] (logic low @space)
  - Data [5-9], Parity [0-1]
  - Stop [1-2] (logic high)

Note: Logic high can be most +ve (e.g. 5V) OR most -ve (e.g. 0V)



Note: **mark** and **space** are terms used in telecommunications. Sometimes, still used to explain serial communication.

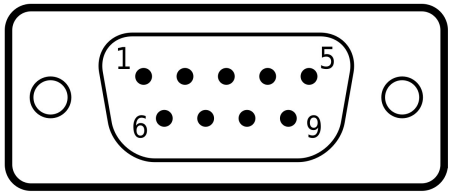
UART: Signal Protocol and Timing

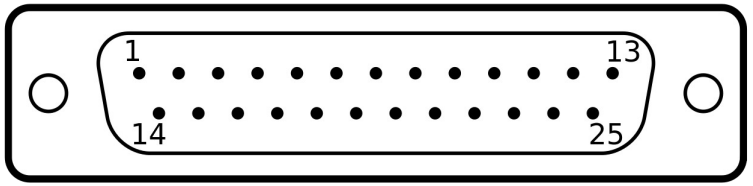
- RS232 – Recommended Standard 232
  - in telecommunications, for serial communications
  - defines signals between DCE and DTE
  - commonly used protocol in microcontroller systems
- RS232 specifies (among others):
  - electrical signal characteristics (logic level, timing, etc.)
  - mechanical characteristics (connectors, pin id, etc.)

Note: **DCE** is **D**ata **C**ommunication (@Circuit-terminating) **E**quipment (like a modem) and **DTE** is Data Terminal Equipment (like a computer).

RS232 Interface

- Physical Port
  - DB9 (most common) or DB25
  - no longer a ‘standard’ port on modern PC





- Virtual Port
  - USB virtualization → USB-to-serial converters
  - Utilizes 0V – 5V range instead

RS232 Signals

- Line count:
  - defines many signals (using up to 25-pin connector)
  - only 3 core signals: TX (Data Transmit), RX (Data Receive), GND (Voltage reference)
- Voltage level:
  - logic 1 (@mark) at -15V to -3V range
  - logic 0 (@space) at 3V to 15V range
  - not TTL → require line driver like MAX232

8051 Serial Port

- Full-duplex serial port
  - RX @P3.0 , TX @P3.1
- Variable baudrate controlled using Timer 1
  - using 8-bit auto-reload (mode 2)
- Port settings in SCON register (@0x98)
- Serial data buffer in SBUF register (@0x99)
  - 2 physical registers (TX/RX) using same address
  - TX buffer will be transmitted when written to
  - RX buffer will be written when a byte is fully received (double buffer)

Serial Control Register (SCON)

MSB LSB

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Bit	Name	Description
SCON.7	SM0	Serial Port Mode bit 0
SCON.6	SM1	Serial Port Mode bit 1
SCON.5	SM2	Multiprocessor Communication Enable
SCON.4	REN	Receive Enable Set to enable reception. CLR to disable reception.
SCON.3	TB8	Bit 8 of message to transmit Used to transmit optional parity bit
SCON.2	RB8	Bit 8 of received message Receives optional parity bit
SCON.1	TI	Transmit Interrupt Flag Set when Byte in SBUF is completely transmitted.
SCON.0	RI	Receive Interrupt Flag Set when a valid byte is received into SBUF

8051 Serial: Mode 1

- 8-bit UART, variable baud rate
  - most commonly used (RS232)
- 10-bits transmission for both TX / RX
  - Start-bit, Data-bit (x8), Stop-bit
- Transmit (TX)
  - starts when SBUF(TX) is written to
  - TI flag asserted when Stop-bit is on TX line
- Receive (RX)
  - can only happen if REN = 1 (in SCON)
  - RI flag asserted when data copied into SBUF

Baudrate Calculation

- Timer 1 in Mode 2 (overflow rate)
  - TF1 will trigger next bit transfer
  - baudrate calculation:

$$Baud = \frac{2^{SMOD}}{32} \times \frac{F_{osc}}{12} \times \frac{1}{(256 - TH1)}$$

  - thus, TH1 reload value

$$TH1 = 256 - \left(\frac{2^{SMOD}}{32} \times \frac{F_{osc}}{12} \times \frac{1}{Baud} \right)$$

SMOD is MSB in PCON (SFR@0x87)

  - for 11.0592MHz, TH1 = 256 – (28800/Baud)
    - Note: 11.0592MHz/12/32 = 28800 (SMOD=0)

Calc1: 9600 baud

- Assume f_{osc} = 11.0592MHz

$$TH1 = 256 - (28800/9600)$$
$$= 256 - (288/96)$$
$$= 256 - 3$$
$$= 253$$

8051 Serial: UART RS232

- Configure serial control for mode 1
  - 8-bit, var. baud
- Configure timer 1 as baud rate generato
  - 8-bit auto-reload
- Set timer reload value
  - e.g. 253 for 9600
- Start timer @ baud generator

```
SCON = 0x50;

TMOD &= 0x0F;
TMOD |= 0x20;

TH1 = 253;
TR1 = 1;
```

8051 Serial: Transmit/Receive 1 byte

- Send byte to SBUF
  - assume value 0x55
- Wait for transmit to finish
  - check TI flag
- Reset for next
  - Clear TI flag

```
SBUF = 0x55;
while (TI==0);
TI = 0;
```

- Wait for receive to complete
  - check RI flag
- Read byte from SBUF
  - assume var ‘sdat’
- Reset for next
  - Clear RI flag

```
while (RI==0);
sdat = SBUF;
RI = 0;
```

8051 Code: Continuous TX/RX

- Code to continuously send ASCII ‘A’ @9600

```
#include <reg51.h>
void main(void) {
  SCON = 0x50;
  TMOD = 0x21;
  TH1 = 253; TR1 = 1;
  while (1) {
    SBUF = ‘A’; //0x41
    while (TI==0);
    TI = 0;
  }
}
```

- Code to continuous read serial and pass to P1 @9600

```
#include <reg51.h>
void main(void) {
  SCON = 0x50;
  TMOD = 0x21;
  TH1 = 253; TR1 = 1;
  while (1) {
    while (RI==0);
    P1 = SBUF;
    RI = 0;
  }
}
```

Additional Serial Modes

(not really used in the lab)

8051 Serial: Mode 2

- 9-bit UART, fixed baud rate
  - allows parity bit inclusion
- 11-bits transmission for both TX / RX
  - Start-bit, Data-bit (x8), TB9/RB9, Stop-bit
- Baud rate at 1/32 OR 1/64 system frequency (f_{osc})
  - Internal phase 2 clock
  - SMOD=0, Baud=f_{osc}/64
  - SMOD=1, Baud=f_{osc}/32

8051 Serial: Mode 3

- 9-bit UART, variable baud rate
  - combination of modes 1 & 2
- 11-bits transmission for both TX / RX
  - Start-bit, Data-bit (x8), TB9/RB9, Stop-bit

8051 Serial: Mode 0

- Half-duplex synchronous transfer
  - RX as data , TX as clock
- 8-bit without frame
  - LSB first
- Data rate at 1/12 system frequency (f_{osc})
  - fixed rate, machine cycle frequency
- Transfer
  - single negative pulse (shift clock) per bit
  - TI/RI set after 8th shift clock

End of Lecture06