Interfaz Serie

M. En C. Victor Hugo García Ortega

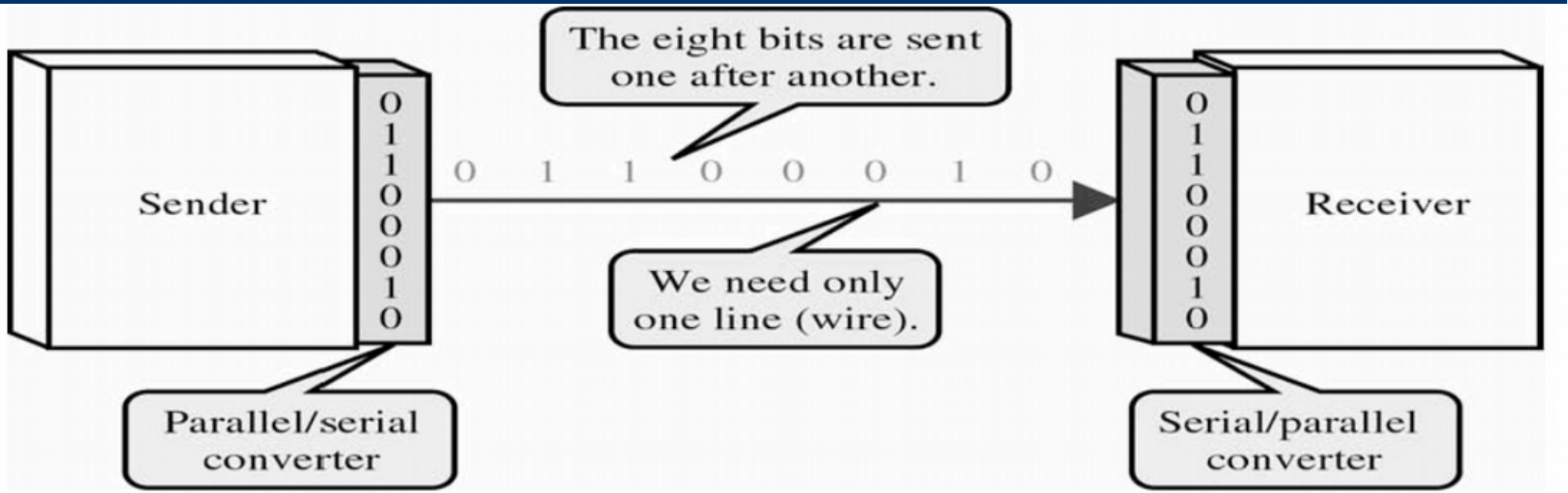
Escuela Superior de Cómputo – IPN
Av. Juan de Dios Batiz s/n
Unidad Profesional Zacatenco
07738, México, D.F.
vgarciaortega@yahoo.com.mx, vgarciao@ipn.mx

Comunicación serie

La comunicación serie es una interfaz de transmisión de datos digitales, frecuentemente utilizado por procesadores y periféricos, donde la información es transmitida bit a bit, enviando un solo bit a la vez, es decir, secuencialmente.

Comunicación serie

Dentro de los sistemas basados en procesadores, la comunicación suele llevarse a cabo en paralelo. Por lo tanto, se requieren dispositivos que conviertan los datos de paralelo a serie y viceversa.



Interfaces para comunicación serie

Existen varias implementaciones de estándares e interfaces para realizar comunicación de datos en forma serial. Algunas de ellas son:

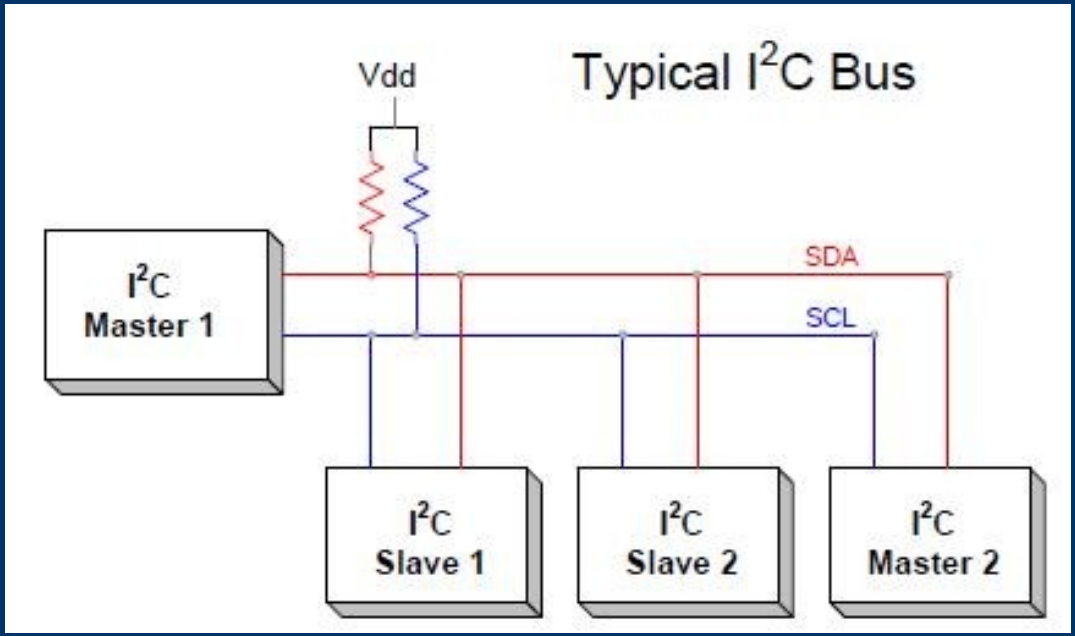
SCI – Serial Communications Interface (UART)
SPI – Serial Peripheral Interface
USB – Universal Serial Bus
IIC – Inter Integrated Circuit
CAN – Controller Area Network
IIS – Integrated Interchip Sound

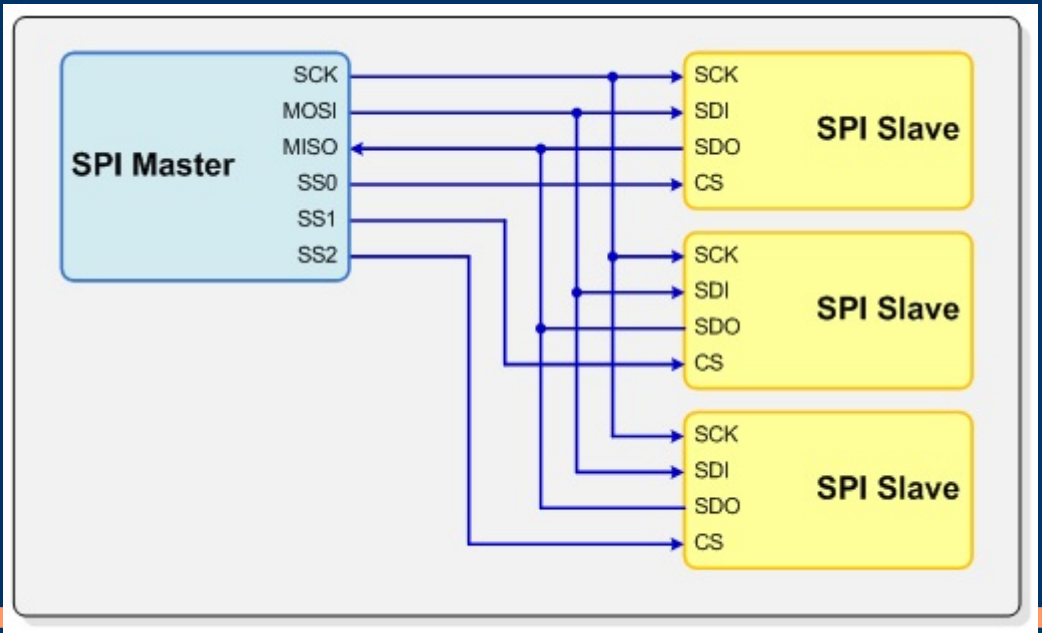
Comunicación síncrona

Estas implementaciones realizan la comunicación serie de forma síncrona o asíncrona.

En la transmisión de datos síncronos cada uno de los bits que compone la información se ordena en un paquete sin espacios entre ellos. Para conseguir el sincronismo se comparte una señal de reloj entre los dos dispositivos a comunicar. Las interfaces SPI, IIC, IIS son síncronas.

Comunicación síncrona

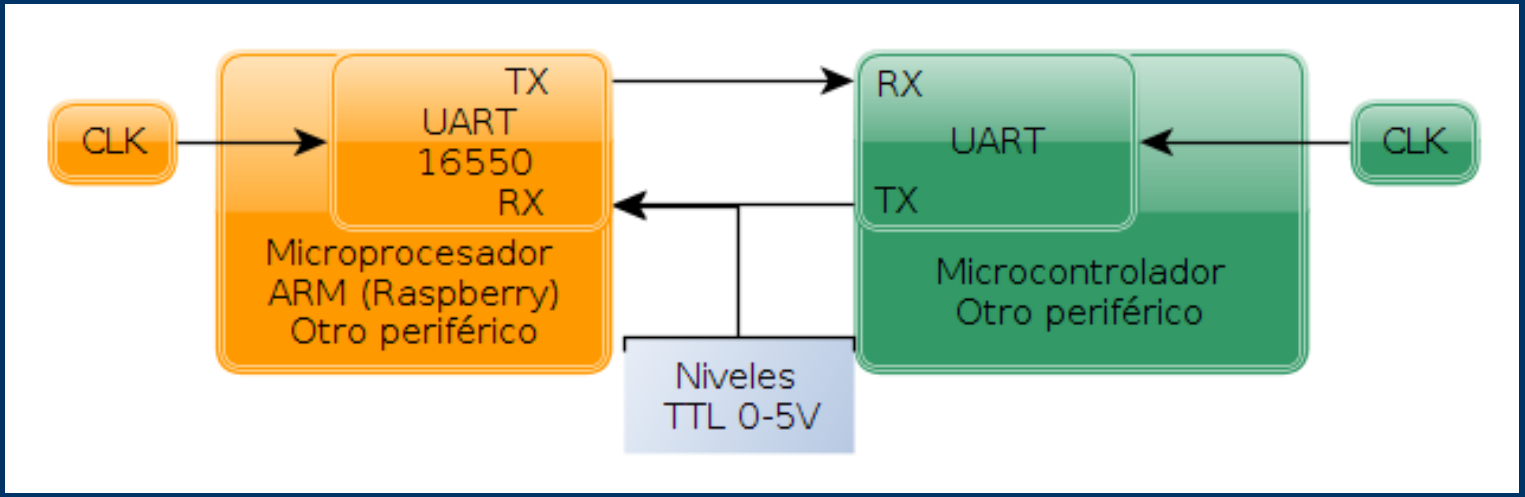


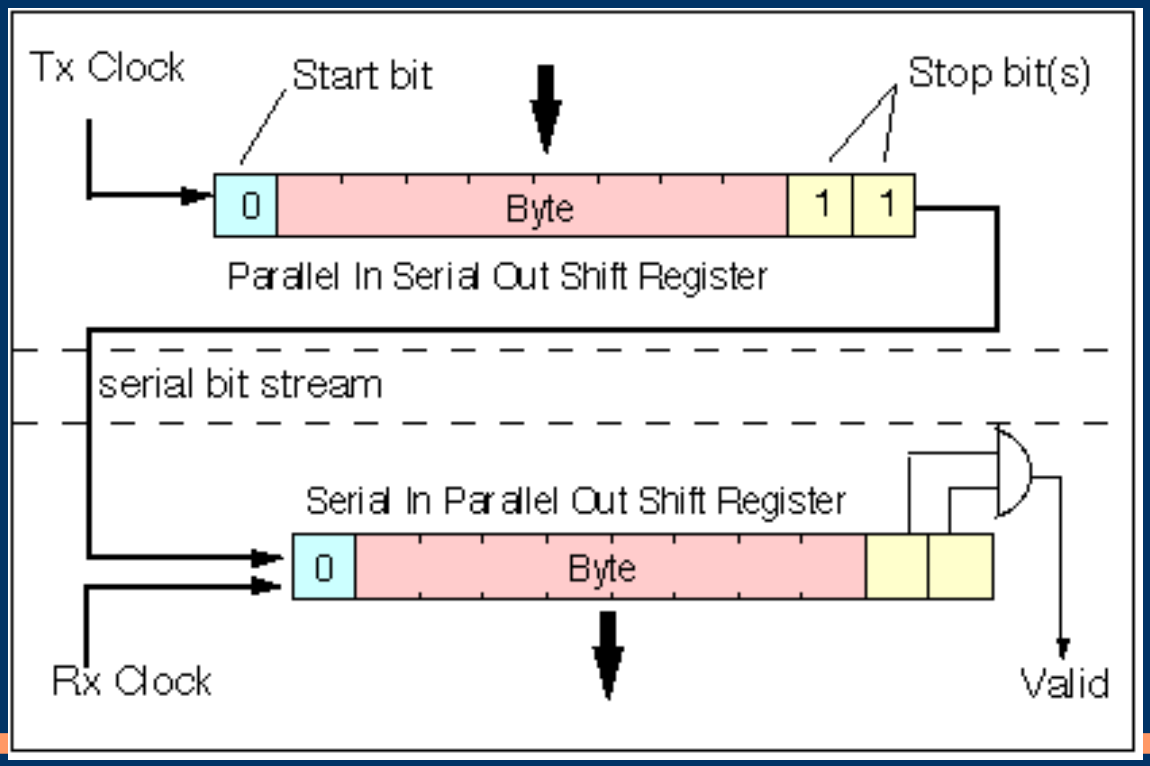


Comunicación asíncrona

En la transmisión de datos asíncronos cada byte de datos incorpora bits (start/stop) o campos (SYNC) de sincronismo. Las interfaces SCI, CAN, USB son asíncronas.

Comunicación asíncrona





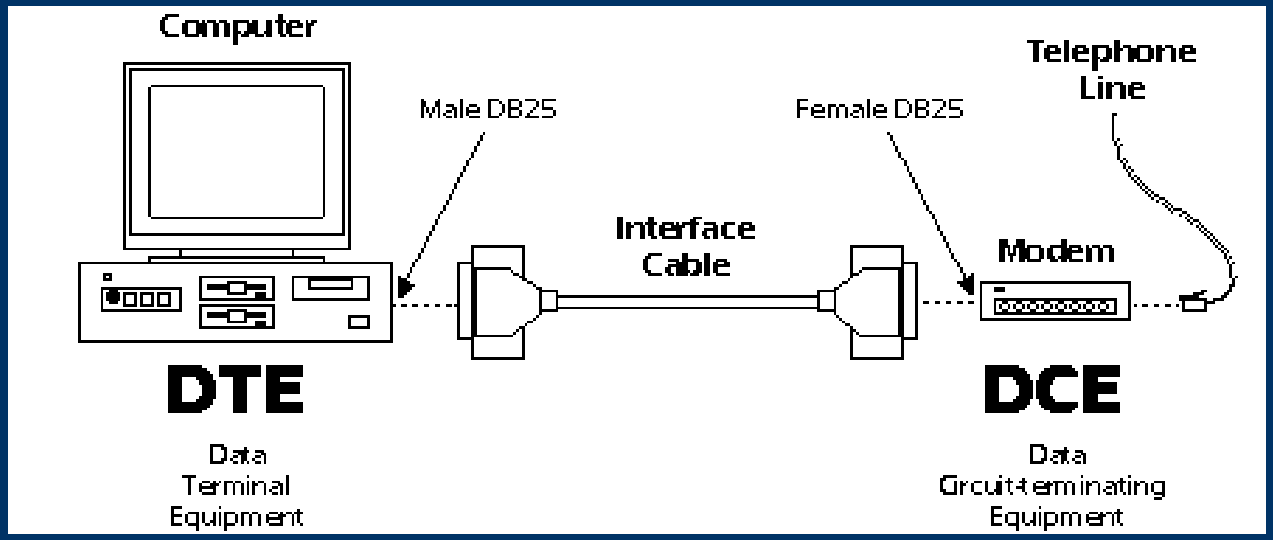
Estándar RS232

En la década de los 60’s la *Electronic Industries Association (EIA)* desarrollo un estándar de interfaz común para equipos de comunicación de datos.

Ese estándar es conocido como RS-232.

El estandar define la comunicación entre un dispositivo DTE (Data Terminal Equipment, usualmente una computadota) y un dispositivo DCE (Data Circuit-terminating Equipment, usualmente un modem).

Estándar RS232



En 1991 la EIA hace una de las modificaciones al estándar cambiando el nombre de RS232 a EIA232E.

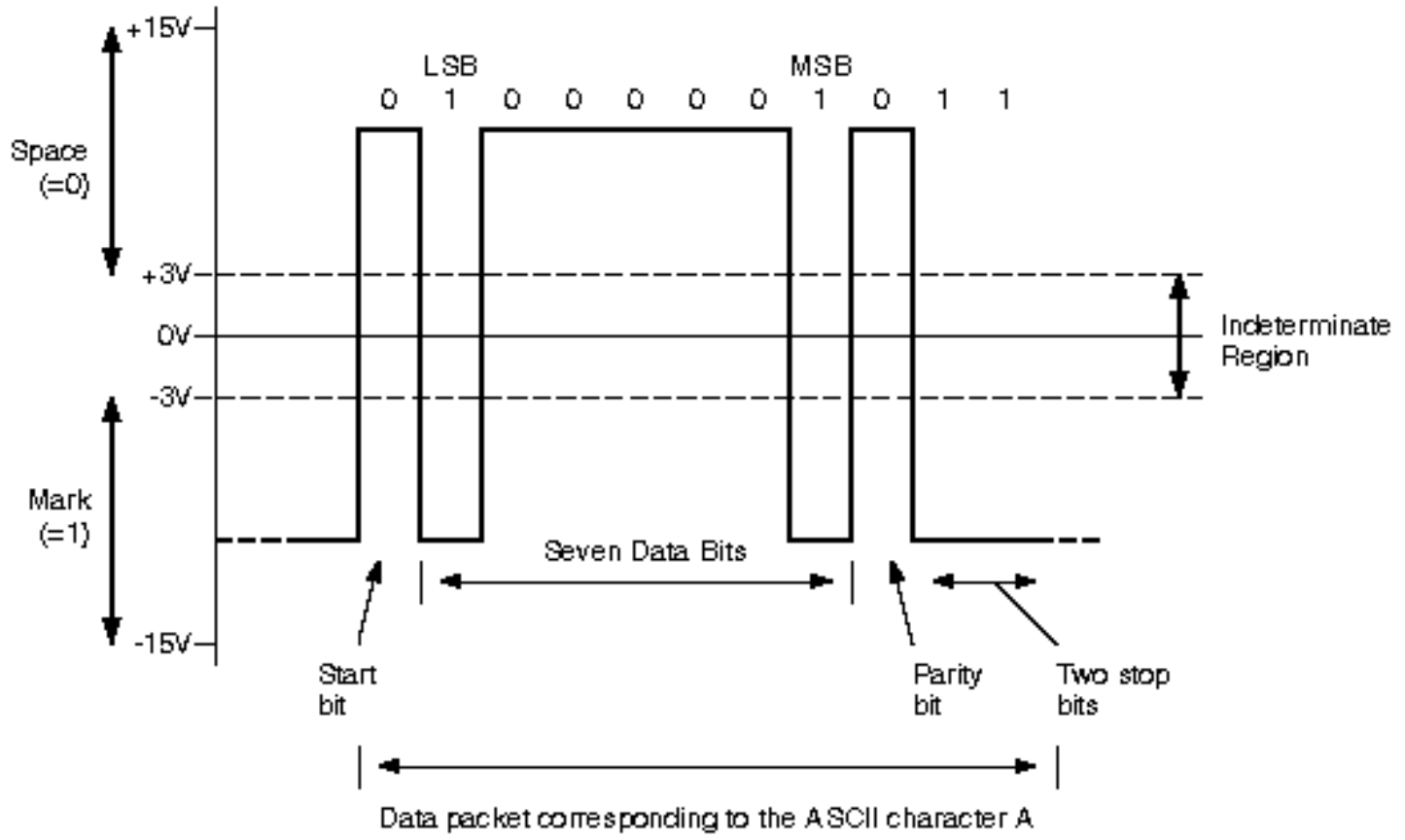
SCI es una interfaz asíncrona que se basa en este estándar, mejor conocida como interfaz serie.

Estándar RS232

Este estándar establece entre otras cosas:

Un espacio(0 lógico) entre +3 y +25 volts.
Una marca(1 lógico) entre -3 y -25 volts.
La región entre +3 y -3 volts esta indefinida.
El voltaje de circuito abierto nunca debe exceder los 25 volts (En referencia a GND).
La corriente de corto circuito no debe exceder los 500 mA.
El formato de la trama de comunicación.
Máxima velocidad en baudios, etc.

Estándar RS232



Transmisión asíncrona

En el modo de transmisión asíncrona cada byte de datos incorpora los bits de sincronismo start/stop y opcionalmente bits de paridad.

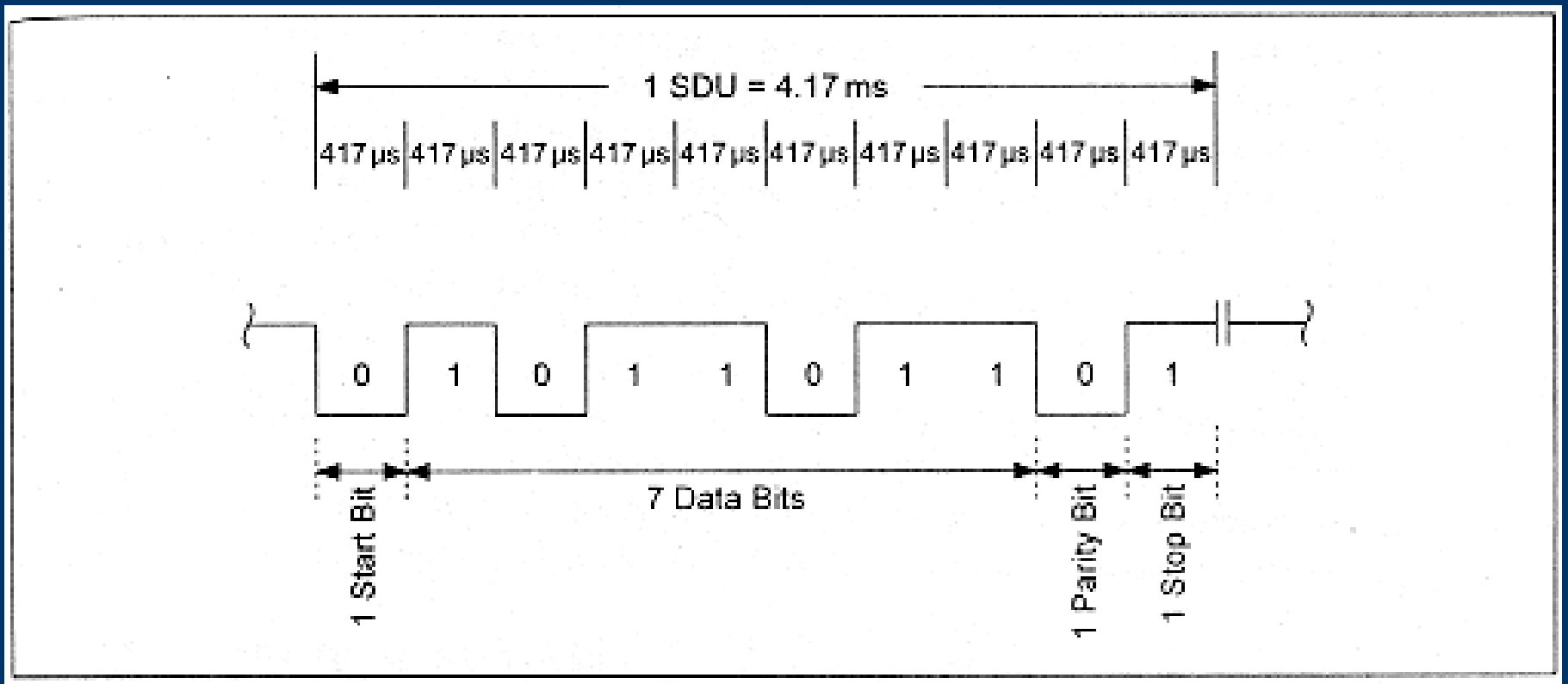
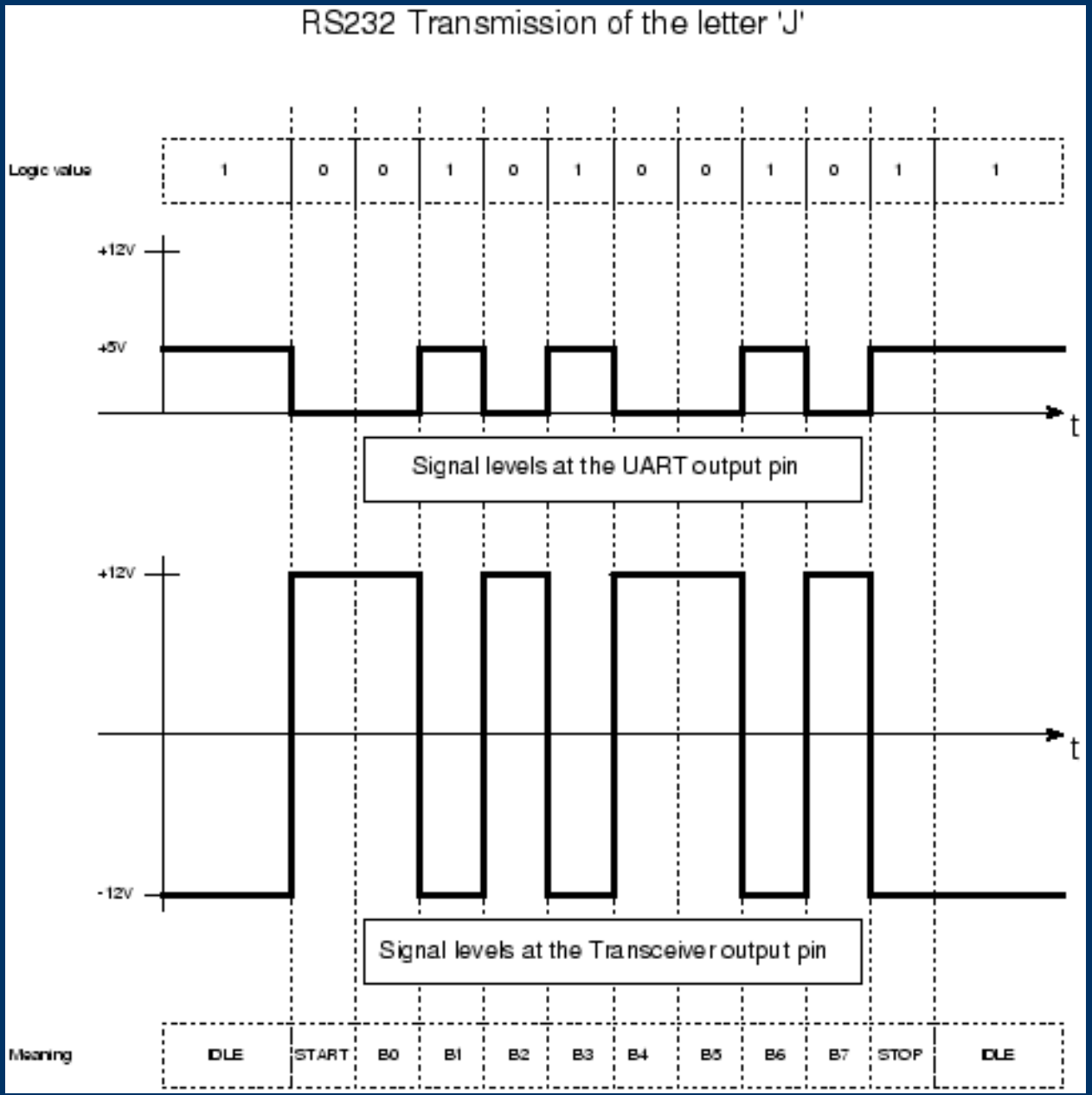


Figure 32.11: SDU. The serial data unit shown has one start bit, seven data bits, one parity bit and one stop bit. At the selected baud rate of 2400, each bit is transferred within 417 µs.

Transmisión asíncrona



Elementos de la Trama

Bits de Start y Stop. Tienen la misión de sincronizar la transmisión. El bit Start es un bit bajo y los bits Stop son bits altos.

Bit de Paridad. Este bit se utiliza para hacer detección de errores, mediante la paridad par o impar del número de unos de la trama.

Velocidad de Transmisión. Es el número de bits enviados por segundo(bps). El baudio es el número de unidades de información transmitidas por segundo.

Tasa de transferencia

Para una velocidad máxima típica de 115200 baudios tenemos:

Formato de Trama	Tasa de Transferencia (TT)
1 Bit de inicio 8 Bits por dato 1 Bit de paro	TT = Velocidad en baudios / Bits de trama TT = 115200 / 10 = 11520 Bytes/Seg
1 Bit de inicio 8 Bits por dato 1 Bit de paridad 1 Bit de paro	TT = Velocidad en baudios / Bits de trama TT = 115200 / 11 = 10472 Bytes/Seg
1 Bit de inicio 8 Bits por dato 1 Bit de paridad 2 Bits de paro	TT = Velocidad en baudios / Bits de trama TT = 115200 / 12 = 9600 Bytes/Seg

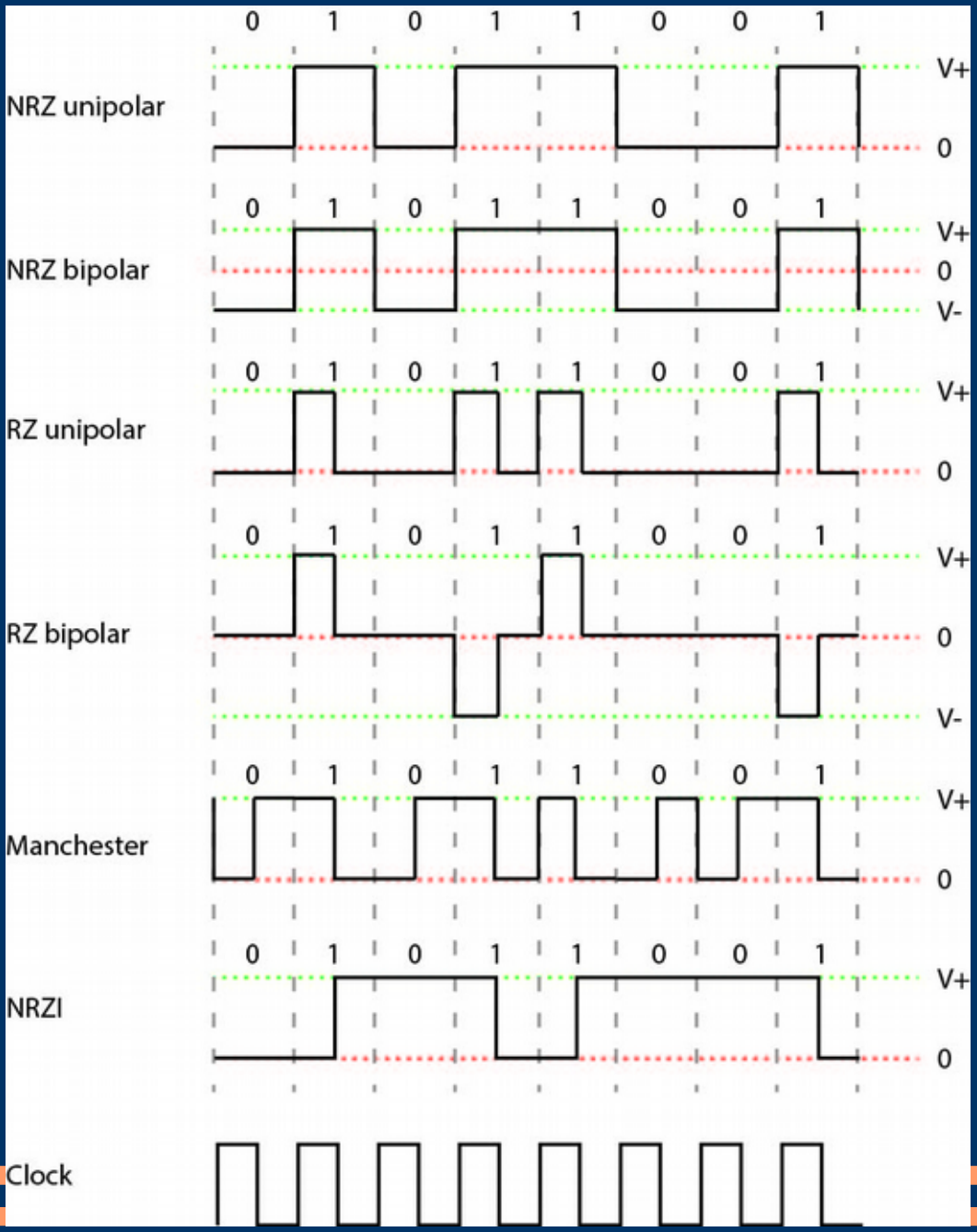
Mecanismos de codificación

Existen mecanismos de codificación serial que ayudan a representar niveles lógicos. Algunos de ellos son:

Non-return-to-zero (NRZ).
Non-return-to-zero-inverted (NRZI).
Return-to-zero (RZ).
Manchester.

Cada uno tiene diferentes características que los hacen útiles en diferentes aplicaciones.

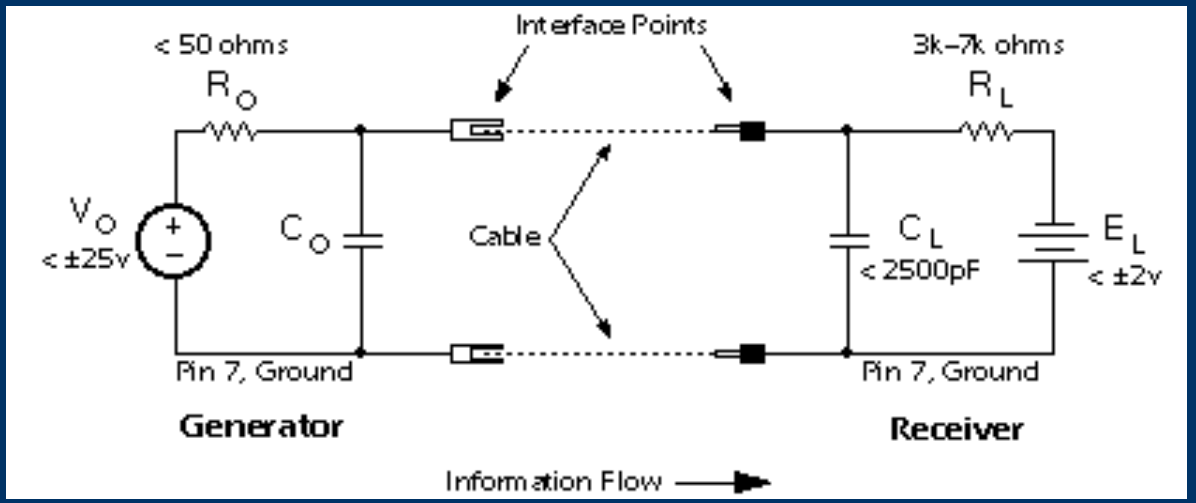
Mecanismos de codificación



Codificación NRZ

Esta codificación permite tener un mayor intervalo de voltaje por lo que la distancia de transmisión es mucho mayor (15 metros) que la que se tiene en lógica TTL.

NRZ es usada en el estándar EIA232E.



Modos de comunicación

Simplex. Es unidireccional.

Half Duplex. Es bidireccional pero NO simultánea.

Full Duplex. Es bidireccional simultánea.

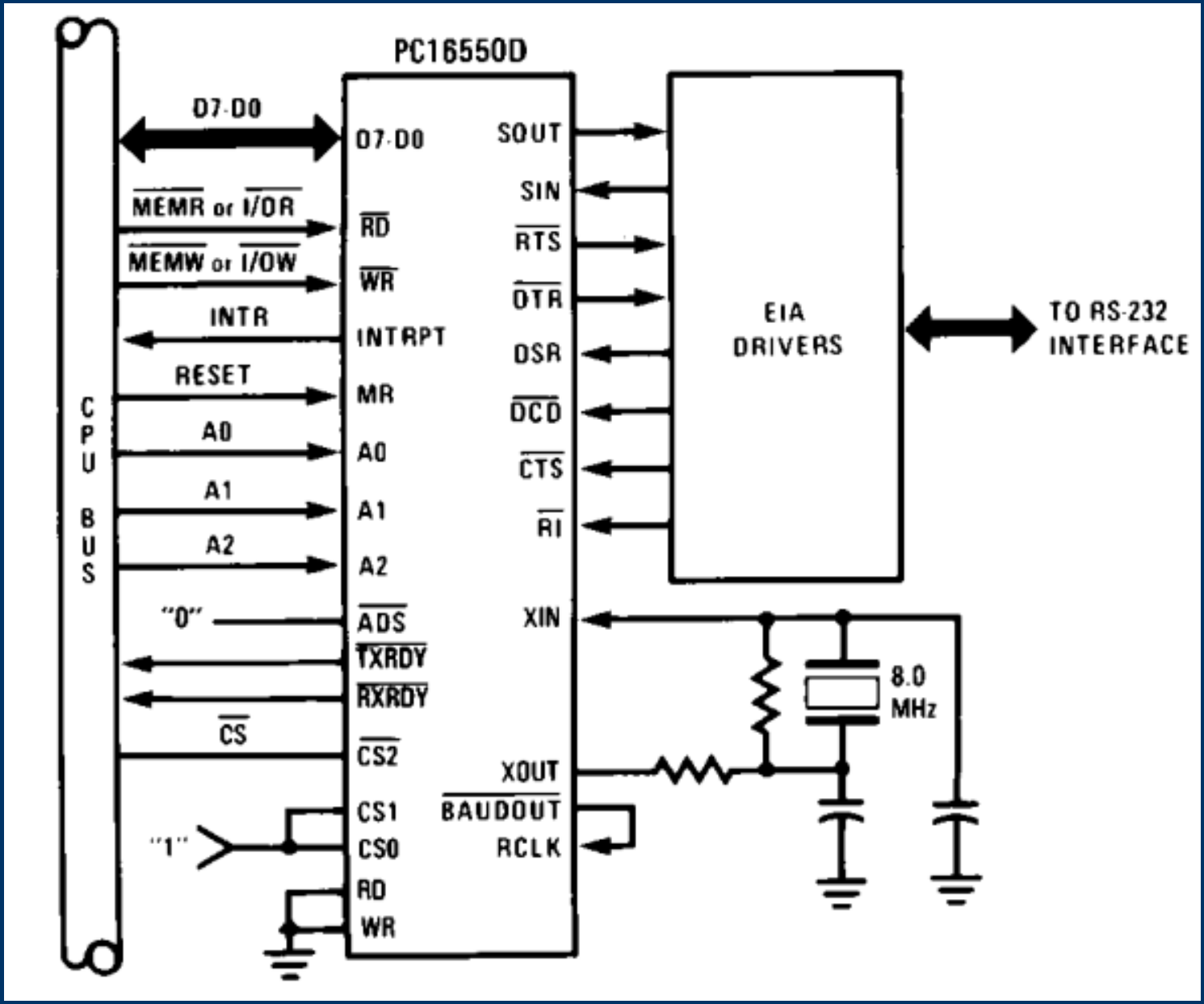
La comunicación serial por UART usa comunicación Full Duplex mediante las líneas Tx y Rx.

UART

El corazón de la interfaz EIA232 es el UART (Universal Asynchronous Receiver Transmitter) que es programable (8251, 16450 y 16550). Este circuito realiza las conversiones de paraleloa serial y viceversa.

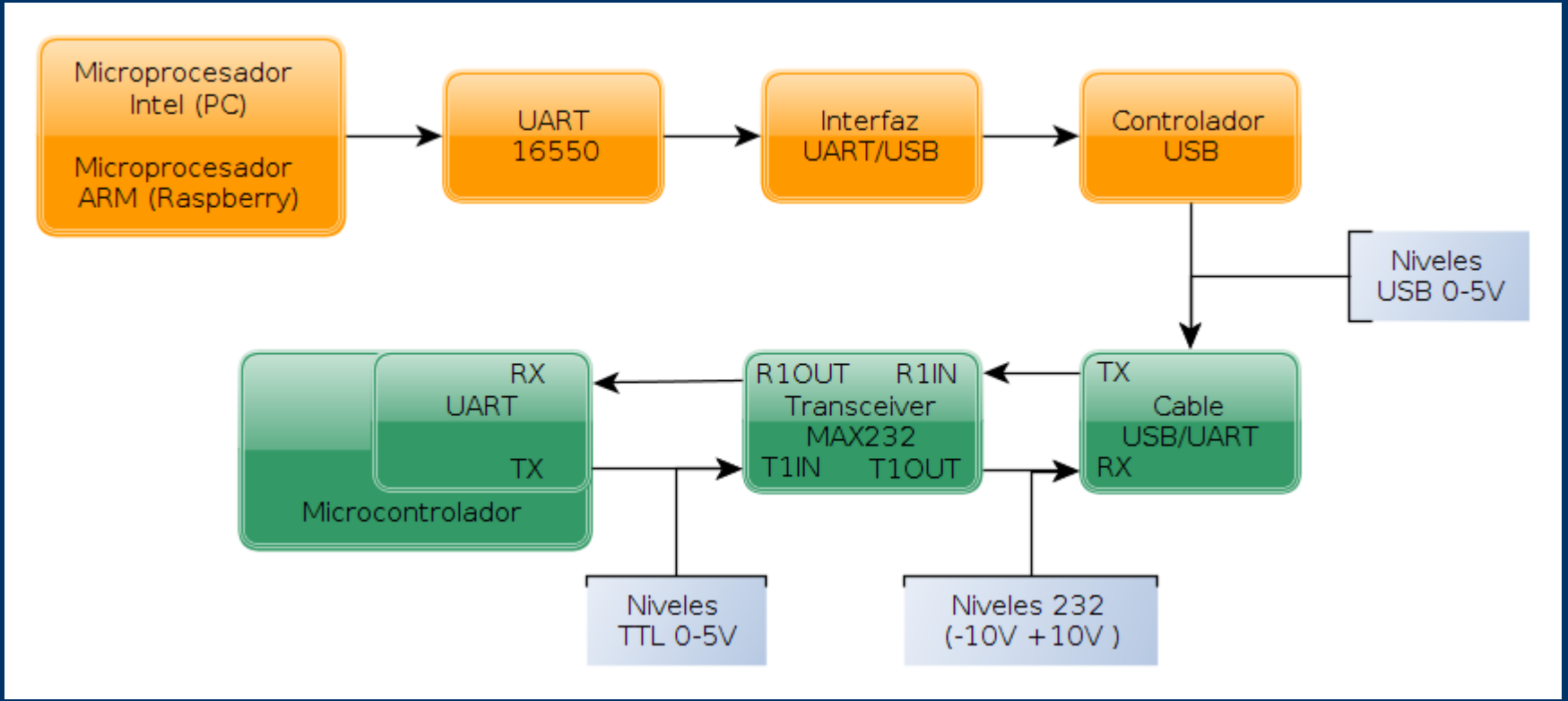
Este circuito maneja niveles TTL por lo que se requiere la lógica necesaria para realizar la conversión entre los niveles de tensión de la computadora(0v y 5v) y las señales eléctricas de la norma EIA232(-25v y 25v).

UART


PC16550D
D7-D0
D7-D0
SOUT
SIN
MEMR or I/OR
RD
RTS
MEMW or I/OW
WR
OTR
INTR
INTRPT
DSR
RESET
MR
DCD
A0
A0
CTS
A1
A1
RI
A2
A2
"0"
ADS
XIN
TXRDY
RXRDY
8.0
MHz
CS
CS2
XOUT
CS1
BAUDOUT
"1"
CS0
RCLK
RD
WR
CPU BUS
EIA
DRIVERS
TO RS-232
INTERFACE

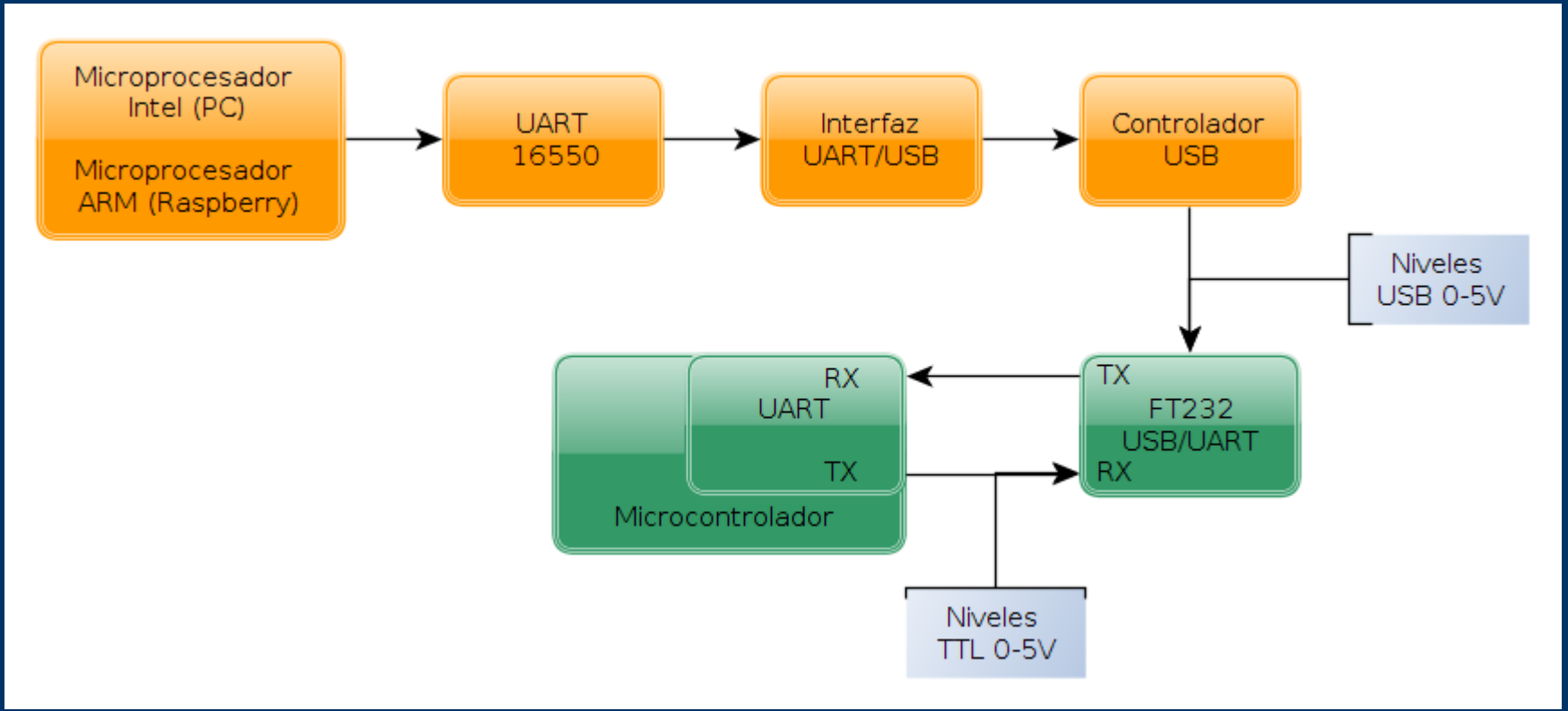
Topología I

En esta topología se realiza la conexión de la interfaz serie de un microcontrolador hacia un microprocesador usando su controlador USB mediante un cable USB – Serial.



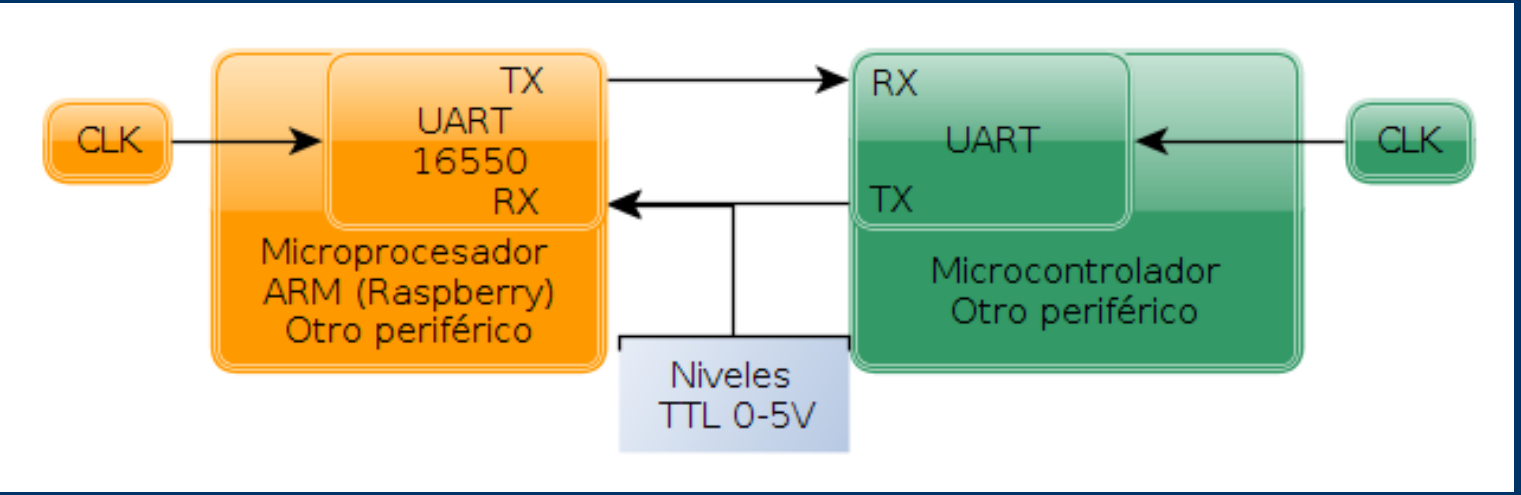
Topología II

En esta topología se realiza la conexión de la interfaz serie de un microcontrolador hacia un microprocesador usando su controlador USB mediante un módulo FT232.



Topología III

En esta topología se realiza la conexión de la interfaz serie de un microcontrolador hacia un microprocesador embebido o cualquier periférico con interfaz UART.



Comunicación serie vs paralela

Comunicación serie	Comunicación paralela
Se usan niveles de ±25V con codificación NRZ bipolar	Se usan niveles TTL (0-5V) con codificación NRZ unipolar
Se requieren pocos cables para la comunicación, solo 3 en comunicación con UART	Se requieren varios cables para la comunicación, almenos 9 con un puerto de 8 bits
La distancia de comunicación es de 15 m	La distancia de comunicación es aprox. 5 m
La velocidad de comunicación es lenta. La tasa de transferencia es de 11520 B/S para una velocidad de 115200 baudios y una trama mínima	La velocidad de comunicación es rápida. La tasa de transferencia es del orden de MB/S

Configuración del UART en Linux

Linux organiza sus archivos y directorios en un árbol general interconectado, que comienza en el directorio raíz y se extiende a los directorios de sistema y usuario. La organización y el diseño de sus directorios de sistema se determina con el Estándar de Jerarquía de Sistema de Archivos (FHS – File System Hierarchy Standard).

Una forma de configurar el UART en Linux es mediante el FHS.

Configuración del UART en Linux

El directorio /dev contiene interfaces de archivos generados dinámicamente para dispositivos como la terminal y la impresora.

En este directorio se crean los dispositivos de carácter para tener acceso al UART del procesador. Estos dispositvos se crean con el nombre tty*. Para el caso de los módulos FT232, son creados con el nombre de ttyUSB0, ttyUSB1, etc.

Configuración del UART en Linux

Los dispositivos en Linux son tratados como archivos, por lo que lo primero a realizar es arbrir el archivo.

```
#include <fcntl.h>
int open(const char *pathname, int flags);
```

O_RDWR - Se abre el descriptor para lectura y escritura

O_NOCTTY - El dispositivo terminal no se convertira en el terminal del proceso

~O_NONBLOCK - Se hace bloqueante la lectura de datos

Configuración del UART en Linux

Se usa la estructura termios, la cual contiene los modos usados en la comunicación y configuración del UART. Los modos son:

```
tcflag_t c_iflag;      /* input modes */
tcflag_t c_oflag;      /* output modes */
tcflag_t c_cflag;      /* control modes */
tcflag_t c_lflag;      /* local modes */
cc_t c_cc[NCCS];  /* special characters */
```

Configuración del UART en Linux

Se usan las funciones:

```
// Configura la velocidad de salida del UART
cfsetospeed( &struct termios, baudios )
// Configura la velocidad de entrada del UART
cfsetispeed( &struct termios, baudios )
// Limpia el buffer de entrada
tcflush( fd, TCIFLUSH )
// Limpia el buffer de salida
tcflush( fd, TCOFLUSH )
```

Contacto

vgarciaortega@yahoo.com.mx

Gracias……..