COMENTARIO TÉCNICO

Buceando en los MCUs Freescale.....

Por Ing. Daniel Di Lella
Dedicated Field Application Engineer
EDUDEVICES
www.edudevices.com.ar
dilella@arnet.com.ar

www.edudevices.com.ar

"Matemática de Punto Flotante"

Por el CETAD – UNLP
Coordinador Ing. Jose Rapallini
Participantes:
Sebastián Ledesma
e-mail: sledesma@barcala.ing.unlp.edu.ar
Federico Costantino,
e-mail: fcostant@barcala.ing.unlp.edu.ar
Jorge R. Osio,
e-mail: josio@gioia.ing.unlp.edu.ar

2da. Parte.

Nota: La presente serie de artículos está basada en el trabajo realizado por el Centro de Técnicas Analógicas – Digitales (CETAD) de la universidad Nacional de La Plata, y constituye una Nota de Aplicación para los MCU´s de 8 Bits de las familias HC908 y HC9S08 de Freescale Semiconductor.

Comunicación serie entre la PC y el MCU.

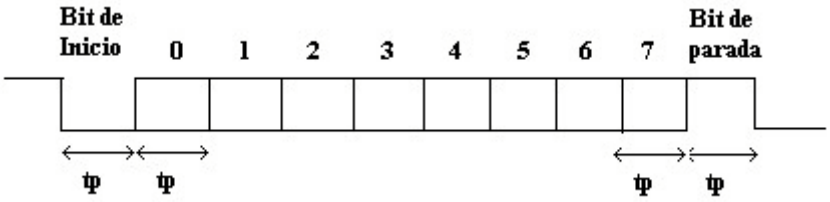
Terminologías y Conceptos de la Comunicación serie.

Operación Half-Duplex.

En la operación Half-duplex, solo un nodo transmite a la vez. El MCU no puede recibir mientras está transmitiendo, y no puede transmitir mientras está recibiendo. Esta incapacidad es una desventaja en comparación con el Hardware SCI, que puede transmitir y recibir al mismo tiempo. Esto se conoce como sistema **full duplex.**

Formato de Transmisión.

El SCI usa el formato estándar de transmisión **non-return-to-zero (NRZ)**, que consiste en un bit de inicio seguido de 8 bits de datos y un bit de parada. Esto es comúnmente referenciado como Formato 8-N-1 (8 bits de datos, sin bit de paridad, un bit de parada). El dato es transmitido y recibido desde el bit menos significativo (LSB). Cada bit tiene una duración T_p que define el Baud Rate (tasa de baudio).



t_p = periodo de un bit = 1/ tasa de baudio

La comunicación serie trabaja con la línea en alto (idle line), y el bit de inicio es un cero lógico. Cada bit de dato puede ser ya sea un uno o un cero lógico. El bit de parada es un uno lógico. El bit de inicio, el bit de parada y cada bit de dato constituyen una trama de datos.

Detección de error de Trama.

El bit de parada está definido como un uno lógico, si el bit de parada recibido es un cero, quiere decir que ocurrió un error de Trama.

Descripción de la comunicación serie en el MCU.

Los requerimientos para que las rutinas funcionen correctamente son mínimos y, como es de esperarse poseen Pequeños errores de Timming pero en la mayoría de los casos son inferiores al 1%, y en algunos casos no existen.

Cabe destacar que la rutina de Recepción Posee solamente control de Error de Trama (Frame) y no otros tipos de errores (Paridad, Ruido, etc).

La velocidad de Transmisión como la de Recepción son independientes, lo que posibilita usar la aplicación como un adaptador de baud rate si es necesario.

Descripción de las subrutinas.

Estás subrutinas funcionan muy bien, con un Cristal de 9.8304MHz, lo que da una frecuencia de Bus de 2.4576MHz y un período T = 0.406901useg, con el que se calculan las rutinas y tablas de tiempo.

La adaptación de las rutinas para algún eventual cambio en la frecuencia del Cristal no debería generar mayores inconvenientes, puesto que se incluye toda la información sobre el cálculo mismo y los diagramas de tiempo.

La aplicación está realizada como un archivo include: SCI_TXRX.INC para que su adaptación a cualquier programa de usuario sea mucho mas simple. El archivo include consta de tres rutinas principales:

- **saca_caracter** - Rutina de transmisión sobre un pin TXD definido como salida, usa una tabla baud_tx para el seteo de la velocidad en la transmisión.
- **lee_carácter** - Rutina de recepción sobre un pin RXD (Se recomienda el Bit 0 de cualquier puerto disponible) definido como entrada, y usa una tabla baud_rx para el seteo de la velocidad de recepción.
- **delay_7a** - Rutina de tiempo, la cual es llamada por las rutinas de transmisión ó recepción.

Minimizando los tiempos en el manejo de la comunicación se pueden obtener velocidades de hasta 57600 baudios con el cristal mencionado. Dejando abierta la posibilidad de aumentar la frecuencia de cristal para lograr velocidades mayores sin demasiado esfuerzo de cambio en las líneas de código.

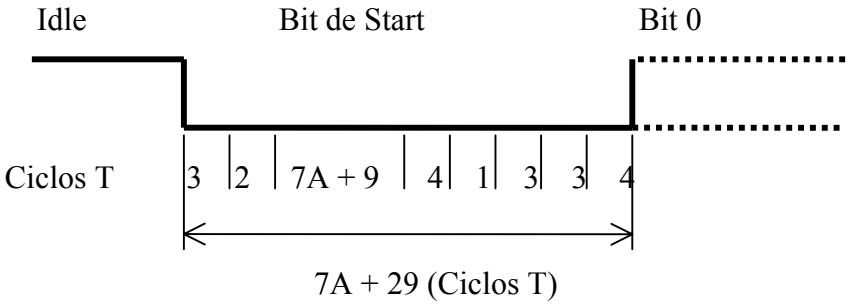
Transmisión.

La rutina de Transmisión es la mas simple puesto que una vez comenzada la transmisión y enviado el bit de inicio, solo se trata de contar tiempo de acuerdo a la tabla baud_tx y cambiar el estado lógico del pin TXD (“0” ó “1”) hasta contar los 8 bits del byte transmitido, luego se pone TXD = 1 (bit de STOP) y se espera ½ bit aproximadamente antes de buscar el siguiente byte a ser transmitido (suponiendo que ésa “búsqueda” consumirá el otro ½ bit restante antes de comenzar la transmisión del próximo byte).

El formato de transmisión es estándar NRZ n,8,1 y el orden de los bits es el siguiente:

bit START – bit 0 – bit 1 – bit 2 – bit 3 – bit 4 – bit 5 – bit 6 – bit 7 – bit STOP

El siguiente análisis de tiempo está basado en la rutina saca_caracter del archivo SCI_TXRX.INC y la unidad de medición de tiempo son los Ciclos T (0,406901 useg) de acuerdo a las condiciones especificadas anteriormente.



Este análisis permite calcular el valor de A más aproximado al ideal para cada baud rate, de lo cual sale la siguiente tabla:

BAUDIOS en TX	Ancho de bit Ideal (useg)	Valor de A hex – dec	Ciclos T (7A + 29)	Tiempo de bit (useg)	Diferencia al Ideal (useg)	Error (%)
57600	17,36	02 – 2	43T	17,49	0,13	0,74
38400	26,04	05 – 5	64T	26,04	0	0
19200	52,08	0E – 14	127T	51,67	-0,41	0,78
9600	104,2	20 – 32	253T	102,94	1,25	1,19
4800	208,3	45 – 69	512T	208,33	0,03	0,01
2400	416,7	8F – 143	1023T	416,25	-0,45	0,1

Una vez que el pin TXD subió a “1” Lógico comenzando el bit de STOP, no se puede esperar (7A + 9) Ciclos T para buscar el siguiente Byte a transmitir, como en los bits anteriores. Esto hace que luego de subir la línea TXD solo se espere una fracción del tiempo necesario y se ocupe el resto en salir de la rutina é ir a buscar el siguiente byte. Esto se hace menos problemático cuanto menor es la velocidad de transmisión.

El tiempo mínimo esperado para el bit de STOP es (7A+16) Ciclos T, pero A = baud_tx/2. Esto genera algunos inconvenientes para las siguientes velocidades:

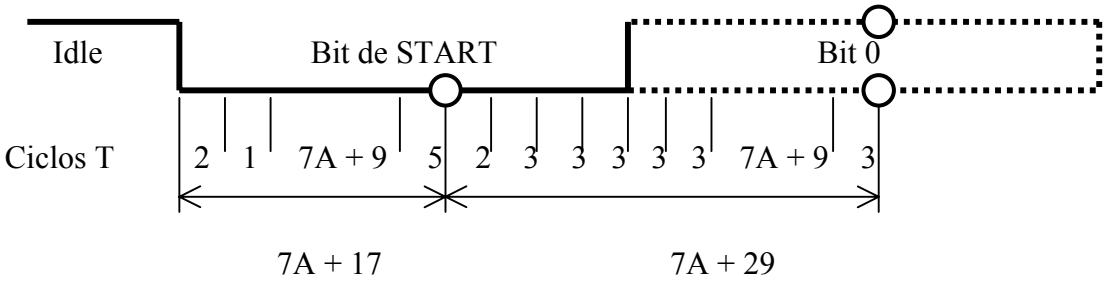
57600 baudios:

baud_tx = 2; A = 1; 23 ciclos T = 9.3587 useg; mientras que el bit Ideal = 17.36 useg. La diferencia es = 8 useg; o sea aprox 20 Ciclos T de Instrucciones para cargar el siguiente byte y volver a ingresar a la rutina de Tx. Si ahora se tiene en cuenta que este “reingreso” a la rutina saca_caracter, consume 18 Ciclos T para llegar al instante donde “bajamos“ la línea TXD para comenzar el bit de START (previo seteo variables, etc.), se observa que es casi imposible cumplir exactamente con el bit de STOP. Para superar esta dificultad se recomienda que el receptor esté seteado a 2 bits de STOP.

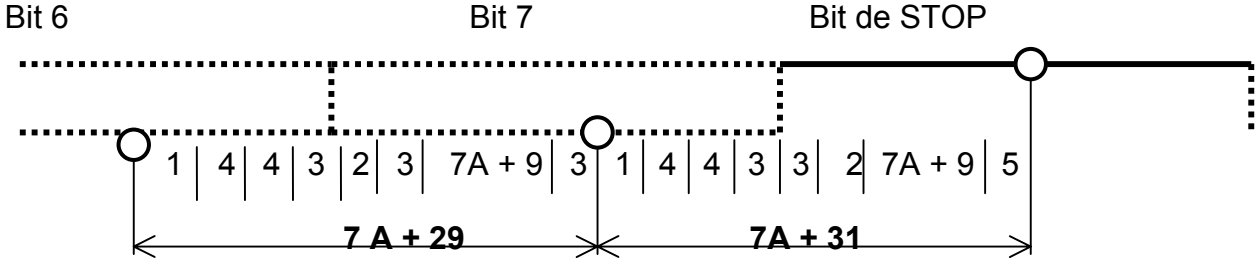
Recepción.

La rutina de Recepción funciona de la siguiente manera: Una vez que se recibió el bit de START, se espera el tiempo correspondiente a ½ bit aproximadamente (7A + 17 Ciclos T donde A = baud_rx/2) y se hace una lectura para asegurar el bit de START, a partir de aquí, se muestrea el pin RXD a intervalos de tiempo correspondiente a un bit de ancho exactamente (7A + 29 Ciclos T donde A = baud_rx) para tomar el valor del bit en la mitad (donde la probabilidad de error es menor).

El siguiente análisis de tiempo esta basado en la rutina lee_caracter del archivo SCI_TXRX.INC y la unidad de medición de tiempo son los Ciclos T (0,406901 useg) de acuerdo a las condiciones especificadas anteriormente.



Una vez ingresados los 7 bits del byte recibido, debemos esperar un “1” lógico (bit de STOP), si esto no sucediera, tendríamos un error de trama. En tal caso la rutina de recepción anula el dato recibido y lo cambia por una bandera de error, que en este caso es el signo “$”. Esto se puede cambiar fácilmente para adaptar el soft a los requerimientos del usuario.



En este último análisis de tiempo se puede observar en detalle lo que sucede mientras se recibe el bit de STOP. Una vez que este haya sido leído, solamente se tiene ½ bit de ancho para guardar el byte recibido y reingresar en la rutina de recepción. Aquí ocurre lo mismo que en la transmisión, cuanto mayor es la velocidad, mayor es el inconveniente.

Una solución es que en el último muestreo se acorte el tiempo para leer el bit de STOP. En el módulo 7A + 31 se hace A = baud_rx – 2, lo que daría un poco mas de tiempo para guardar el dato recibido. Con esta solución prácticamente no surgirían problemas hasta 38400 baudios, pero a 57600 baudios esto es imposible por que el baud_rx para esa velocidad es = 2, y no se puede entrar a la rutina delay_7a con el Acc = 0.

Para esta velocidad se debe cambiar la línea SUB #$02 por SUB #$01 ó DECA. Aun salvando este detalle, no se puede cumplir con los tiempos exactos para 57600 baudios, por lo que se recomienda que el transmisor esté seteado a 2 bits de STOP.

Aplicación.

La aplicación de estas rutinas es muy simple. Como primera medida se deben definir los pines TXD como salida, luego ponerlo a "1" lógico y RXD como entrada, Este siempre debe ser el bit 0 de cualquier puerto (para facilitar el software). Luego se definen las variables **caracter**, **cuenta_bit** y **auxiliar** de 1 byte cada una, haciendo reserva de la memoria ram para ellas. Por último se debe incluir el archivo **SCI_TXRX.INC** dentro del programa realizado, con la siguiente línea:

$INCLUDE "SCI_TXRX.INC" ; Inclusión del archivo de comunicación Serie

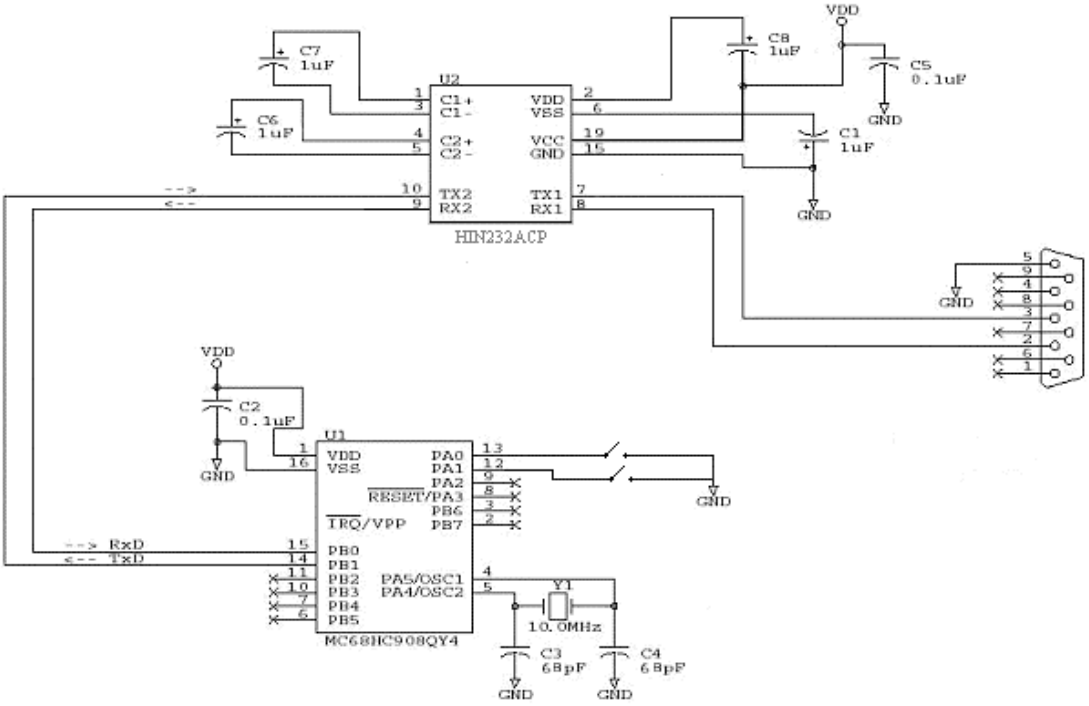
Ahora solo se debe cargar el dato a transmitir en la variable caracter y llamar a la rutina saca_caracter el dato será transmitido a la velocidad seteada. Para la recepción, se debe llamar a la rutina **lee_caracter** y esperar a que salga, una vez que esto suceda, se tiene el dato recibido en la variable caracter. Si el dato recibido es "$" ($24) ha ocurrido un error de trama.

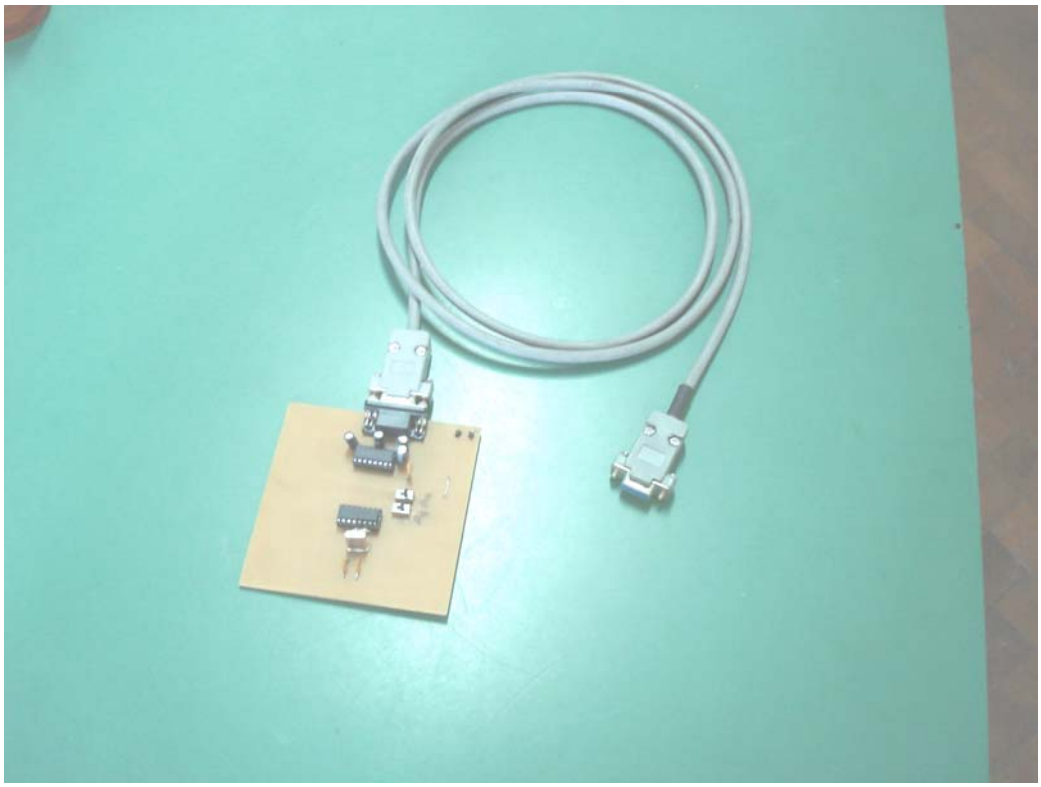
Diagrama Eléctrico.

El diagrama eléctrico está formado por un conector puerto serie, para realizar la conexión a la PC, un integrado RS-232 (HIN232ACP) y el microcontrolador HC908QY4A [6] y [7]. Este diagrama eléctrico contiene 2 llaves, una conectada al puerto A0 y la otra conectada al puerto A1. Estás llaves se utilizan para seleccionar el tipo de operación a realizar en punto flotante como muestra la siguiente tabla:

Operaciones	Puerto A0	Puerto A1
Suma	0	0
Resta	1	0
Multiplicación	0	1
División	1	1

El funcionamiento del microcontrolador se configura con un cristal de 10 MHz, porque le da mas precisión al clock de bus, lo que permite hacer una transmisión con poca probabilidad de error.

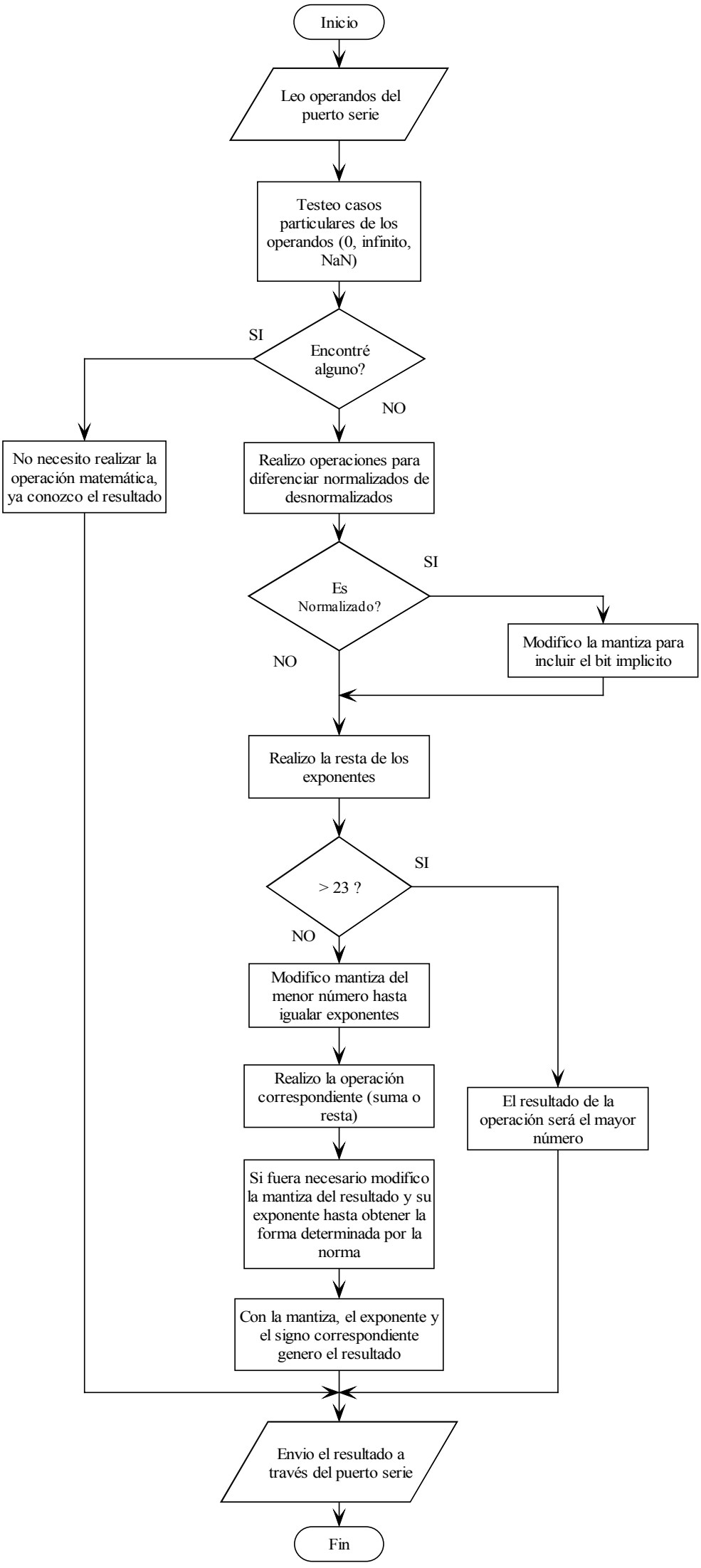




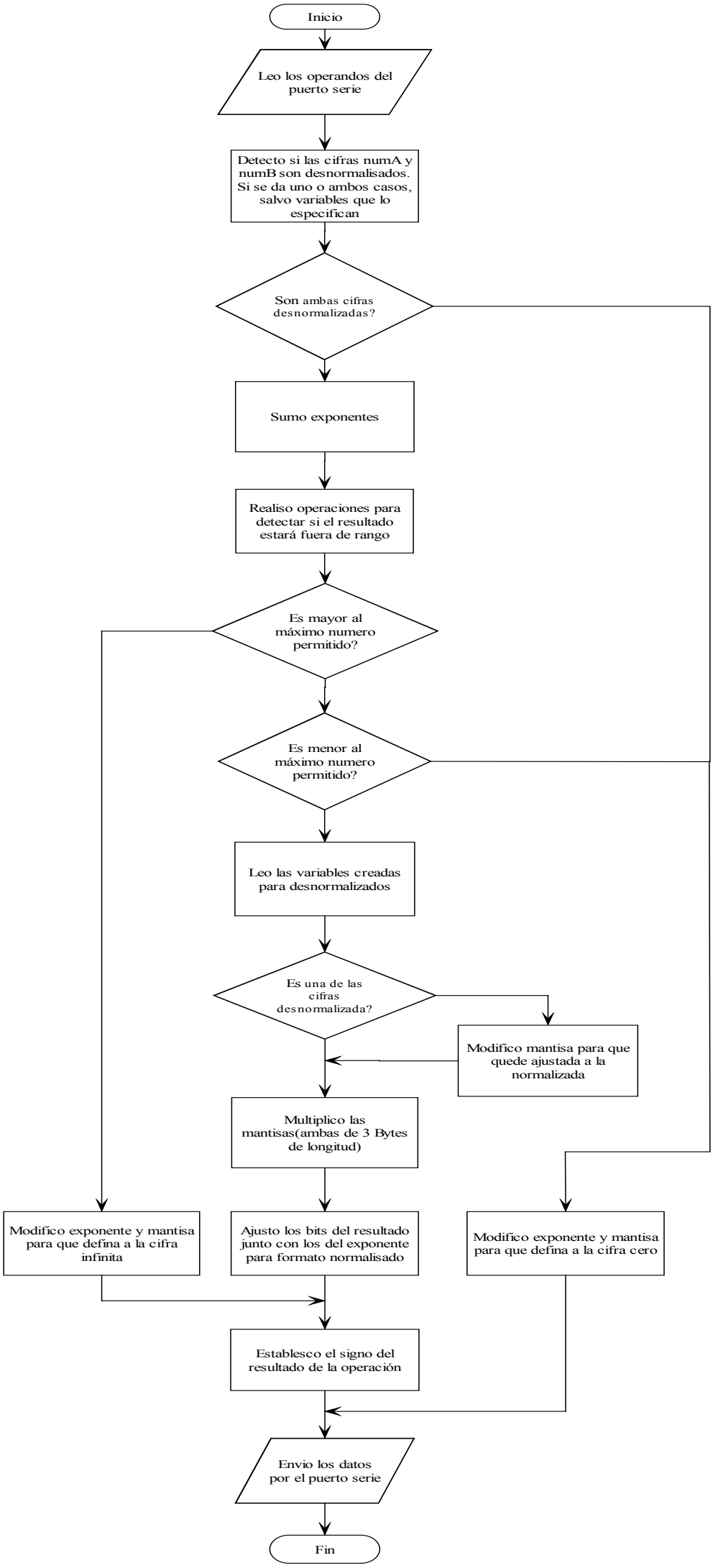
Diagramas de Flujo.

En está sección se encuentran los diagramas de flujos de la suma y resta, multiplicación, división y del algoritmo de comunicación SCI [8] para los microcontroladores que no poseen modulo SCI.

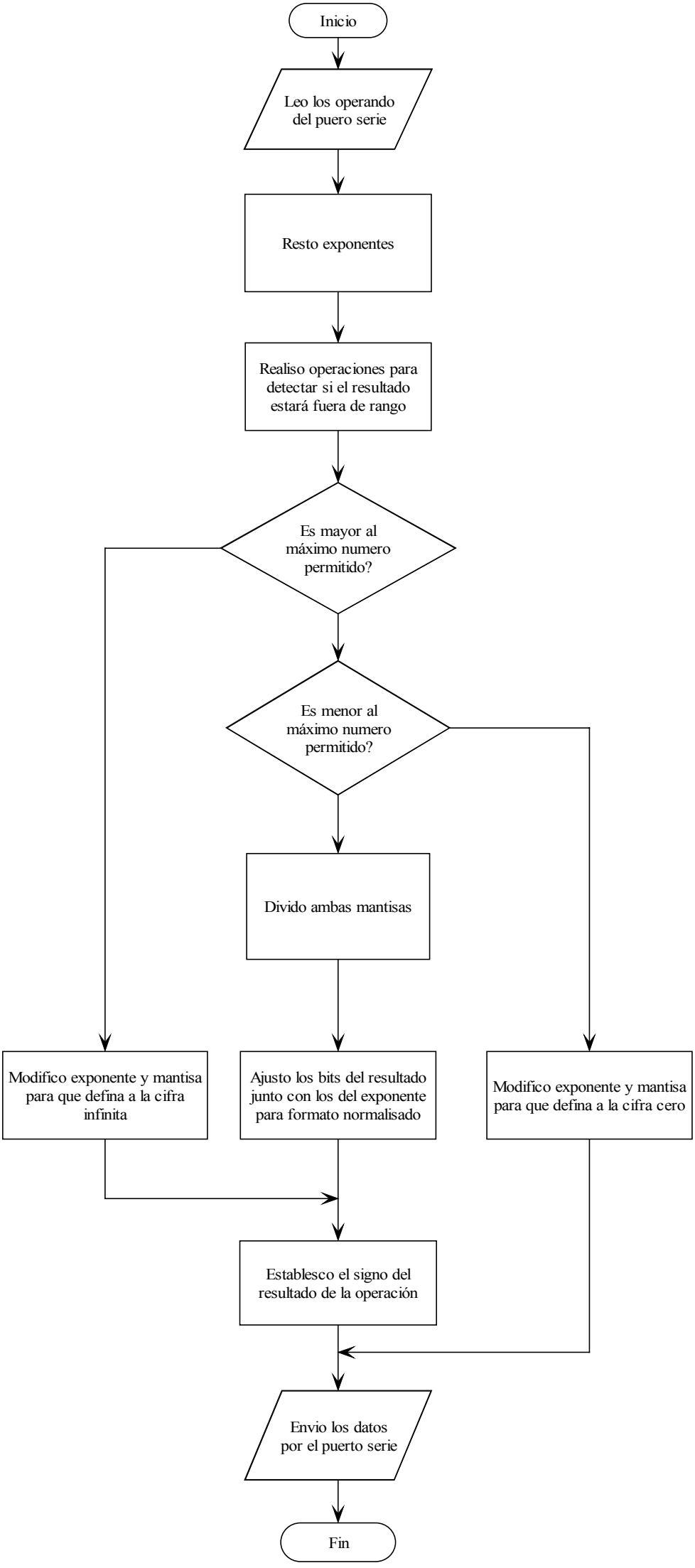
Suma y Resta


Inicio
Leo operandos del puerto serie
Testeo casos particulares de los operandos (0, infinito, NaN)
Encontré alguno?
SI
NO
No necesito realizar la operación matemática, ya conozco el resultado
Realizo operaciones para diferenciar normalizados de desnormalizados
Es Normalizado?
SI
NO
Modifico la mantiza para incluir el bit implicito
Realizo la resta de los exponentes
> 23 ?
SI
NO
Modifico mantiza del menor número hasta igualar exponentes
Realizo la operación correspondiente (suma o resta)
El resultado de la operación será el mayor número
Si fuera necesario modifico la mantiza del resultado y su exponente hasta obtener la forma determinada por la norma
Con la mantiza, el exponente y el signo correspondiente genero el resultado
Envio el resultado a través del puerto serie
Fin

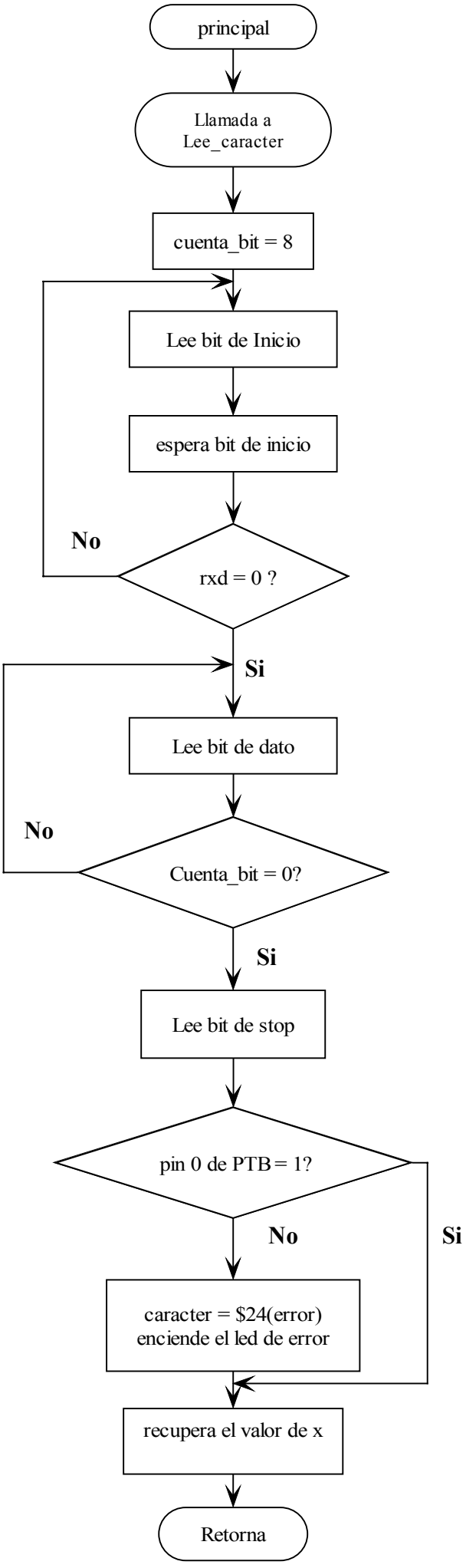
Multiplicación



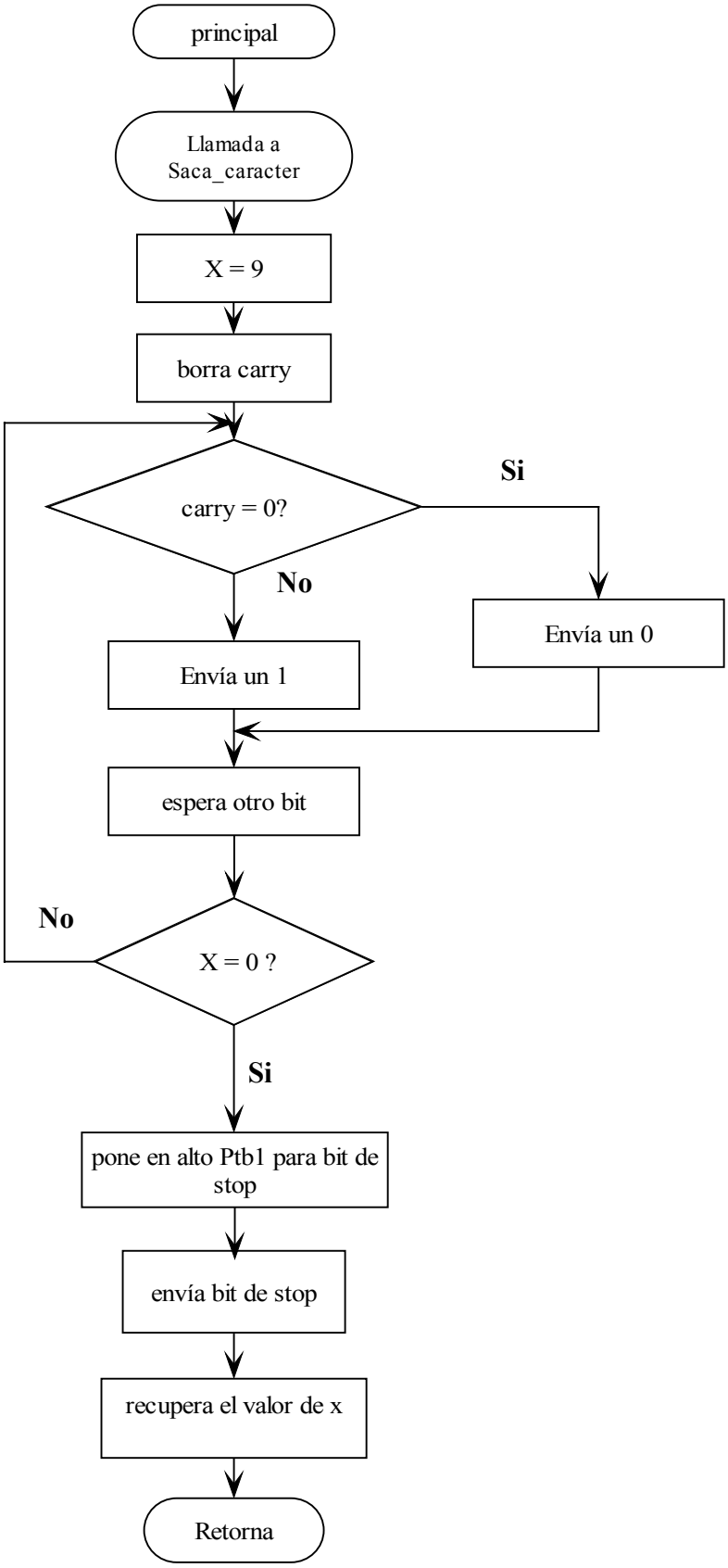
División


Inicio
Leo los operando del puero serie
Resto exponentes
Realiso operaciones para detectar si el resultado estará fuera de rango
Es mayor al máximo numero permitido?
Es menor al máximo numero permitido?
Divido ambas mantisas
Modifico exponente y mantisa para que defina a la cifra infinita
Ajusto los bits del resultado junto con los del exponente para formato normalisado
Modifico exponente y mantisa para que defina a la cifra cero
Establesco el signo del resultado de la operación
Envio los datos por el puerto serie
Fin

SCI - Lee Dato de 8 bits



SCI - Envía Dato de 8 bits



Continuará .....

Nota de Radacción: El lector puede descargar este artículo y artículos anteriores de ***"Buceando..."*** desde la sección ***"Artículos Técnicos"*** en el sitio web de **EduDevices (www.edudevices.com.ar)**