# SERIAL COMMUNICATION

## EE3232 DIGITAL SYSTEMS III
## CLASS NOTES CHAPTER 6

**Department of Electrical Engineering**
**University of New Brunswick**

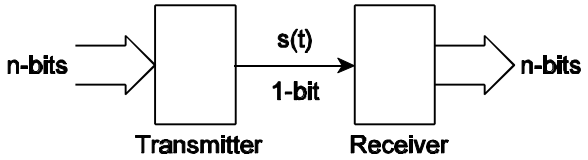
 **January 5, 2000**

## SUMMARY


- Objectives.
- Introduction.
- Asynchronous serial communication.
- Synchronous serial communication.
- UART's.
- The internal architecture of a UART.
- Interfacing the TL16C452 to a system bus.
- Programmers model for the TL16C452.
- Programming the TL16C452.
- Serial data communication standards.
- Examples of serial communication standards.
- Line drivers and receivers.
- The RS-232C standard.
- 20 mA current loops.
- MODEMS.


## OBJECTIVES

- To sketch the logic signal versus time for a character transmitted using serial transmission.
- Given the logic signal versus time, recover the transmitted characters and any receive errors.
- List the differences between synchronous and asynchronous transmission.
- To draw a block diagram of the internal architecture of a UART and concisely explain its operation.
- To show the H/W and S/W interface between a UART, a system bus and a communication line (such as RS-232C).




## INTRODUCTION

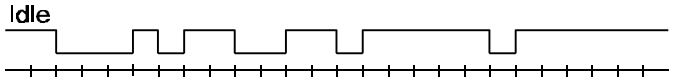
- ***Serial communication*** : the transfer of digital data over a single data transmission line (normally) one bit at a time.
- Why?
  - Over long distances it costs less.
  - Existing telecommunication networks may be used for transmission.
- A communication channel may be classified as :
  - ***Simplex*** : one-way communication,
  - ***Half duplex*** : two-way, one-direction at a time,
  - ***Full duplex*** : two-way simultaneous communication.
- ***Baud Rate*** : is defined as the rate at which serial data is transferred over a channel.

$$\text{Baud Rate} = \frac{1}{\text{Time for one bit cell}}$$

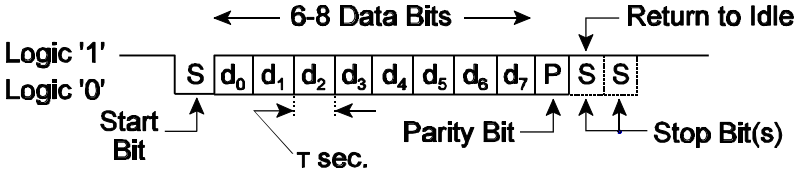
- Common Baud rates include: 110 Bd, 300 Bd, 1200 Bd, 2400 Bd, 4800 Bd, 9600 Bd, 19200 Bd, 56800 Bd.
- Serial data transfer may be classified as,
  1. ***Asynchronous*** : a character is sent whenever ready.
  2. ***Synchronous*** : a block of characters is sent, one immediately following another.

## ASYNCHRONOUS SERIAL COMMUNICATION

- A character (6-8 bits) is sent whenever available.



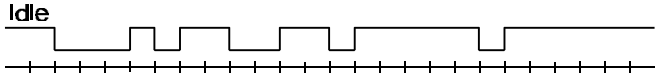

- Asynchronous serial data format,
  - Baud rate = 1/τ
  - Data bits are usually encoded in ASCII or EBCDIC.
  - The least significant bit, $d_0$, is sent first.




- Serial data receive errors:

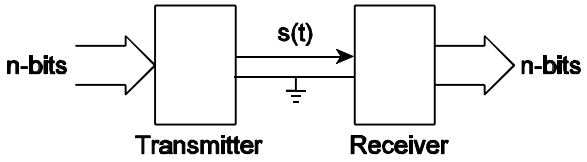
  i. ***Framing*** : An incorrect stop bit is received.
  ii. ***Overrun*** : The receiver receives new data before the previous data has been processed.
  iii. ***Parity*** : An incorrect parity bit is received.

- Parity is said to be ***even*** if the number of '1' s including the parity bit is even, otherwise parity is said to be ***odd***.




## SYNCHRONOUS SERIAL COMMUNICATION

- Characters are sent one following the other as a ***block***.
- No stop / start bits are appended.




## SYNCHRONOUS SERIAL COMMUNICATION (CONT'D)

- Each block begins with one or more synchronization characters and ends with an end of frame character.

| SYNC | SYNC | STX | Char | Char | Char | Char | ETX | BCC | PAD |
|---|---|---|---|---|---|---|---|---|---|

- The transmitter automatically inserts SYNC or NUL characters if information is not received fast enough.
- This ensures synchronization between the Tx and Rc.

## SYNCHRONOUS SERIAL COMMUNICATION (CONT'D)

- Receiver initially operates in hunt mode, checking the serial data stream until a SYNC character(s) is detected.
- Once a SYNC character is detected the receiver synchronizes reading of the entire block of serial data.
- An example : BISYNC.
- In addition to SYNC and NUL characters there are additional special characters : STX, ETX, BCC, PAD.
- For large bocks of data, synchronous communication has less ***overhead*** than asynchronous communication.
- The ratio, η, is less when compared to asynchronous communication for a large number of characters.

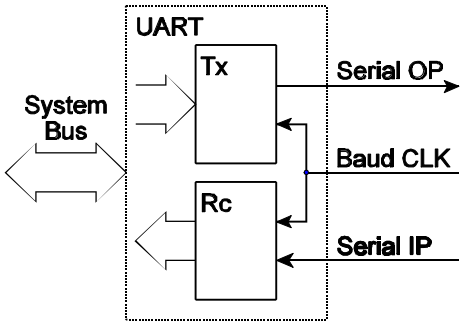
$$\eta = \frac{\text{Total number of information bits}}{\text{Total number of bits}}$$

## UART'S

- ***UART*** : **U**niversal **A**synchronous **R**eceiver **T**ransmitter.
- A peripheral dedicated for asynchronous serial transfer.
- A UART contains two distinct circuits:
  - A ***Transmitter or Tx*** circuit and

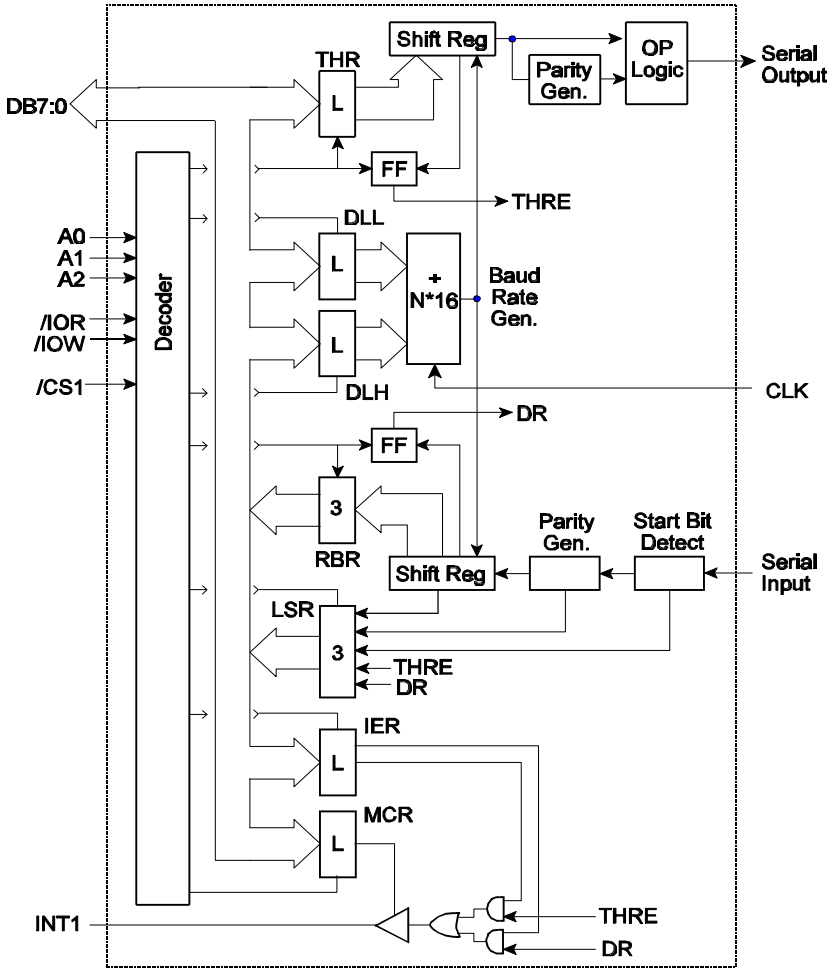
- A ***Receiver or RC*** circuit, ∴ full duplex is supported.

## UART'S (CONT'D)

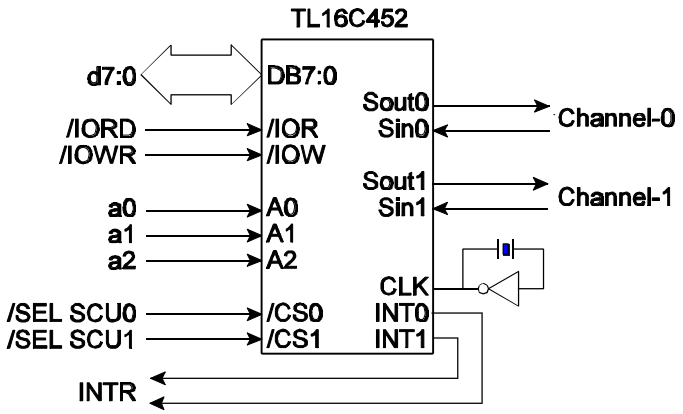



- ***Transmitter Operation :***
  - An n-bit character is accepted from the CPU using parallel handshaking.
  - Start, Parity and Stop bits are inserted.
  - The character is shifted ***out*** at the ***Serial OP*** one bit at a time synchronized to the serial CLK.
- ***Receiver Operation :***
  - Wait until a start bit is detected.
  - Synchronize reading of the ***Serial IP*** at the center of the bit cell.
  - Shift ***in*** the character at the ***Serial IP*** one bit at a time synchronized to the serial CLK.
  - Check for receive errors : parity, framing, overrun.
  - Strip off the Start, Parity and Stop bits.
  - Assemble the data bits into a single n-bit character.
  - The n-bit character is read by the CPU using parallel handshaking.
- Certain aspects of UART operation are programmable,
  - Character size,
  - Parity : even or odd,
  - Number of stop bits and
  - Baud rate.
- The UART status at a particular instant may be read,
  - ***THRE*** : transmit holding register empty, the transmitter is ready to accept a char from the CPU,
  - ***DR*** : Data ready, the receiver has a character available for the CPU,
  - ***FE*** : framing error detected at receiver,
  - ***OE*** : overrun error detected at receiver,
  - ***PE*** : parity error detected at receiver.

## THE INTERNAL ARCHITECTURE OF A UART



## INTERFACING THE TL16C452 TO A SYSTEM BUS




## PROGRAMMERS MODEL FOR THE TL16C452

- The TL16C452 contains two built-in 8250 type UART's.
- The TL16C452 read-write operation table.

| DLAB | $A_2A_1A_0$ | $/CS_1 = 0$ | $/CS_0 = 0$ | Register |
|---|---|---|---|---|
| 0 | 000 | RBR1 | RBR0 | Receive Buffer Register |
| 0 | 000 | THR1 | THR0 | Transmit Holding Register |
| 0 | 001 | IER1 | IER0 | Interrupt Enable Register |
| x | 010 | IIR1 | IIR0 | Interrupt Identification Reg |
| x | 011 | LCR1 | LCR0 | Line Control Register |
| x | 100 | MCR1 | MCR0 | Modem Control Register |
| x | 101 | LSR1 | LSR0 | Line Status Register |
| x | 110 | MSR1 | MSR0 | Modem Status Register |
| x | 111 | SCR1 | SCR0 | Scratch Pad Register |
| 1 | 000 | DLL1 | DLL0 | Divisor Latch Lo Byte |
| 1 | 001 | DLM1 | DLM0 | Divisor Latch Hi Byte |

## PROGRAMMING THE TL16C452 UART'S

- Refer to the data sheets for the TL16C452 in your Lab Manual.
- Initializing a TL16C452 UART :
  - Write '1' to bit-7 of the LCR to access the DLL / DLM.
  - Write word, N, to DLL and DLM to program the Baud Rate.
  - Baud Rate = CLK ÷ (N * 16)
  - Write to the LCR with '0' in bit-7 to program: word length, number of stop bits, parity enable, even / odd parity with no break.

- Reading a character from a TL16C452 UART using polled handshaking:
  - Poll bit-0, DR, of the LSR until '1'.
  - Read received character at RBR.

- Writing a character to a TL16C452 UART using polled handshaking:
  - Poll bit-5, THRE, of the LSR until '1'.
  - Write character to be transmitted to THR.

- For interrupt driven handshaking,

  - Write IER to enable specific combinations of interrupt sources such as : DR, THRE, PE, OE, FE.
  - Write '1/0' to bit-3 of MCR to enable interrupt generation by the TL16C452.

- Inside the interrupt handler,

  - Read IIR to identify the interrupt source (within the TL16C452) that caused the interrupt.
  - Branch control to an appropriate routine that services the interrupt.

## SERIAL DATA COMMUNICATIONS STANDARDS

- ***Data Communications*** : the ability to exchange information intelligibly.
- If everyone adheres to the same standard (or set of rules) then everyone can communicate with each other.
- Examples of organizations that develop standards: EIA, IEEE and CCITT.

- Data communications standards specifications include:

    1. ***The Physical Layer*** : Defines the mechanical connectors, pin assignment, signal definitions and electrical characteristics, and
    2. ***The Data Link Layer*** : Defines the protocol for data exchange, handshaking, etc. ...
    3. Other higher level layers ...

## EXAMPLES OF SERIAL COMMUNICATION STANDARDS

| STD | Length | Data Rates | Logic '0' | Logic '1' |
|---|---|---|---|---|
| RS-232C | < 20 m | 20Kbps/20m | 3V to 25V @ Rc<br>5V to 15V @ Tx | -3V to -25V @ Rc<br>-5V to -15V @ Tx |
| RS-423A | < 1500 m | 100Kbs/10m | 0.2V to 6V @ Rc<br>3.6V to 6V @ Tx | -0.2V to -6V @ Rc<br>-3.6V to -6V @ Tx |
| RS422A | < 1500 m | 100Kbs/1000m | 0.2V to 6V @ Rc<br>2V to 6V @ Tx | -0.2V to -6V @ Rc<br>-2V to -6V @ Tx |

## LINE DRIVERS AND LINE RECEIVERS

- ***Line Driver*** : Converts TTL voltage levels into transmission line voltages, e.g., the MC1488 converts TTL voltages into RS-232C voltage levels.

- ***Line Receiver*** : Converts transmission line voltages into TTL voltage levels, eg., the MC1489 converts RS-232C voltage levels into TTL voltage levels.

## THE RS-232C STANDARD

- The RS-232C standard is the most common serial communication standard.
- Originated as an interface between a TERMINAL and a MODEM, (modulator/demodulator).
- RS-232C was defined in 1962, (before microcomputers).
- 25-pins are defined. Only a few are used in practice,

## THE RS-232C STANDARD (CONT'D)

| Pin | Tag | Function |
|---|---|---|
| 2 | TX | Transmitted data. |
| 3 | RC | Received data. |
| 7 | GND | Signal ground. |
| 20 | DTR | Data terminal ready: dial up another modem. |
| 6 | DSR | Data set ready: acknowledge from the modem. |
| 8 | DCD | Data carrier detect: a distant call is connected. |
| 4 | RTS | Request to send is a request to send data. |
| 5 | CTS | Clear to send is an acknowledge that data may be sent. |

## 20 MA CURRENT LOOPS

- In many industrial applications serial communication is implemented using 20 mA current loops.

- Logic levels are represented by the presence or absence of a certain ***current range*** rather than a ***voltage range***.

- Why? A current signal is less susceptible to electromagnetic interference than a voltage signal.

## MODEMS

- A modem is an acronym for ***MO***dulator - ***DEM***odulator. A modem is used for two functions:

    - Transforming serial data (that is normally at TTL voltage levels) into signals suitable for driving analog telephone lines.
    - Transforming signals recovered from analog telephone line into serial data (at TTL voltage levels).