GRUPO z170238

# Proyecto E/S por Interrupciones

Arquitectura de Computadores 2023-2024

Eduardo Gil Alba
8-5-2024

# TABLA DE CONTENIDO

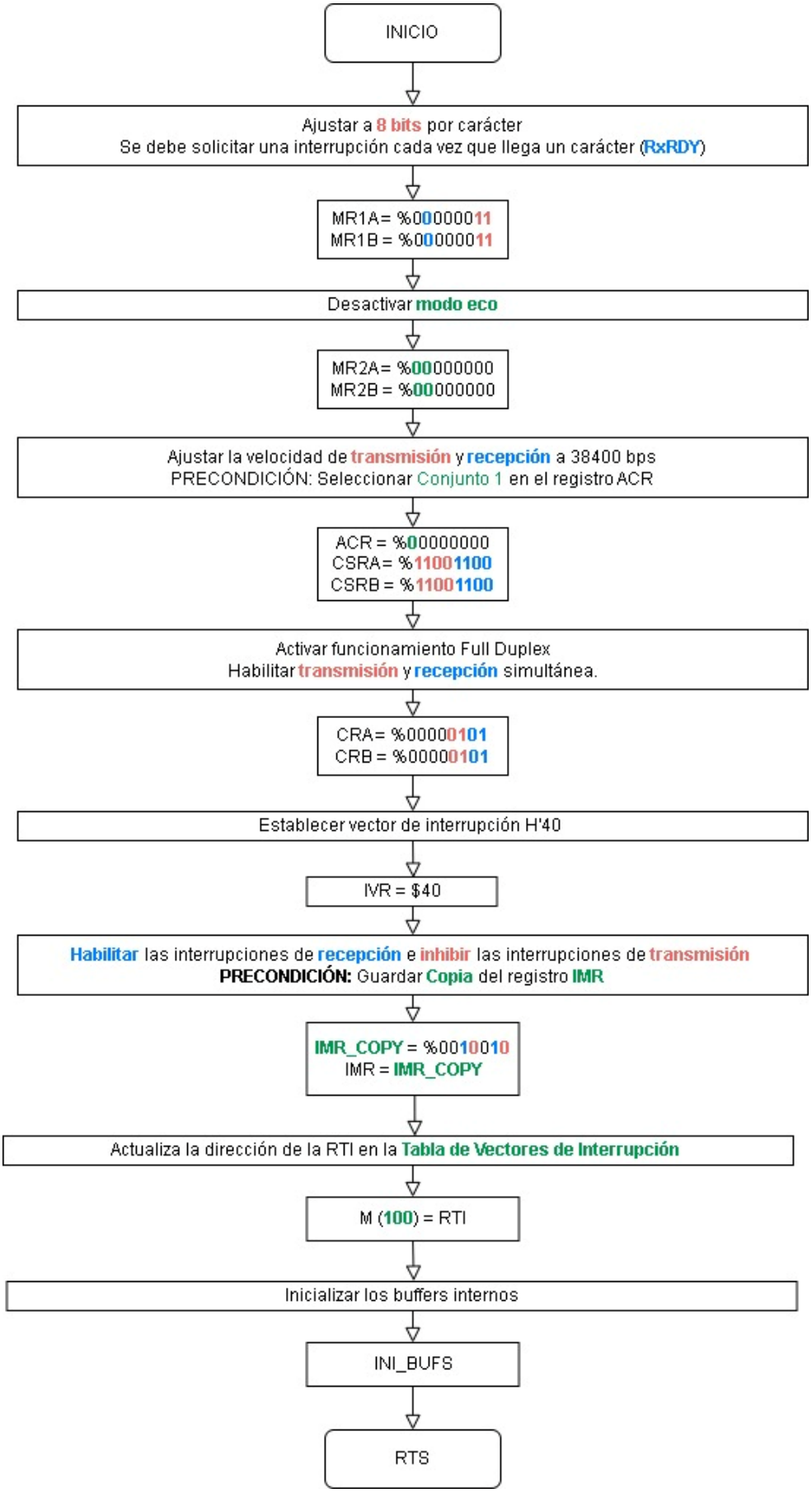
# Subrutina INIT

## Listado comentado

| | |
|---|---|
| Descripción | Esta subrutina se encarga de inicializar los registros del controlador DUART y los buffers internos dónde se almacenarán los caracteres a recibir/transmitir. |
| Parámetros | No tiene parámetros. |
| Algoritmo | 1. Inicializa el registro de modo 1 para establecer el nº de bits por carácter en cada línea y cuando ha de solicitar una interrupción en la recepción.<br>2. En el siguiente acceso al registro de modo se accede al de modo 2 para establecer el modo de la línea.<br>3. Inicializa los registros de control auxiliar y de selección de reloj para seleccionar el conjunto y establecer la velocidad de recepción y transmisión de cada línea.<br>4. Inicializa los registros de control para habilitar la recepción y transmisión simultánea en cada línea.<br>5. Inicializa el registro del vector de interrupción.<br>6. Antes de modificar la máscara de interrupción, para habilitar las interrupciones de recepción e inhibir las de transmisión, por conveniencia[1], se reserva una copia en memoria del registro.<br>7. Actualiza la dirección de RTI en la primera entrada de la Tabla de Vectores de Interrupción, la dirección resultante de multiplicar por 4 el contenido del registro del vector de interrupción.<br>8. Llama a la subrutina INI_BUFS que no tiene parámetros para inicializar los buffers internos.<br>9. Finalmente, hace un retorno de subrutina sin dejar ningún valor representativo en los registros del computador. |
| Dependencias | La subrutina hace una llamada a la subrutina INI_BUFS que se proporciona en el fichero bib_aux.s |
| Fecha de Creación | 16/02/2024 |
| Fecha de Modificación | 05/03/2024 |

[1]. Este registro solo se puede acceder a él en escritura y me interesa poder acceder en lectura.

# Diagrama de flujo

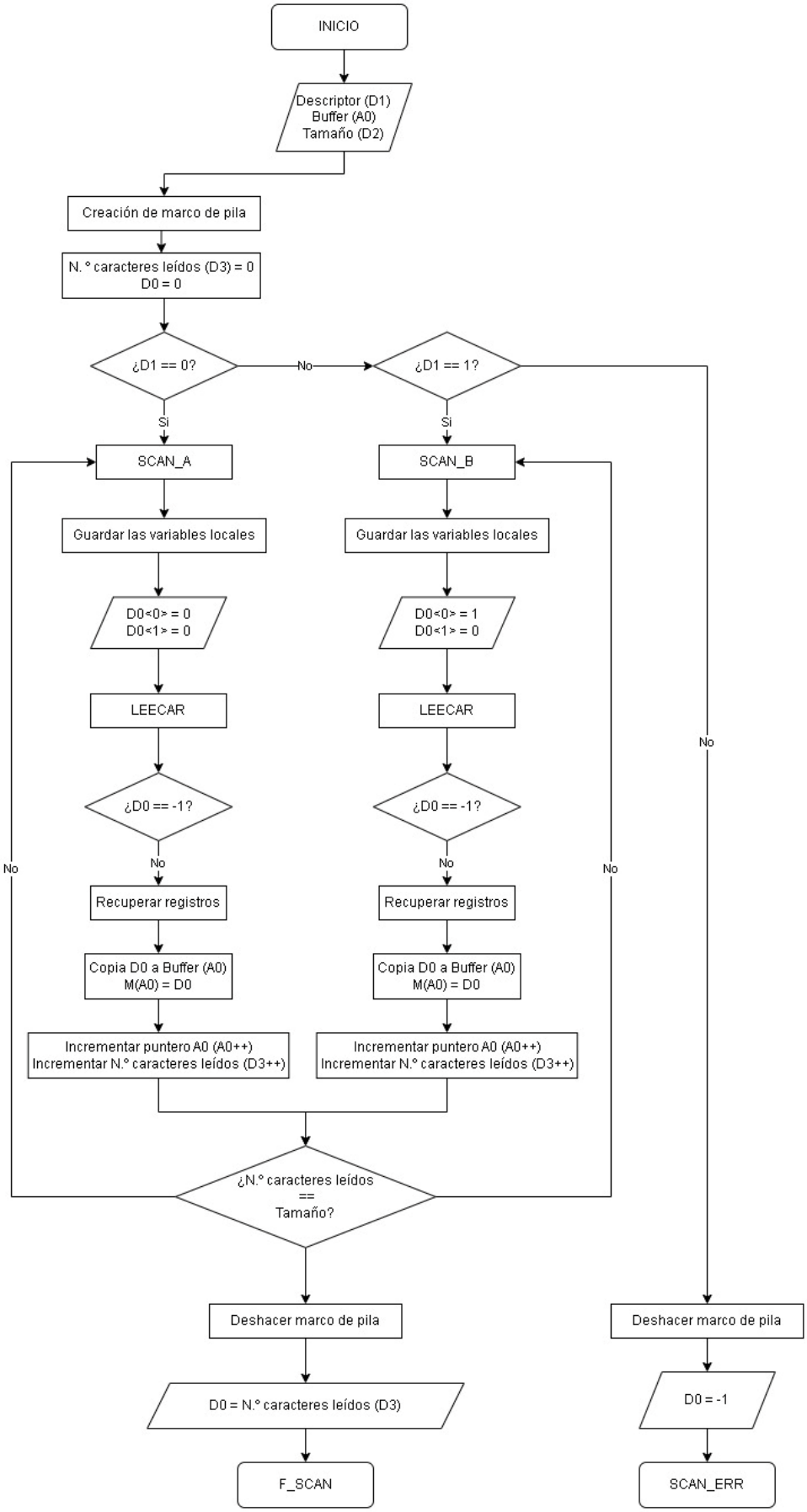


# Subrutina SCAN

## Listado comentado

| | |
|---|---|
| Descripción | Esta subrutina realiza la lectura de un bloque de caracteres de la línea correspondiente. Copia en el parámetro Buffer los Tamaño primeros caracteres almacenados en el buffer interno correspondiente y los elimina de éste. |
| Parámetros | Se pasan por pila:<br>• De salida:<br>– Buffer (4 Bytes) por Dirección<br>• De entrada:<br>– Descriptor (2 Bytes) por Valor<br>– Tamaño (2 Bytes) por Valor |
| Algoritmo | 1. Se crea un marco de pila donde se almacenarán las variables locales, el registro D3 y el puntero A0.<br>2. Se inicializan los registros de la siguiente manera: A0 se carga con la dirección del buffer; D1 con el descriptor; D2 con el tamaño; D3 se establece en 0 para contabilizar el número de caracteres leídos, y D0 se inicializa en 0, posteriormente se utilizará para indicar la salida de la subrutina.<br>3. Se realiza una evaluación del descriptor para determinar la línea desde la cual se leerá del buffer interno. Si el descriptor no corresponde a la línea A o línea B, se deshace el marco de pila y se devuelve -1 en el registro D0.<br>4. Para cada línea, se prepara la llamada a la función LEECAR. Primero se guardan las variables locales y luego se configura D0, donde el bit 0 representa la línea y el bit 1, en este caso, corresponde al buffer interno de recepción.<br>5. Si después de la llamada se obtiene -1 en D0, indica que el buffer está vacío. En este caso, se deshace el marco de pila y se devuelve el número de caracteres leídos hasta el momento (D3) en el registro D0.<br>6. En caso de que el buffer no esté vacío, se copia el carácter extraído en D0 al puntero del buffer (A0).<br>7. Luego se incrementa el puntero A0 y el contador de caracteres leídos D3.<br>8. Se evalúa si se han leído el número de caracteres especificado en el tamaño (D2). Si se ha alcanzado este límite, se deshace el marco de pila y se devuelve el número de caracteres leídos hasta el momento (D3) en D0.<br>9. Si aún no se han leído todos los caracteres especificados, se prepara nuevamente la llamada a LEECAR hasta que el buffer esté vacío o se hayan leído todos los caracteres especificados en el tamaño. |
| Dependencias | La subrutina hace una llamada a la subrutina LEECAR que se proporciona en el fichero bib_aux.s |
| Fecha de Creación | 19/02/2024 |
| Fecha de Modificación | 14/03/2024 |

# Diagrama de flujo

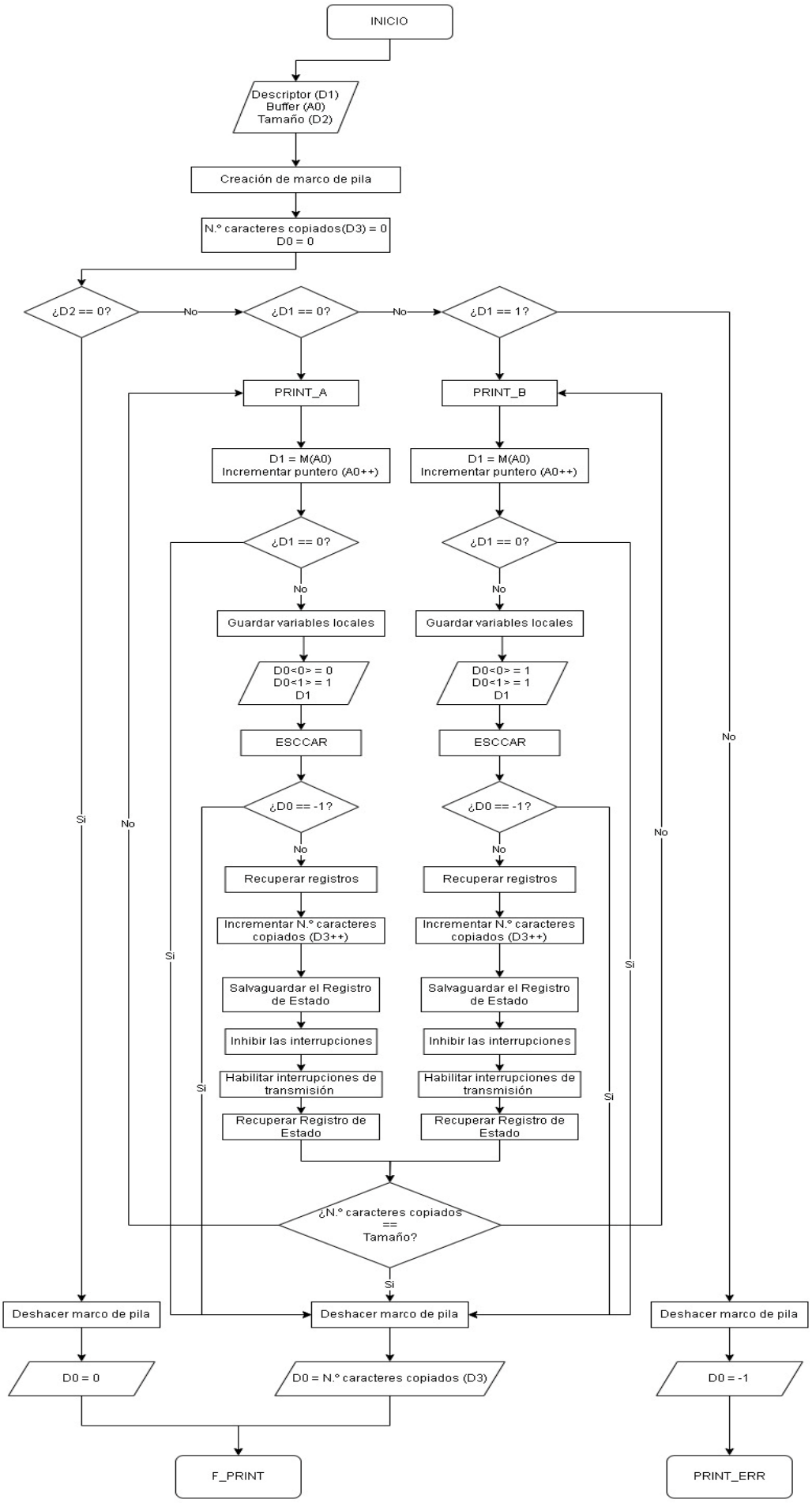


# Subrutina PRINT

## Listado comentado

| | |
|---|---|
| Descripción | Esta subrutina realiza la escritura de un bloque de caracteres de la línea correspondiente. Copia en el buffer interno correspondiente los Tamaño primeros caracteres almacenados en el parámetro Buffer y los elimina del buffer interno. |
| Parámetros | Se pasan por pila:<br>• De entrada:<br>– Buffer (4 Bytes) por Dirección<br>– Descriptor (2 Bytes) por Valor<br>– Tamaño (2 Bytes) por Valor |
| Algoritmo | 1. Se crea un marco de pila donde se almacenarán las variables locales, el registro D3 y el puntero A0.<br>2. Se inicializan los registros de la siguiente manera: A0 se carga con la dirección del buffer; D1 con el descriptor; D2 con el tamaño; D3 se establece en 0 para contabilizar el número de caracteres copiados, y D0 se inicializa en 0, posteriormente se utilizará para indicar la salida de la subrutina.<br>3. Se realiza una comprobación del Tamaño. Si su valor es 0, se termina la ejecución de la subrutina. En caso contrario, se continúa.<br>4. Se realiza una evaluación del descriptor para determinar la línea desde la cual se escribirá al buffer interno. Si el descriptor no corresponde a la línea A o línea B, se deshace el marco de pila y se devuelve -1 en el registro D0.<br>5. Para cada línea, se prepara la llamada a la función ESCCAR. En primer lugar, se configura D0, donde el bit 0 representa la línea y el bit 1, en este caso, corresponde al buffer interno de transmisión. En segundo lugar, se copia el carácter apuntado por A0 al registro D1 e incrementa el puntero. Se hace una comprobación del carácter en D1 para evitar que se copie un carácter nulo. Por último, se guardan las variables locales.<br>6. Si después de la llamada se obtiene -1 en D0, indica que el buffer está lleno. En este caso, se deshace el marco de pila y se devuelve el número de caracteres copiados hasta el momento (D3) en el registro D0.<br>7. En caso contrario, se recuperan las variables locales, se incrementa el número de caracteres copiados y se avisa al controlador que existe un carácter para transmitir. Para ello, se salva el registro de estado, se enmascaran todas las interrupciones para entrar en exclusión mutua, se habilita la transmisión en el bit TxRDYA o TxRDYB del registro IMR y finalmente, se recupera el registro de estado anterior.<br>8. Se evalúa si se han copiado el número de caracteres especificado en el tamaño (D2). Si se ha alcanzado este límite, se deshace el marco de pila y se devuelve el número de caracteres copiados hasta el momento (D3) en D0.<br>9. Si aún no se han copiado todos los caracteres especificados, se prepara nuevamente la llamada a ESCCAR hasta que el buffer esté lleno o se hayan copiado todos los caracteres especificados en el tamaño. |
| Dependencias | La subrutina hace una llamada a la subrutina ESCCAR que se proporciona en el fichero bib_aux.s |
| Fecha de Creación | 19/02/2024 |
| Fecha de Modificación | 14/03/2024 |

# Diagrama de flujo

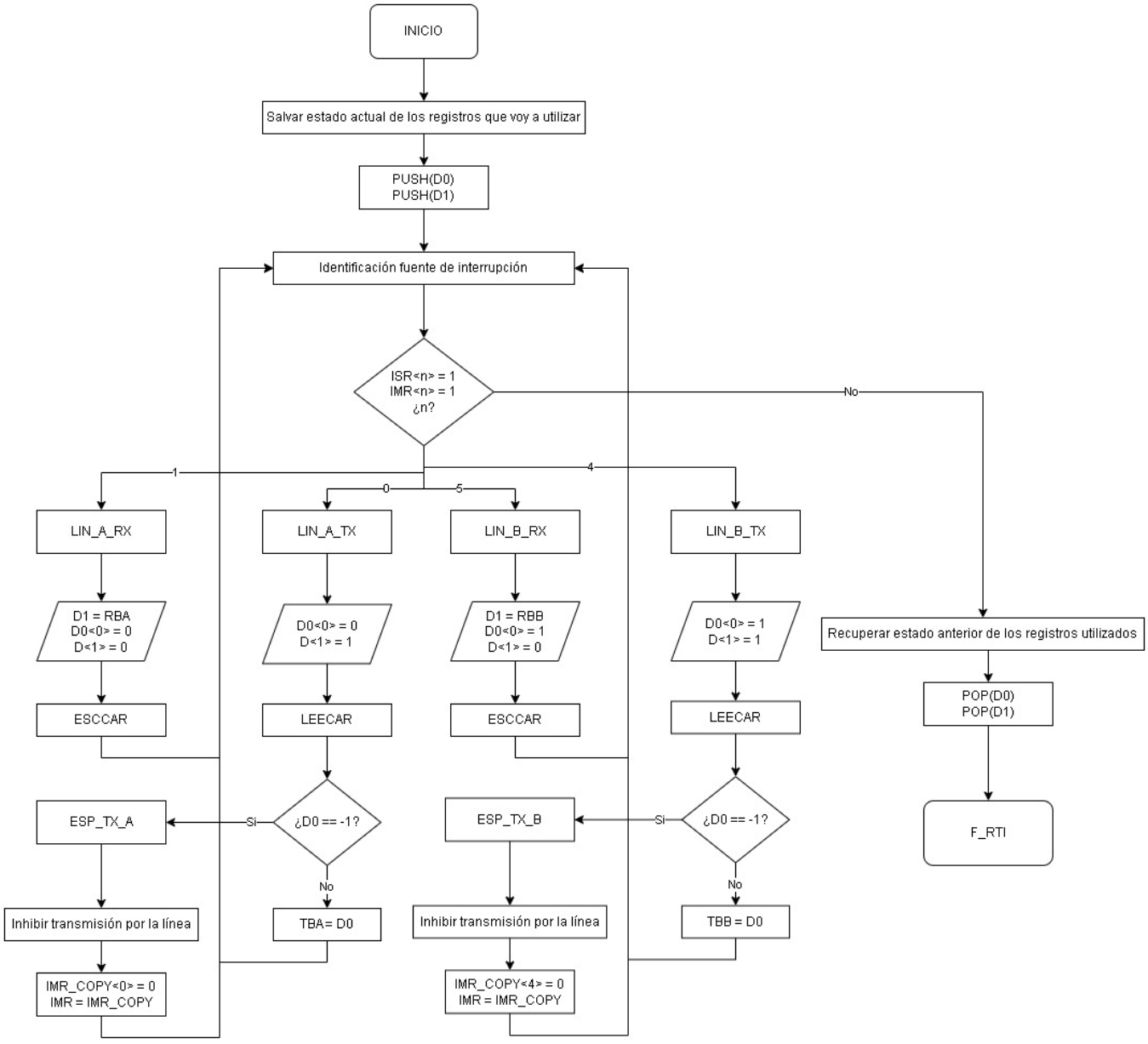


# Subrutina RTI

## Listado comentado

| | |
|---|---|
| Descripción | Esta subrutina se encarga del tratamiento de una interrupción como resultado de la ejecución de secuencia de reconocimiento de interrupciones. |
| Parámetros | No tiene parámetros. |
| Algoritmo | 1. Guarda los registros D0 y D1, los cuales se utilizarán para las llamadas a las subrutinas ESCCAR o LEECAR.<br>2. Inicia el proceso de identificación de la fuente de interrupción. Si algún bit en el registro ISR está activo (valor 1) y el mismo bit en el registro IMR está habilitado (valor 1), entonces, si corresponde a una interrupción por recepción, indica que se ha recibido un carácter y que la línea correspondiente está habilitada. Si corresponde a una interrupción por transmisión, esto señala que el buffer de transmisión está vacío y la línea está habilitada.<br>3. Una vez identificado el bit activo, en el caso de una interrupción por recepción, se prepara la llamada a ESCCAR para transferir el carácter del registro del buffer de recepción al buffer interno de recepción de la línea correspondiente.<br>4. En el caso de una interrupción por transmisión, se prepara la llamada a LEECAR para leer del buffer interno de transmisión de la línea correspondiente el carácter que luego se guardará en el buffer de transmisión.<br>5. Existen casos especiales que deben ser considerados. En el caso de una interrupción por recepción, si el buffer interno de recepción está lleno, se lee el carácter, pero no se inserta en el buffer interno de recepción. En el caso de una interrupción por transmisión, como el buffer interno de transmisión aún no ha sido leído hasta después de la llamada a LEECAR, se debe deshabilitar la transmisión por la línea.<br>6. Se continúa el proceso de identificación de la fuente de interrupción hasta que todas las interrupciones pendientes sean atendidas en esa llamada. Si no hay más interrupciones pendientes, se restaura el estado anterior de los registros utilizados y se sale de la subrutina. |
| Dependencias | La subrutina hace una llamada a la subrutina LEECAR y ESCCAR que se proporciona en el fichero bib_aux.s |
| Fecha de Creación | 05/03/2024 |
| Fecha de Modificación | 14/03/2024 |

# Diagrama de flujo



# Juego de Ensayo

Para realizar las pruebas, he utilizado de base el ejemplo de programa de prueba de la pág. 75 del Manual, he modificado las llamadas para probar ambas líneas, recibir por una línea y transmitir por la otra, el número de caracteres a recibir / transmitir y el tamaño de cada operación PRINT y SCAN. Con el depurador, he seguido la ejecución del programa, por ejemplo, frente a casos especiales, en el que el buffer interno se encuentra lleno o vacío y comprobar cómo mis subrutinas se comportan.

```
BUFFER:   DS.B  3000    * Buffer para lectura y escritura de caracteres
PARDIR:   DC.L  0       * Dirección que se pasa como parámetro
PARTAM:   DC.W  0       * Tamaño que se pasa como parámetro
CONTC:    DC.W  0       * Contador de caracteres a imprimir
DESA:     EQU   0       * Descriptor línea A
DESB:     EQU   1       * Descriptor línea B
TAMBS:    EQU   10      * Tamaño de bloque para SCAN
TAMBP:    EQU   3000    * Tamaño de bloque para PRINT

        * 3000 BYTES en bloques de 10B recibidos por línea A  e imprimir por línea B

MAIN:           MOVE.L #BUS_ERROR,8          *Install bus error handler
                MOVE.L #ADDRESS_ER,12        *Install address error handler
                MOVE.L #ILLEGAL_IN,16        *Install illegal instruction handler
                MOVE.L #PRIV_VIOLT,32        *Install privilege violation handler
                MOVE.L #ILLEGAL_IN,40        *Install illegal instruction handler
                MOVE.L #ILLEGAL_IN,44        *Install illegal instruction handler

                BSR INIT
                MOVE.W #$2000,SR        * Permite interrupciones

BUCPR:          MOVE.W #TAMBS,PARTAM         * Inicializa parámetro de tamaño
                MOVE.L #BUFFER,PARDIR        * Parámetro BUFFER = comienzo del buffer

OTRAL:          MOVE.W PARTAM,-(A7)          * Tamaño de bloque
                MOVE.W #DESA,-(A7)      * Puerto A
                MOVE.L PARDIR,-(A7)     * Dirección de lectura

ESPL:           BSR SCAN
                ADD.L #8,A7             * Restablece la pila

                ADD.L D0,PARDIR         * Calcula la nueva dirección de lectura
                SUB.W D0,PARTAM         * Actualiza el número de caracteres leídos
                ADD.W D0,CONTC
                CMP.W #TAMBP,CONTC
                BNE OTRAL

                MOVE.W #TAMBS,CONTC          * Inicializa contador de caracteres a imprimir
                MOVE.L #BUFFER,PARDIR        * Parámetro BUFFER = comienzo del buffer

OTRAE:          MOVE.W #TAMBP,PARTAM         * Tamaño de escritura = Tamaño de bloque

ESPE:           MOVE.W PARTAM,-(A7)          * Tamaño de escritura
                MOVE.W #DESB,-(A7)      * Puerto B
                MOVE.L PARDIR,-(A7)     * Dirección de escritura
```

```
                BSR PRINT
                ADD.L #8,A7             * Restablece la pila

                ADD.L D0,PARDIR         * Calcula la nueva dirección del buffer
                SUB.W D0,CONTC          * Actualiza el contador de caracteres
                BEQ SALIR               * Si no quedan caracteres se acaba

                SUB.W D0,PARTAM         * Actualiza el tamaño de escritura
                BNE ESPE                * Si no se ha escrito todo el bloque se insiste
                CMP.W #TAMBP,CONTC      * Si el no de caracteres que quedan es menor que
                                        * el tamaño establecido se imprime ese número
                BHI OTRAE               * Siguiente bloque
                MOVE.W CONTC,PARTAM
                BRA ESPE                *Siguiente bloque

SALIR:          BRA BUCPR

BUS_ERROR:      BREAK
                NOP

ADDRESS_ER:     BREAK
                NOP

ILLEGAL_IN:     BREAK
                NOP

PRIV_VIOLT:     BREAK
                NOP
```

# Observaciones finales

La realización de esta práctica me ha tomado aproximadamente un mes. Tuve mayores dificultades al implementar las subrutinas PRINT y RTI. Inicialmente, en mi implementación de PRINT, habilitaba las interrupciones de transmisión una vez que había copiado todos los caracteres del parámetro Tamaño del buffer interno, o los disponibles en ese momento, hasta que el buffer quedara vacío. Esta estrategia, que funcionaba correctamente en pruebas locales, resultó problemática durante las pruebas con el tester, generando una excepción BUS ERROR y conduciendo al programa a un bucle infinito.

Tras revisar exhaustivamente tanto el enunciado como el manual, comprendí que la transmisión debía efectuarse a nivel de carácter. Esto implicaba que las interrupciones de transmisión solo debían ser habilitadas cuando se confirmara la presencia de un nuevo carácter en el buffer de transmisión.

En comparación con mis experiencias previas con el Motorola 88110, encontré más facilidad al programar en este nuevo microprocesador, gracias a su amplio conjunto de instrucciones. Para expresar lo mismo he utilizado menos instrucciones y en el proceso de depuración del programa se torna más comprensible.