Conexión a puerto serial (sin trucos)

Esvin González

Comunicación

Comunicación entre computadoras

Estándares para definir marca/espacio

- TTL
- RS232
- Lazo de corriente de 20mA

Estándar	"1“ Lógico (Marca)	"0“ Lógico (Espacio)
TTL	5V	0
Lazo 20mA	20mA	0mA
RS 232C	-3V a -15V	+3V a +15V

Velocidad de transmisión

- La velocidad a la que se transmiten los datos en un enlace de comunicación serial debe ser la misma en ambos dispositivos, se mide en baudios.
  - Velocidades estándar son:
    - 75
    - 110
    - 150
    - 300
    - 600
    - 1200
    - 2400
    - 4800
    - 9600
    - 19200

Canales de transmisión

- Simplex
  - Una sola línea, en un solo sentido.
- Duplex
  - Una sola línea, pero en ambos sentidos.
- Full Duplex
  - Dos líneas, una para cada sentido.

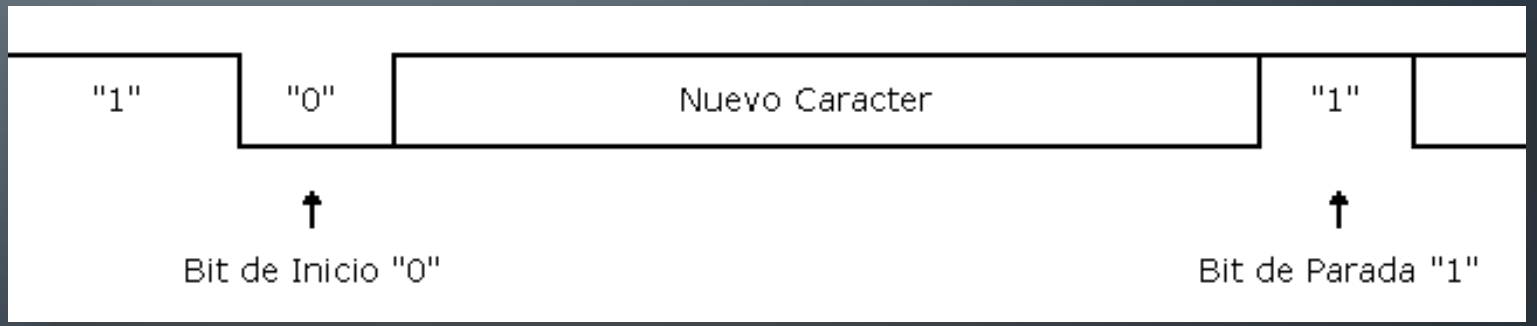
**Adicional en cada una se necesita tener un nivel de referencia para ambos dispositivos (tierra en común)*

Modos de transmisión

- Comunicación serial síncrona
  - Para llevar el control de la transferencia utiliza una línea de reloj.
- Comunicación serial asíncrona
  - Controla la comunicación a través de un marco definido.
  - No utiliza línea de reloj.
  - Manejo de errores en la transmisión.

Comunicación serial asíncrona

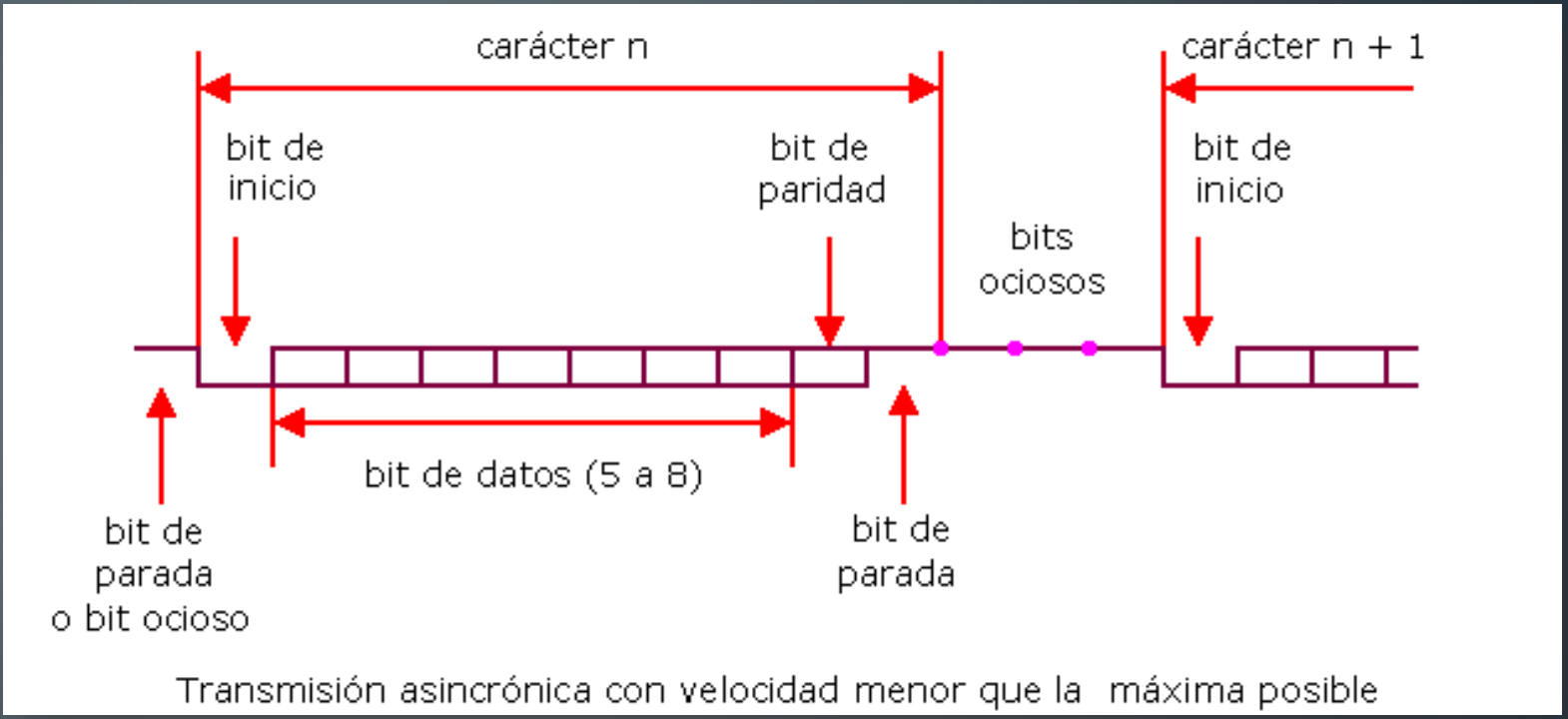
- Para el control de la transmisión se debe contar con un marco uniforme para la comunicación, este marco se compone de varios elementos.
  - Bit de inicio
  - Bit de parada
  - Bit de paridad
  - Bits del mensaje



Reglas para la transmisión

- Cuando no se envían datos por la línea, ésta se mantiene en estado alto (1).
- Cuando se desea transmitir un carácter, se envía primero un bit de inicio que pone la línea a estado bajo (0) durante el tiempo de un bit.
- Durante la transmisión, si la línea está a nivel bajo, se envía un 0 y si está a nivel alto se envía un 1.
- A continuación se envían todos los bits del mensaje a transmitir con los intervalos que marca el reloj de transmisión. Por convenio se transmiten entre 5 y 8 bits.
- Se envía primero el bit menos significativo, siendo el más significativo el último en enviarse.
- A continuación del último bit del mensaje se envía el bit (o los bits) del final que hace que la línea se ponga a 1 por lo menos durante el tiempo mínimo de un bit.

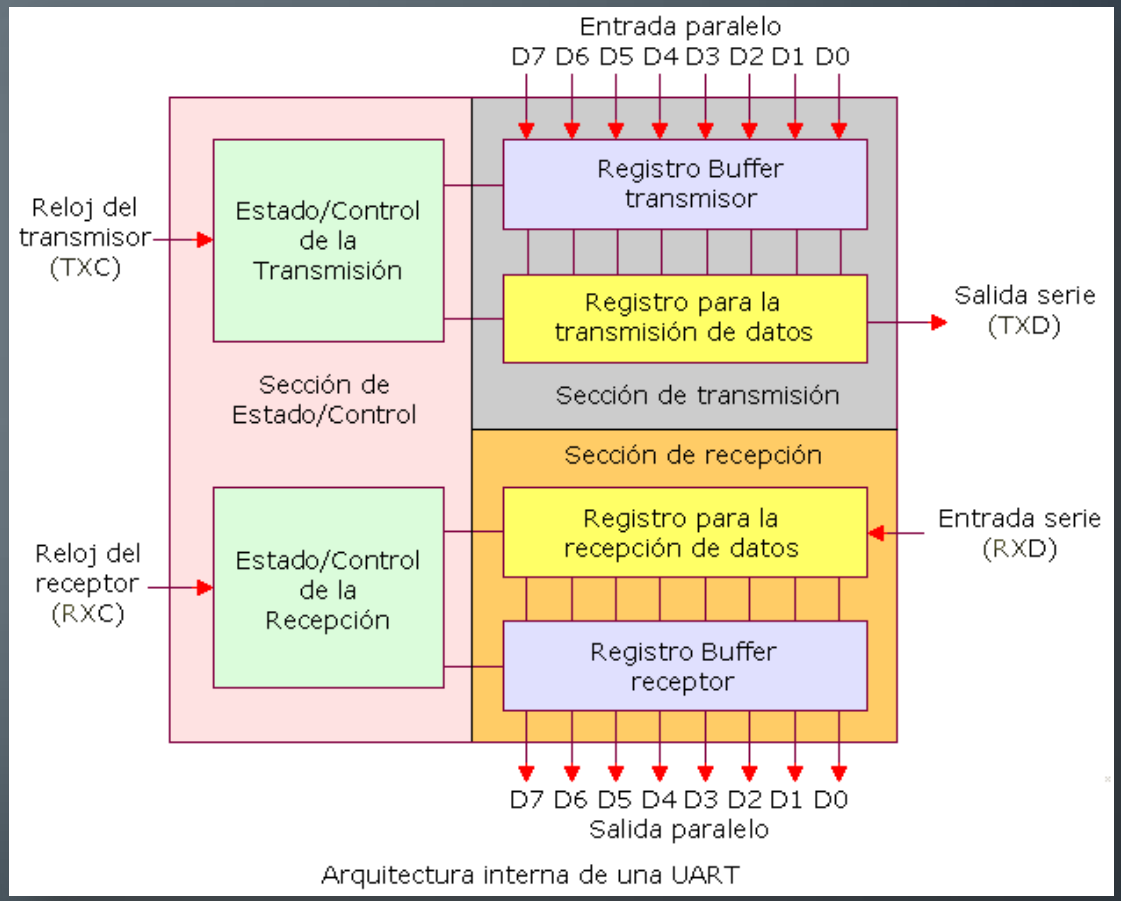
Marco de la transmisión



Transmisión asincrónica con velocidad menor que la máxima posible

Las UART y su arquitectura

- Universal Asynchronous Receiver-Transmitter



Arquitectura interna de una UART

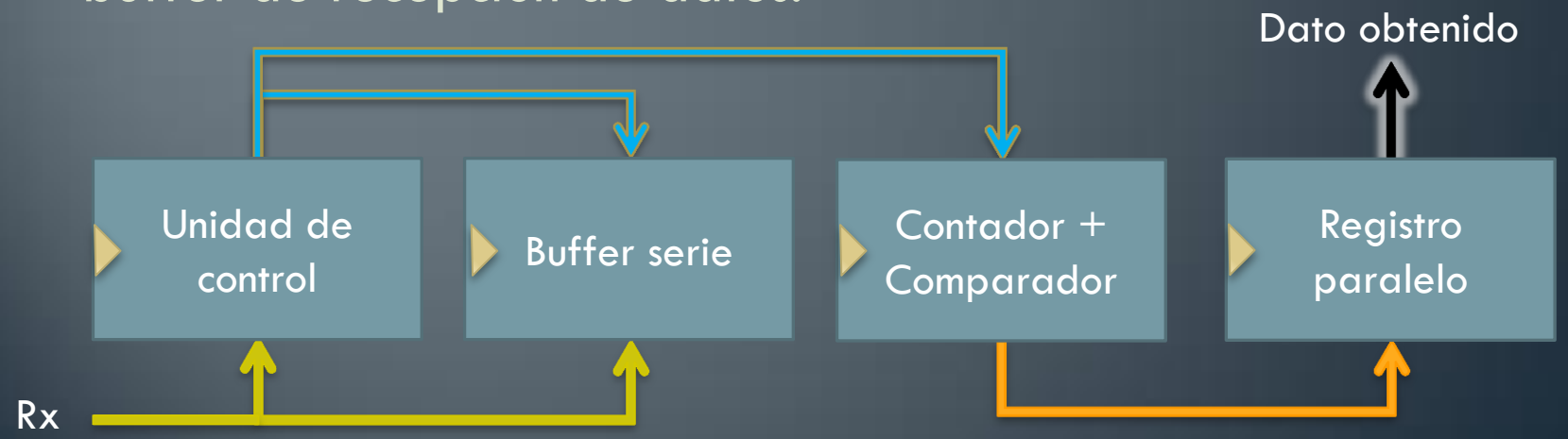
Construyendo el módulo Rx

¿Qué necesitamos"?

- Circuito de control
  - Diseño con lógica secuencial, flip-flops, contadores, comparadores, etc.
- Buffer para recibir datos en serie
  - Registro de carga en serie y salida en paralelo.
  - El elegido es: 74ls164
- Registro para almacenar los datos recibidos
  - Registro de carga en paralelo.
  - El elegido es: 74ls194

Un esquema muy general...

- **Rx** es la única línea externa que recibiremos.
- **CLK** es el reloj interno de nuestro circuito (llega a todos los CI).
- **LOAD** será el resultado de comparar la cantidad de bits leídos y la cantidad de bits que se espera del mensaje.
- **ACTIVAR** será quien lleve el control del contador de bits y del buffer de recepción de datos.

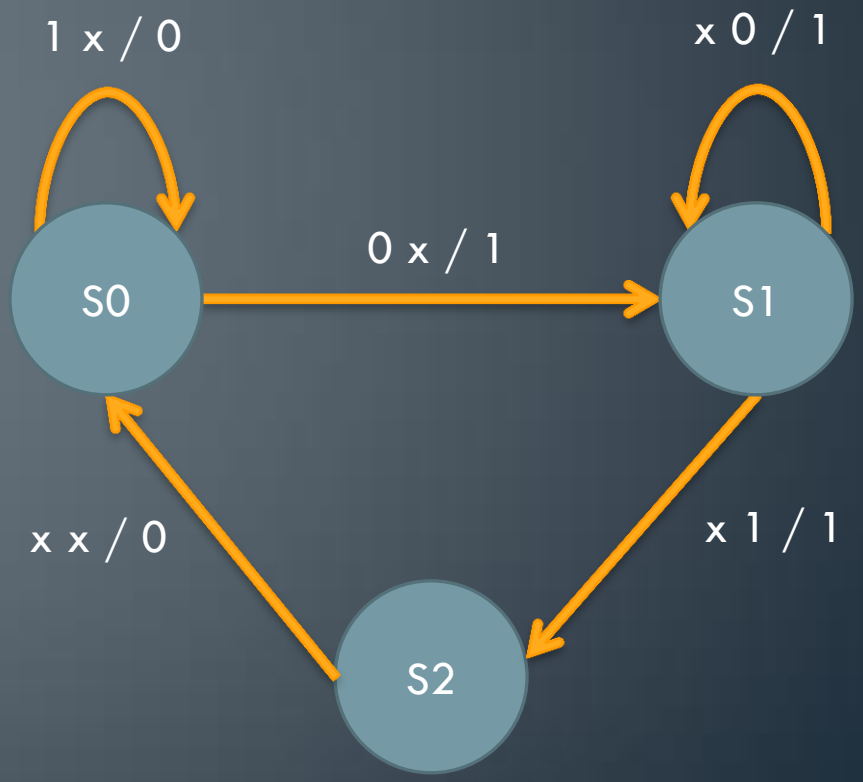


Primero el control…

- Para ello emulamos el comportamiento de la lectura a través de un diagrama de estados, red Mealy.

Entradas Salidas

Rx LOAD / ACTIVAR



Aplicamos la lógica secuencial...

	Estado actual			Entradas		Siguiente estado			Flip flops				Salida
	Qb		Qa	Rx	Ld	Qb+		Qa+	Jb	Kb	Ja	Ka	Act
0	0	S0	0	0	0	0	S1	1	0	x	1	x	1
1	0	S0	0	0	1	0	S1	1	0	x	1	x	1
2	0	S0	0	1	0	0	S0	0	0	x	0	x	0
3	0	S0	0	1	1	0	S0	0	0	x	0	x	0
4	0	S1	1	0	0	0	S1	1	0	x	x	0	1
5	0	S1	1	0	1	1	S2	0	1	x	x	1	1
6	0	S1	1	1	0	0	S1	1	0	x	x	0	1
7	0	S1	1	1	1	1	S2	0	1	x	x	1	1
8	1	S2	0	0	0	0	S0	0	x	1	0	x	0
9	1	S2	0	0	1	0	S0	0	x	1	0	x	0
10	1	S2	0	1	0	0	S0	0	x	1	0	x	0
11	1	S2	0	1	1	0	S0	0	x	1	0	x	0
12	1	--	1	0	0	x	--	x	x	x	x	x	x
13	1	--	1	0	1	x	--	x	x	x	x	x	x
14	1	--	1	1	0	x	--	x	x	x	x	x	x
15	1	--	1	1	1	x	--	x	x	x	x	x	x

Mapas de Karnaugh

- Para no morir en el intento de hacer todos los mapas, recomendación personal: http://www.32x8.com/

Jb = Qa • Ld

Kb = 1

Ja = Rx‘ • Qb'

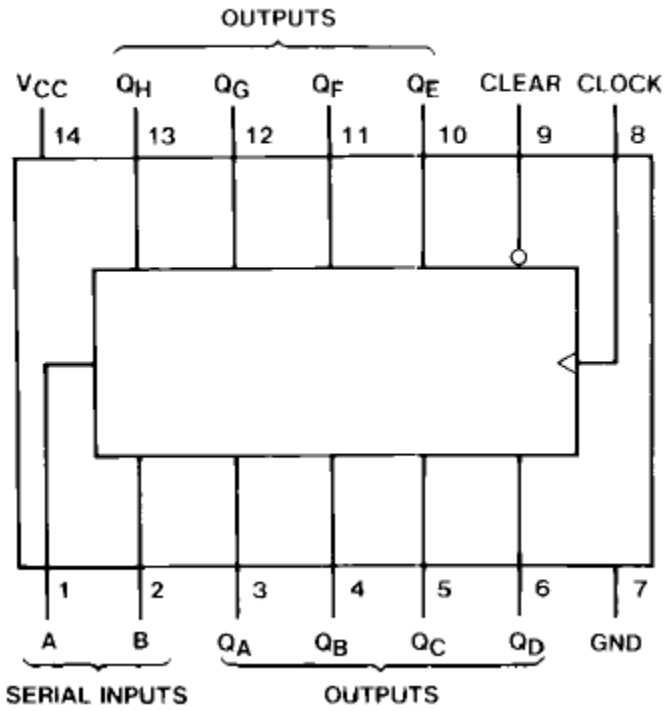
Ka = Ld

Act = Qa + Rx' • Qb ' (Activar)

El buffer (recibe de Rx)

- Para el buffer que recibe los datos de forma serial, el siempre confiable: 74LS164

Connection Diagram



Function Table

Inputs				Outputs			
Clear	Clock	A	B	Q_A	Q_B	...	Q_H
L	X	X	X	L	L	...	L
H	L	X	X	Q_{A0}	Q_{B0}	...	Q_{H0}
H	↑	H	H	H	Q_{An}	...	Q_{Gn}
H	↑	L	X	L	Q_{An}	...	Q_{Gn}
H	↑	X	L	L	Q_{An}	...	Q_{Gn}

H = HIGH Level (steady state)
L = LOW Level (steady state)
X = Don't Care (any input, including transitions)
↑ = Transition from LOW-to-HIGH level
Q_{A0}, Q_{B0}, Q_{H0} = The level of Q_A, Q_B, or Q_H, respectively, before the indicated steady-state input conditions were established.
Q_{An}, Q_{Gn} = The level of Q_A or Q_G before the most recent ↑ transition of the clock; indicates a one-bit shift.

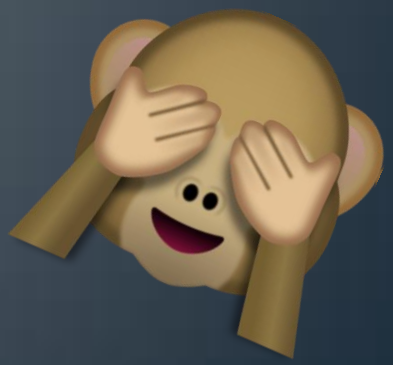
¿Y para la carga de datos en paralelo?

- Para cumplir la función de registro, el completísimo y muy versátil: 74LS194

Inputs										Outputs			
Clear	Mode		Clock	Serial		Parallel				Q_A	Q_B	Q_C	Q_D
	S1	S0		Left	Right	A	B	C	D				
L	X	X	X	X	X	X	X	X	X	L	L	L	L
H	X	X	L	X	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}
H	H	H	↑	X	X	a	b	c	d	a	b	c	d
H	L	H	↑	X	H	X	X	X	X	H	Q_{An}	Q_{Bn}	Q_{Cn}
H	L	H	↑	X	L	X	X	X	X	L	Q_{An}	Q_{Bn}	Q_{Cn}
H	H	L	↑	H	X	X	X	X	X	Q_{Bn}	Q_{Cn}	Q_{Dn}	H
H	H	L	↑	L	X	X	X	X	X	Q_{Bn}	Q_{Cn}	Q_{Dn}	L
H	L	L	X	X	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}

Ojo con el contador...

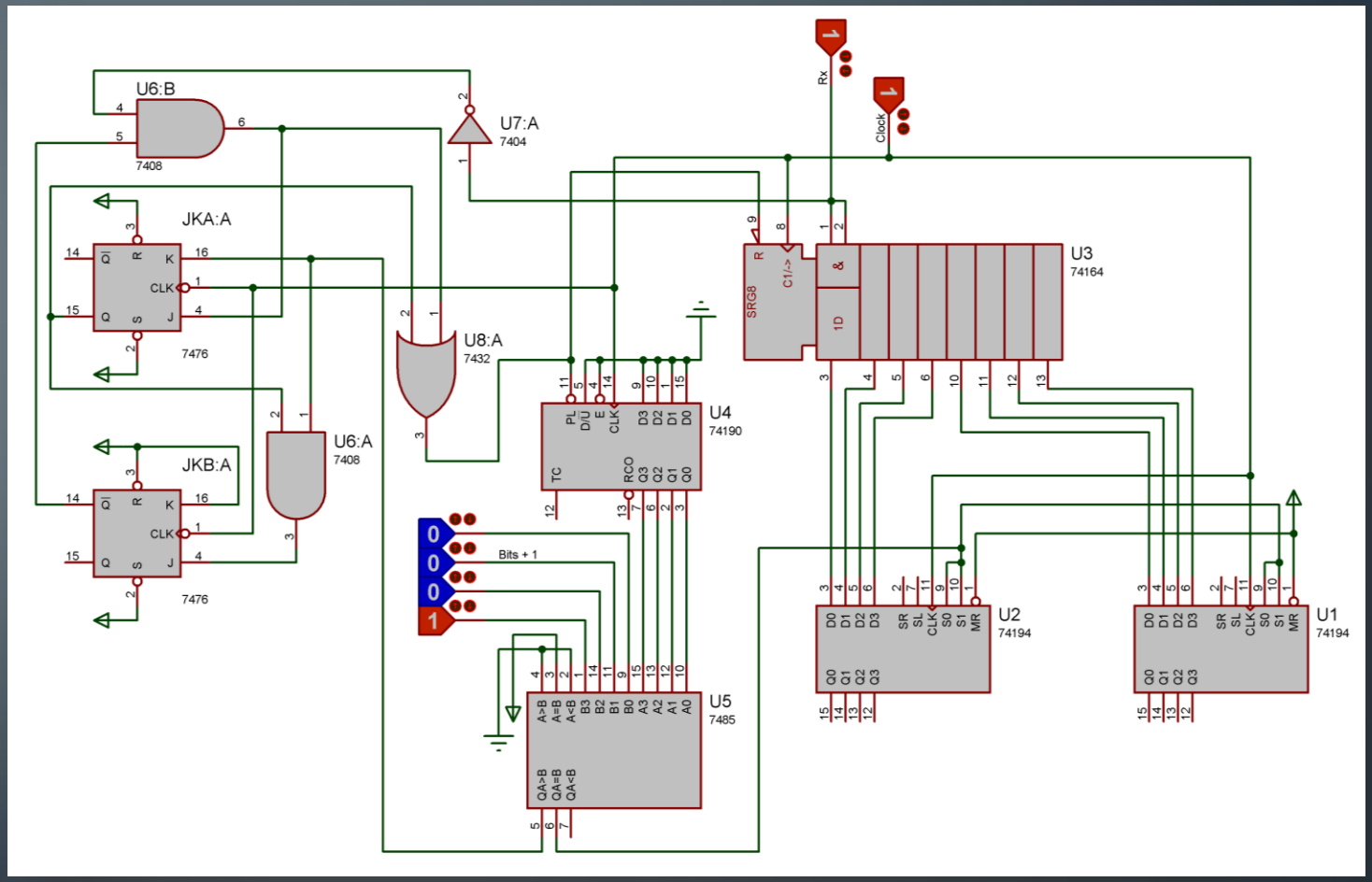
- Para fines de este taller en la simulación (y en el circuito que no terminé) utilicé el 74LS190, contador programable con carga en pedestal y bla, bla, bla... Pero que se resetea al llegar a 10.
- Si hacemos cuentas, el primer bit (bit de inicio), los 8 bits del mensaje, y el último bit (bit de parada), en un marco simple son... 10 bits, si este contador llega a 10... No va a haber chance de completar el algoritmo y entonces...



*Te odio 74LS190, pero eres útil.

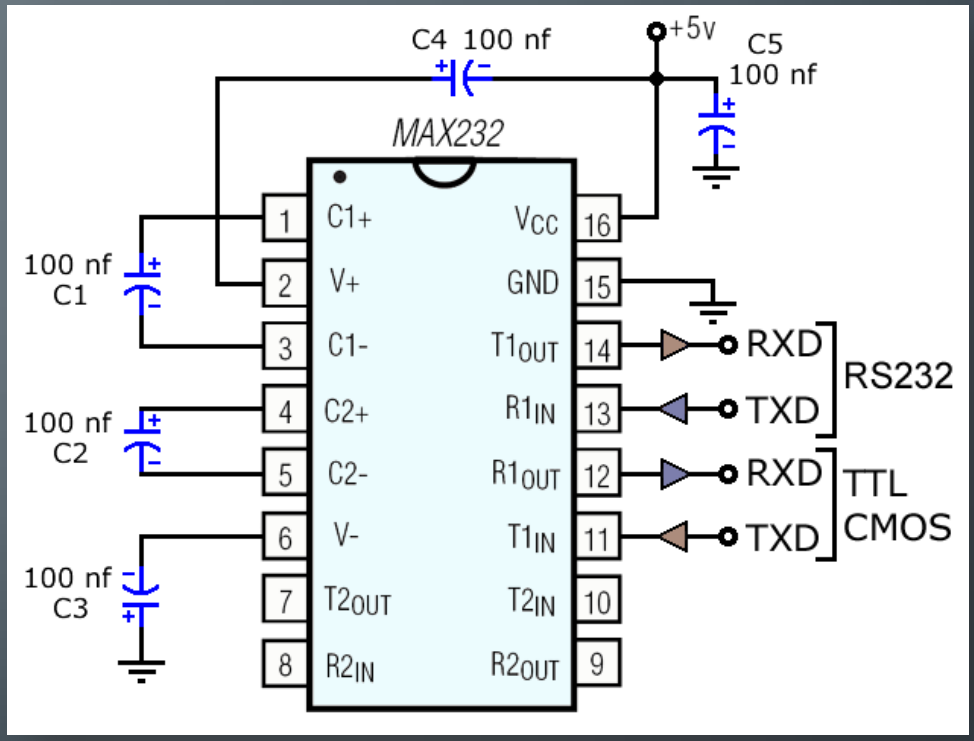
Luego de pasar horas en Proteus...

- La simulación del circuito completo está lista.



Ya tengo mi módulo Rx ¿Ahora qué?

- Hay que conectarlo a la PC por medio del DB9, pero para eso es necesario crear un puente entre el estándar RS232 de la PC y el TTL de nuestros integrados, para ello viene al rescate el MAX232.



Problemas sin resolver...

¿Cómo sincronizar el reloj del circuito con el reloj de la PC?

- NE555 (lo ideal)
- Cristales de cuarzo (lo pro)
- Microcontrolador (lo tonto)

¿Cómo programar una app que se comunique con mi circuito?

- Es más sencillo de lo que parece si se usa un lenguaje “amigable” con el puerto serial, cualquier lenguaje de .Net permite hacer cosas sencillas como lo siguiente.
- ¿Y si quiero usar Java?

Recomendaciones

- Para la comunicación serial asíncrona es indispensable asimilar los conceptos del marco de transmisión.
- La gestión del reloj en los circuitos se puede hacer con un reloj muy alto que al empezar la transmisión ceda el paso a un reloj más lento (lean de subdivisión de reloj).
- En el proceso de comunicación inversa (desde el circuito enviar información a la PC) deben ensamblar la cadena de bits con el mismo marco de transmisión antes de enviarla.
- Pregunten, pregunten y pregunten.

Referencias bibliográficas

- Lógica digital y diseño de computadores
  - Morris Mano
- El cuaderno de Orga de mi novia
  - Sarina Bolaños
- La comunicación serie
  - Disponible en: http://goo.gl/8r2uPU
- Que sabes tú de la comunicación serial
  - Disponible en: http://goo.gl/MkD8jC

Gracias por su atención

- Este y otros materiales están disponibles en mi cuenta de GitHub.

/Esvux