

# INTERRUPCIONES EN LOS MICROCONTROLADORES

Clase virtual para la asignatura  
MICROCOMPUTADORAS  
Profesor: Antonio Salvá Calleja  
Abril de 2020

# INTERRUPCIONES EN LOS MICROCONTROLADORES

En lo fundamental, las interrupciones en los microcontroladores pueden ser de dos tipos :

Interrupciones de hardware

Interrupciones de software

**Las interrupciones de hardware.** Consisten en el cambio en la secuencia de ejecución de un programa, esto cuando se da un determinado evento de hardware, para pasar a ejecutar una rutina asociada con el evento de hardware implicado, a ésta frecuentemente se le denomina “***rutina de servicio***”. ***Una vez que concluye la ejecución de la rutina de servicio, el microcontrolador deberá retornar a la secuencia de ejecución que fue interrumpida.***

Para un determinado MCU, eventos de hardware podrán ser entre otros: **que el puerto serie asíncrono haya recibido un byte y éste esté listo para ser leído, que se de un flanco de bajada en un determinado bit de puerto, etc.**

En esta exposición, a los diversos eventos de hardware que pueden hacer que se de una interrupción, se les denominará como **INSTANCIAS DE INTERRUPCIÓN** y **cuando éstas llegan a presentarse, se dice que se tiene un REQUERIMIENTO DE INTERRUPCIÓN (RI)**.

# INTERRUPCIONES EN LOS MICROCONTROLADORES

En lo fundamental, las interrupciones en los microcontroladores pueden ser de dos tipos :

Interrupciones de hardware

Interrupciones de software

**Las interrupciones de software.** Se invocan no porque se de un determinado evento de hardware, sino porque se coloque en el código del programa una instrucción que invoque a la rutina de servicio asociada, siguiendo el MCU los mismos pasos que se llevan a cabo cuando suceden interrupciones de hardware. Por ejemplo, para el MCU MC9S08SH32, solo existe una interrupción de software, y ésta se da cuando se ejecuta la instrucción SWI. En esta exposición se tratará solo con interrupciones de hardware.

# INTERRUPCIONES EN EL MCU MC9S08SH32

## CONCEPTOS GENERALES

Todos los microcontroladores de la familia HCS08 siguen el mismo patrón de respuesta a los diversos requerimientos de interrupción que se les pudieran presentar, lo que varía de uno a otro miembro de la familia es, en esencia, el número de instancias de interrupción soportables.

El MCU MC9S08SH32 soporta 19 instancias de interrupción de hardware y una sola interrupción de software.

En la tabla 5-2, presente en la página 64 del documento MC9S08SH32.pdf, se muestra información relevante acerca de cada una de las instancias de interrupción soportadas por el MCU MC9S08SH32. Ahí se aprecia que cada una de ellas tiene un número. Por ejemplo, la instancia de interrupción 23 se denomina como **Vadc y el evento de hardware asociado es que el convertidor analógico digital haya completado una conversión, y el dato binario generado esté listo para ser leído.**

# INTERRUPCIONES EN EL MCU MC9S08SH32

## PRIORIDADES

Si se llegaran a presentar simultáneamente dos o más requerimientos de interrupción, el MCU atiende primero el requerimiento que tenga asociado el número más pequeño, para después proseguir a atender los subsecuentes requerimientos en orden ascendente de sus números asociados. Por ejemplo, si se dan simultáneamente las instancias de interrupción digamos 23 y 11, el MCU ejecuta primero la rutina de servicio asociada con la instancia 11, y una vez que retorna de ésta, pasa a ejecutar la rutina de servicio asociada con la instancia 23, y después procede a seguir ejecutando el programa que fue interrumpido.

# INTERRUPCIONES EN EL MCU MC9S08SH32

## HABILITACIÓN GLOBAL DE INTERRUPCIONES

El que el MCU responda a los diversos requerimientos de interrupción está gobernado por el bit “I” presente en el registro de banderas (CCR) y denominado “mascara global de interrupciones”. Si éste es 1, el MCU no responde a los requerimientos de interrupción que se pudieran presentar; por otra parte si el bit “I” es cero, el MCU puede responder a los requerimientos de interrupción que se pudieran llegar a presentar.

# INTERRUPCIONES EN EL MCU MC9S08SH32

## HABILITACIÓN GLOBAL DE INTERRUPCIONES

El que el MCU responda a los diversos requerimientos de interrupción está gobernado por el bit “I” presente en el registro de banderas (CCR) y denominado **“mascara global de interrupciones”**. Si éste es 1, el MCU no responde a los requerimientos de interrupción que se pudieran presentar; por otra parte si el bit “I” es cero, el MCU puede responder a los requerimientos de interrupción que se pudieran llegar a presentar.

Al reset, el bit “I” es inicializado en uno lógico; por lo tanto, por defecto el MCU no responde a requerimientos de interrupción.

# INTERRUPCIONES EN EL MCU MC9S08SH32

## HABILITACIÓN GLOBAL DE INTERRUPCIONES

El que el MCU responda a los diversos requerimientos de interrupción está gobernado por el bit “I” presente en el registro de banderas (CCR) y denominado “mascara global de interrupciones”. Si éste es 1, el MCU no responde a los requerimientos de interrupción que se pudieran presentar; por otra parte si el bit “I” es cero, el MCU puede responder a los requerimientos de interrupción que se pudieran llegar a presentar.

Al reset, el bit “I” es inicializado en uno lógico; por lo tanto, por defecto el MCU no responde a requerimientos de interrupción.

A grandes rasgos, lo que sucede es que si el bit “I” es cero lógico, cada vez que el MCU termina la ejecución de una instrucción, checa si hay algún requerimiento de interrupción, si es el caso, procede a llevar a cabo una serie de pasos que conducen a que se pase a ejecutar la rutina de servicio asociada. Si no hay requerimiento de interrupción, el MCU procede a la ejecución de la instrucción subsecuente en el programa. Por otra parte, si el bit “I” es uno lógico, cada vez que se termina la ejecución de una instrucción, el MCU no checa si hay un requerimiento de interrupción y procede a la ejecución de la instrucción siguiente presente en el programa.



# INTERRUPCIONES EN EL MCU MC9S08SH32

## HABILITACIÓN GLOBAL DE INTERRUPCIONES

El que el MCU responda a los diversos requerimientos de interrupción está gobernado por el bit “I” presente en el registro de banderas (CCR) y denominado **“mascara global de interrupciones”**. Si éste es 1, el MCU no responde a los requerimientos de interrupción que se pudieran presentar; por otra parte si el bit “I” es cero, el MCU puede responder a los requerimientos de interrupción que se pudieran llegar a presentar.

Al reset, el bit “I” es inicializado en uno lógico; por lo tanto, por defecto el MCU no responde a requerimientos de interrupción.

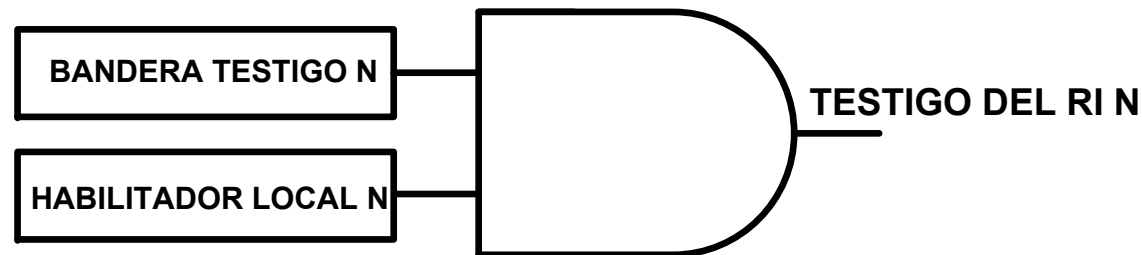
**A grandes rasgos, lo que sucede es que si el bit “I” es cero lógico, cada vez que el MCU termina la ejecución de una instrucción, checha si hay algún requerimiento de interrupción, si es el caso, procede a llevar a cabo una serie de pasos que conducen a que se pase a ejecutar la rutina de servicio asociada, si no hay requerimiento de interrupción, el MCU procede a la ejecución de la instrucción subsecuente en el programa. Por otra parte, si el bit “I” es uno lógico, cada vez que se termina la ejecución de una instrucción, el MCU no checa si hay un requerimiento de interrupción y procede a la ejecución de la instrucción siguiente presente en el programa.**

Si una determinada aplicación va usar interrupciones, al inicio del programa asociado con ésta, se debe hacer que el nivel del bit “I” sea cero, esto puede lograrse colocando, al final del bloque inicial del programa después del código que inicializa los recursos del MCU que se van a emplear, la instrucción CLI.

# INTERRUPCIONES EN EL MCU MC9S08SH32

## TESTIFICACIÓN DE LOS REQUERIMIENTOS DE INTERRUPCIÓN (explicación genérica)

Como se ha visto en láminas anteriores, cada requerimiento de interrupción (RI), tiene asociado un número, que aquí denotamos como “N”. El RI N es testificado con el que un bit asociado con éste, denominado aquí como, “**TESTIGO DEL RI N**”, sea uno lógico. Si es el caso, el MCU procederá a la ejecución de los pasos que lo llevarán a la ejecución de la rutina de servicio asociada con el RI cuyo número asociado es N. Dicho bit es la salida de una compuerta AND de dos entradas. Una entrada a la compuerta es un bit, denominado aquí genéricamente como “**BANDERA TESTIGO N**”, que se pone en uno si se da el evento de hardware asociado con el RI implicado. La otra entrada es un bit que aquí denominamos como “**HABILITADOR LOCAL N**”. Para cada instancia de interrupción, existe dentro del MCU una compuerta AND asociada con las diversas instancias de interrupción que soporta el MCU. Cabe señalar que al reset, todos los habilitadores locales de las diversas instancias de interrupción *amanecen* en cero lógico. **Nótese que si el habilitador local es cero lógico, aún cuando se haya dado el evento de hardware, el requerimiento de interrupción no se dará; por lo tanto, si una determinada aplicación va a usar interrupciones, los habilitadores locales de cada una de las instancias de interrupción implicadas, deberán inicializarse a uno lógico.**



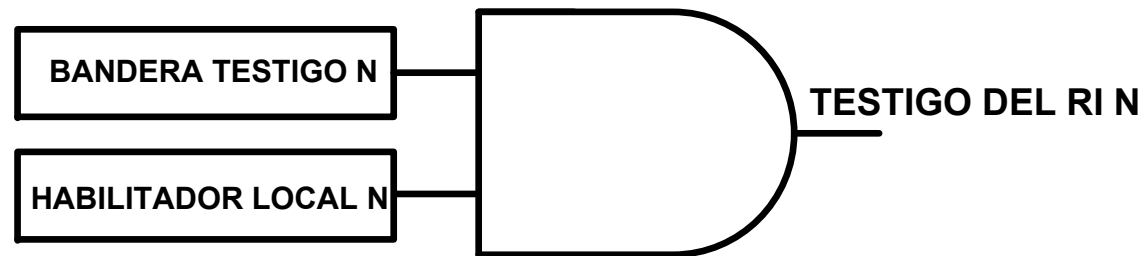
# INTERRUPCIONES EN EL MCU MC9S08SH32

## TESTIFICACIÓN DE LOS REQUERIMIENTOS DE INTERRUPCIÓN

(explicación genérica)

Es importante destacar que la bandera testigo está validada en cada caso por un LATCH, esto implica que una vez que se pone en uno, debido a la ocurrencia en cada caso del evento de hardware asociado, **deberá ser retornada a cero lógico; esto debe hacerse en la rutina de servicio asociada.** Si esto no se hace, al retornar de la interrupción la salida de la compuerta lógica que testifica el RI, seguirá en uno lógico, lo cual hará que el MCU entre a un ciclo infinito de entradas y salidas de la rutina de servicio, **sin atender absolutamente nada más.**

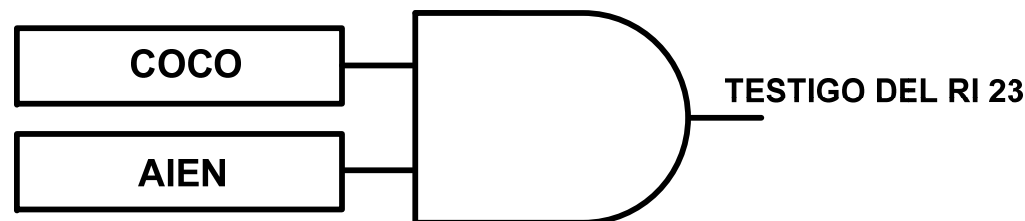
Los pasos a seguir para retornar a cero las banderas testigo, se detallan en las explicaciones propias de éstas, presentes en el manual del MCU.



# INTERRUPCIONES EN EL MCU MC9S08SH32

## TESTIFICACIÓN DE LOS REQUERIMIENTOS DE INTERRUPCIÓN (caso explícito)

La denotación de los bits explícitos para la bandera testigo y el habilitador local que corresponden a cada una de las instancias de interrupción, soportadas por el MCU MC9S08SH32, puede verse en las columnas “source” y “enable” de la tabla 5-2 de la página 64 del documento mc9s08sh.pdf. En cada caso, tales bits forman parte de registros asociados con el control y operación del periférico del MCU asociado con el evento de hardware que generaría el requerimiento de interrupción. Por ejemplo, a partir de la tabla puede verse que para la instancia 23 de interrupción, que genera un RI cuando el convertidor analógico digital ha completado una conversión y el dato binario asociado está listo para ser leído, la bandera testigo 23 se denomina “COCO” y el habilitador local 23 se denota como “AIEN”. El registro donde están estos dos bits se denomina ADCSC1 y la descripción de la funcionalidad de sus bits está en la página 128 del documento mc9s08sh.pdf.



# INTERRUPCIONES EN EL MCU MC9S08SH32

## VECTORES DE INTERRUPCIÓN

Al par de bytes que denotan la dirección de memoria donde inician cada una de las rutinas de servicio, asociadas con cada una de las instancias de interrupción, que use una determinada aplicación, se le denomina “**vector de interrupción**” propio de la instancia implicada.

Para cada una de las instancias de interrupción existe un par de direcciones de memoria donde se deben colocar este par de bytes, ya que cuando el MCU responde a un RI, en algún momento pasa a leer de esas localidades la dirección de memoria a donde debe saltar para iniciar la ejecución de la rutina de servicio asociada con el RI. Las direcciones de memoria donde el programador debe colocar cada uno de los vectores de interrupción propios de cada una de las instancias que use su aplicación, puede verse en la columna ADDRESS (High/Low) de la tabla 5-2 aquí multicitada.

Por ejemplo, para la instancia de interrupción 23 las direcciones de colocación del vector de interrupción son \$FFD0 y \$FFD1.

# INTERRUPCIONES EN EL MCU MC9S08SH32

## **DIRECCIONES DE COLOCACIÓN DE VECTORES DE INTERRUPCIÓN PARA UN DISPOSITIVO CHIPBAS8SH**

Las direcciones de colocación de los vectores de interrupción mostradas en la tabla 5-2 del manual, son validas siempre que el MCU no contenga firmware de base (FB). Si en el MCU existe firmware de base, como es el caso es el caso del dispositivo CHIPBAS8SH presente en la tarjeta FACIL\_08SH, las direcciones de colocación de los vectores de interrupción estarán fuera del intervalo de 10k que ocupa el FB (\$D800 a \$FFFF); de hecho, el FB configura el que las direcciones de colocación de los vectores estén 10k por debajo de las indicadas en la tabla 5-2. Por ejemplo, para un dispositivo CHIPBAS8SH, las direcciones de colocación del vector asociado con la instancia de interrupción 23 serán \$D7D0 y \$D7D1 y las de la instancia 11 serán \$D7E8 y \$D7E9.

**Se aprecia que para obtener la direcciones de colocación de los vectores de interrupción para un dispositivo CHIPBAS8SH, simplemente el byte alto del par de direcciones que se muestran en la tabla 5-2 debe ser cambiado de “”FF” a “D7”.**

# INTERRUPCIONES EN EL MCU MC9S08SH32

## **PASOS QUE SIGUE EL MCU AL RESPONDER A UN REQUERIMIENTO DE INTERRUPCIÓN**

Una vez que el MCU ha reconocido un requerimiento de interrupción, los pasos que sigue para proceder a ejecutar la rutina de servicio asociada, se describen a continuación

# INTERRUPCIONES EN EL MCU MC9S08SH32

## PASOS QUE SIGUE EL MCU AL RESPONDER A UN REQUERIMIENTO DE INTERRUPCIÓN

PASO 1: Guarda en el stack cinco bytes en el siguiente orden:

PCL (byte bajo del PC de retorno)

PCH (byte alto del PC de retorno)

Registro X

Registro A

Registro CCR

**Nótese que no se guarda en el stack el registro H, entonces si el código de la rutina de servicio modifica H, es responsabilidad del programador el poner código en la rutina de servicio, que al entrar lo guarde, y rescate antes de salir de ésta, empleando para esto las instrucciones PSHH y PULH.**



# INTERRUPCIONES EN EL MCU MC9S08SH32

## PASOS QUE SIGUE EL MCU AL RESPONDER A UN REQUERIMIENTO DE INTERRUPCIÓN

PASO 2: El bit “I” se pone en uno lógico

$$I \leftarrow 1$$

**Nótese que el efecto de esto es que una rutina de servicio de interrupción no podrá, a su vez, ser interrumpida.**

# INTERRUPCIONES EN EL MCU MC9S08SH32

## **PASOS QUE SIGUE EL MCU AL RESPONDER A UN REQUERIMIENTO DE INTERRUPCIÓN**

PASO 3: Lee el valor del vector de interrupción asociado, esto desde el par de direcciones, propias de la instancia de interrupción implicada.

# INTERRUPCIONES EN EL MCU MC9S08SH32

## **PASOS QUE SIGUE EL MCU AL RESPONDER A UN REQUERIMIENTO DE INTERRUPCIÓN**

PASO 4: Salta a la dirección denotada por el vector de interrupción capturado en el paso anterior, esto es, inicia la ejecución de la rutina de servicio asociada con la instancia de interrupción para la cual se está respondiendo.

# INTERRUPCIONES EN EL MCU MC9S08SH32

## TIEMPO DE LATENCIA DE INTERRUPCIÓN

El tiempo que transcurre entre el instante en que se da un requerimiento de interrupción, y el instante en que el MCU inicia la ejecución de la rutina de servicio asociada, se denomina TIEMPO DE LATENCIA DE INTERRUPCIÓN, que aquí se denota como “ $T_{lat}$ ”. Puede verse que para microcontroladores de la familia hcs08,  $T_{lat}$  estará en el intervalo definido por la siguiente desigualdad:

$$1.1T_b \leq T_{lat} \leq 2.2T_b$$

Donde  $T_b$  es el periodo del reloj de bus. Por ejemplo, si éste es de 20 MHz,  $T_{lat}$  podrá estar entre 0.55  $\mu$ s y 1.1  $\mu$ s

# INTERRUPCIONES EN EL MCU MC9S08SH32

## **FORMATO DE UN PROGRAMA QUE USE INTERRUPCIONES**

En general, un programa que maneje interrupciones, estará integrado por cuatro bloques funcionales, denominados aquí como BLOQUE1, BLOQUE2, BLOQUE3 y BLOQUE4.

### **ESTRUCTURA DE UN PROGRAMA QUE USE INTERRUPCIONES**

BLOQUE 1

BLOQUE 2

BLOQUE 3

BLOQUE 4

A continuación se menciona que deben contener cada uno de los bloques.

# INTERRUPCIONES EN EL MCU MC9S08SH32

## FORMATO DE UN PROGRAMA QUE USE INTERRUPCIONES

En general, el BLOQUE 1 contendrá:

- Código que inicialice los periféricos del MCU que se usen, para fines de la funcionalidad de éstos en la aplicación
- Código que inicialice en uno lógico a todos los habilitadores locales, propios de cada una de las instancias de interrupción, que se usen en la aplicación
- Al final de este bloque deberá estar la instrucción **CLI**, lo cual hará que el MCU pueda responder a los requerimientos de interrupción implicados en la aplicación

# INTERRUPCIONES EN EL MCU MC9S08SH32

## **FORMATO DE UN PROGRAMA QUE USE INTERRUPCIONES**

En general, el BLOQUE 2 contendrá:

- Código que por lo regular es un lazo donde el MCU efectúa acciones propias de la funcionalidad básica de la aplicación.

# INTERRUPCIONES EN EL MCU MC9S08SH32

## FORMATO DE UN PROGRAMA QUE USE INTERRUPCIONES

En general, el BLOQUE 3 contendrá:

- Código de cada una de las rutinas de servicio asociadas con cada una de las instancias de interrupción implicadas en la aplicación.

**Es importante destacar que las rutinas de servicio de interrupción, deben concluirse con la instrucción RTI, y no con la instrucción RTS.**

- Código de cada una de las rutinas llamables con la instrucción JSR que se usen en la aplicación, éstas desde luego que se concluyen con la instrucción RTS.



# INTERRUPCIONES EN EL MCU MC9S08SH32

## FORMATO DE UN PROGRAMA QUE USE INTERRUPCIONES

En general, el BLOQUE 4 contendrá en el orden indicado:

1. Si es necesario, código que declare tablas de datos que pudieran requerirse en la aplicación.
2. Código que declare la colocación de cada uno de los vectores de interrupción, asociados con cada una de las instancias de interrupción implicadas en la aplicación.

# INTERRUPCIONES EN EL MCU MC9S08SH32

## FORMATO DE LAS DECLARACIONES PARA COLOCAR VECTORES DE INTERRUPCIÓN

Esto se ilustra aquí con un ejemplo explícito.

Supóngase que una aplicación usa las instancias de interrupción 11 y 20, y que el programador denota el inicio de las dos rutinas de servicio con las etiquetas “servint11:” y “servint20:”.

De acuerdo con la tabla 5-2 de la página 64 del documento MC9S08SH32.PDF, las direcciones de colocación de los vectores asociados son: \$ffe8:\$ffe9 para la instancia 11, y \$ffd6:\$ffd7 para la instancia 20.

Dado que para un dispositivo CHIPBAS8SH, se sabe que las direcciones de colocación de los vectores se obtienen cambiando en cada caso el byte alto FF por el byte D7, en las direcciones obtenidas directamente de la tabla 5-2; las direcciones de colocación de los vectores de las instancias 11 y 20 respectivamente serían \$d7e8:\$d7e9 y \$d7d6:\$d7d7.

# INTERRUPCIONES EN EL MCU MC9S08SH32

## FORMATO DE LAS DECLARACIONES PARA COLOCAR VECTORES DE INTERRUPCIÓN

Considerando las direcciones recabadas, la declaración de la colocación del vector asociado con la instancia 11 sería:

```
org $d7e8  
dw servint11
```

Para la instancia 20 la declaración de colocación del vector sería:

```
org $d7d6  
dw servint20
```

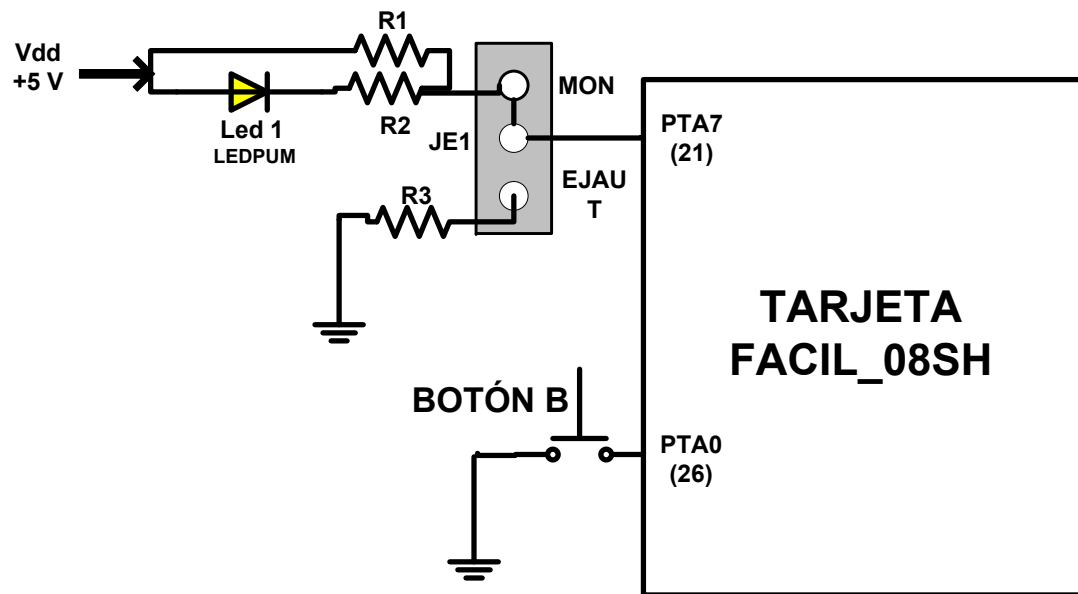
# INTERRUPCIONES EN EL MCU MC9S08SH32

## EJEMPLO ILUSTRATIVO

Para aclarar ideas se plantea aquí un programa ilustrativo que usa la instancia 20 de interrupción. El evento de hardware asociado es la detección de un flanco de bajada en el bit pta0. El programa está contenido en el archivo “**ejemplo1\_intflpta0.asm**”.

Al ejecutarse éste, deberá observarse un parpadeo del ledpum, presente en la tarjeta FACIL\_08SH con una cadencia (apagado un segundo)/(encendido 25 ms), el código asociado con esta acción estaría en el bloque 2.

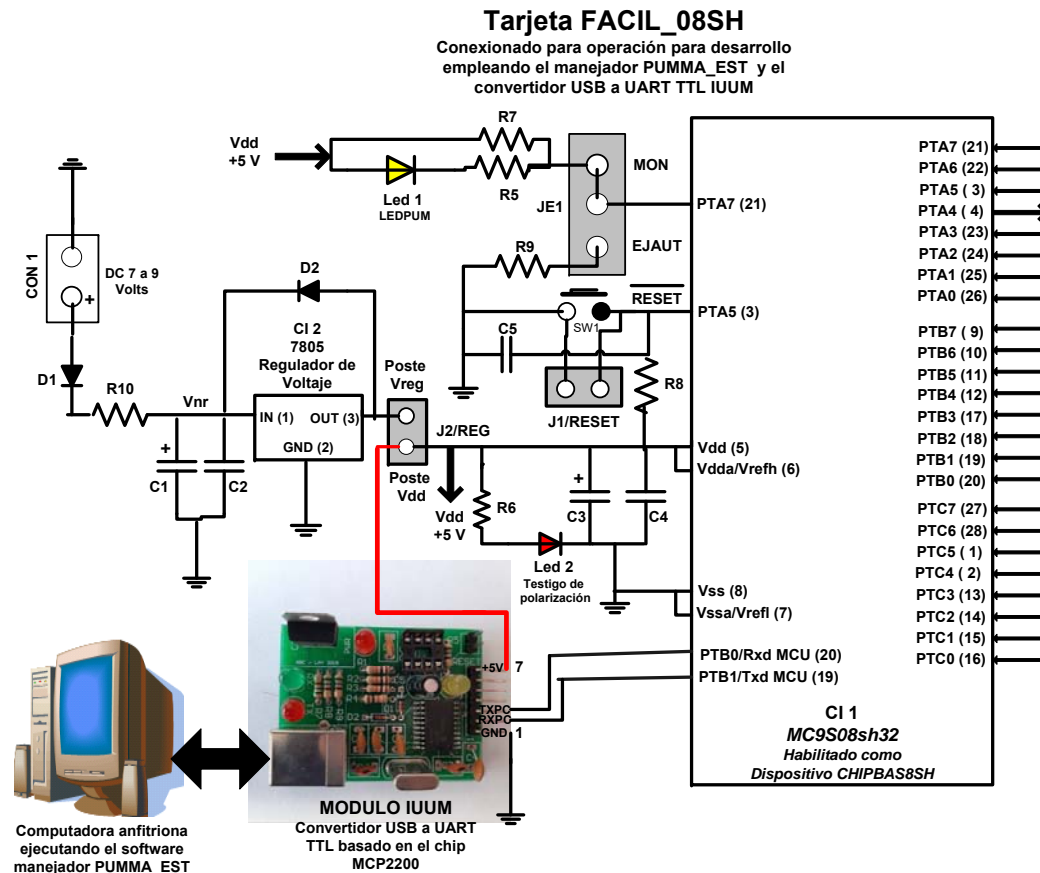
Por otra parte, al oprimirse y soltarse el botón B, el MCU pasaría a ejecutar la rutina de servicio, la cual haría que el ledpum permanezca encendido durante un segundo, para después retornar a la cadencia original. **Es importante que se lean los comentarios explicativos presentes en el programa.**



## PASOS A SEGUIR PARA EJECUTAR EN LA TARJETA FACIL\_08SH EL PROGRAMA ILUSTRATIVO

CONTENIDO EN EL ARCHIVO ejemplo1\_intflpta0.asm

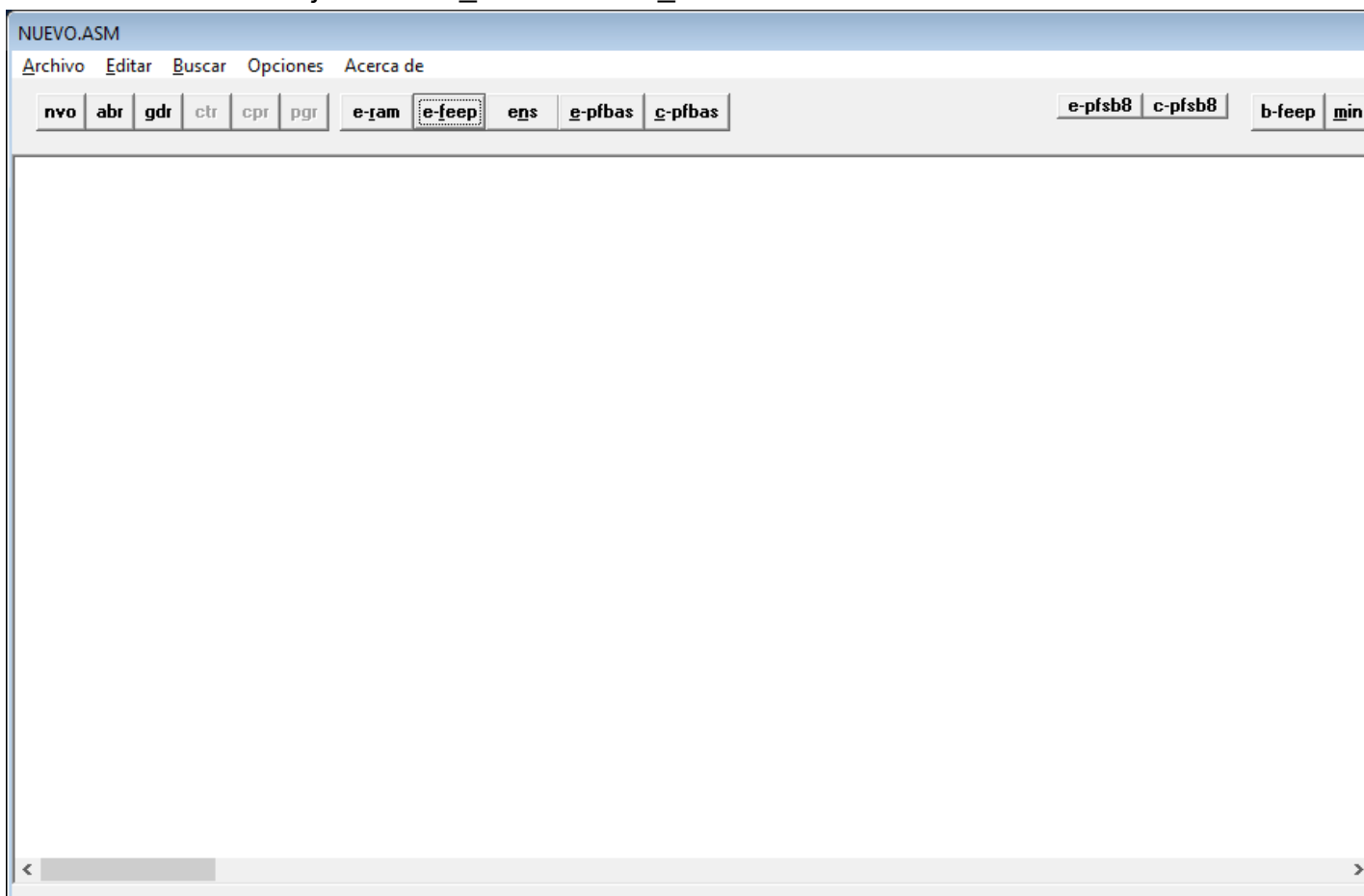
**Paso 1:** Conectar a la PC, empleando el modulo IUUM como se muestra en la figura, la tarjeta FACIL\_08SH. El ledpum en las tarjeta deberá parpadear, indicando que ésta está en posibilidad de recibir comandos desde el software manejador PUMMA\_EST



**PASOS A SEGUIR PARA EJECUTAR EN LA TARJETA FACIL\_08SH EL  
PROGRAMA ILUSTRATIVO  
CONTENIDO EN EL ARCHIVO ejemplo1\_intflpta0.asm**

**Paso 2:** Ejecutar en la PC el software PUMMA\_EST. Una vez que se haya inicializado PUMMA\_EST, y el usuario haya verificado la comunicación leyendo una página de memoria, deberá aparecer la ventana del editor de PUMMA\_EST como se muestra en la figura.

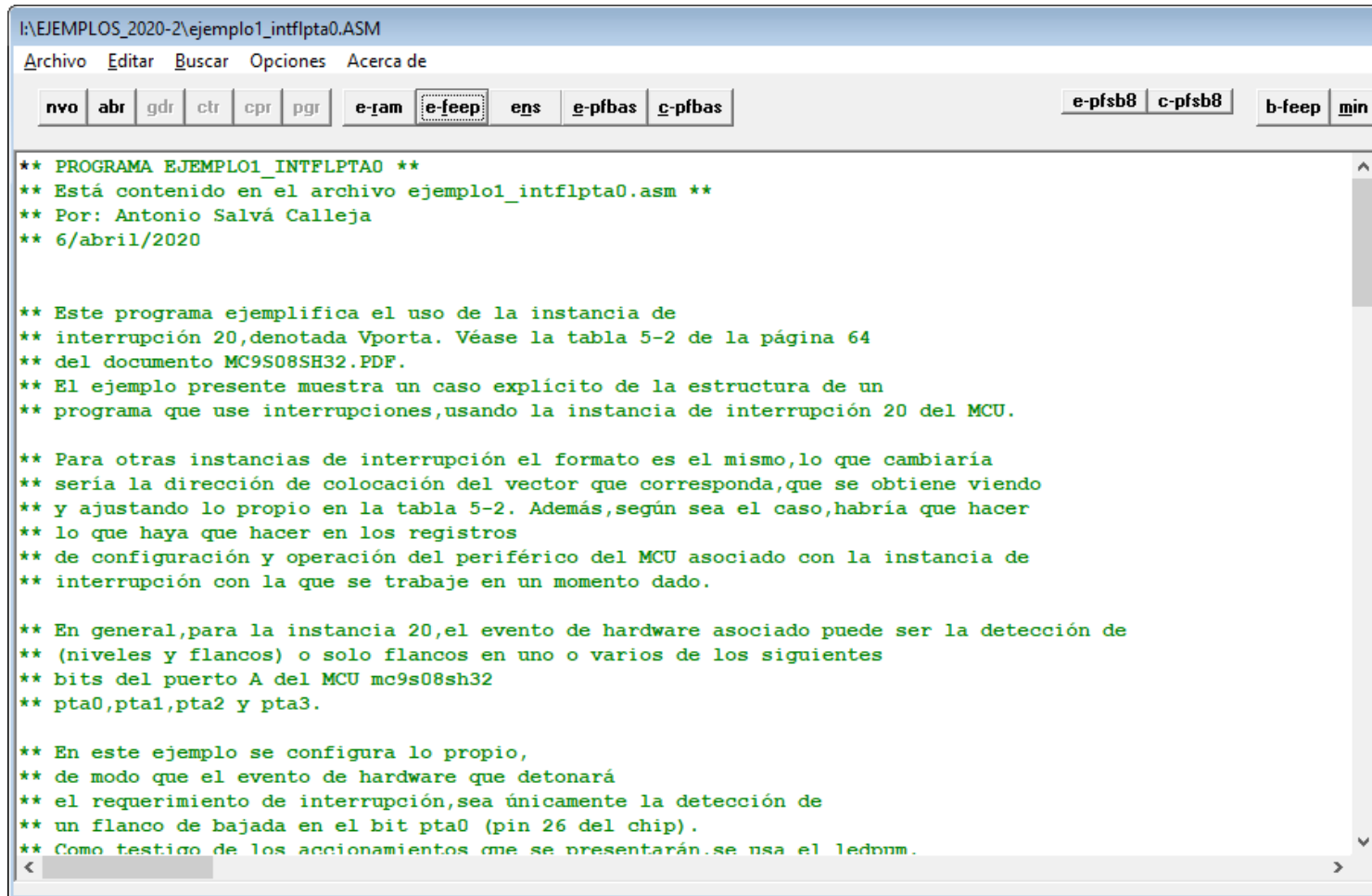
De ser necesario, pueden verse en la presentación Inisespum-facil\_08sh.pdf, los pasos a seguir para iniciar una sesión de trabajo PUMMA\_EST – FACIL\_08SH.



## PASOS A SEGUIR PARA EJECUTAR EN LA TARJETA FACIL\_08SH EL PROGRAMA ILUSTRATIVO

### CONTENIDO EN EL ARCHIVO ejemplo1\_intflpta0.asm

**Paso 3:** Abrir el archivo **ejemplo1\_intflpta0.asm**. Después de esto, oprimir el botón <b-feep>, lo cual borrará la memoria FLASH de usuario. El ledpum se apagará unos segundos, para regresar al parpadeo una vez que se haya concluido con el borrado de la memoria no volátil disponible para el usuario.



```
I:\EJEMPLOS_2020-2\ejemplo1_intflpta0.ASM
Archivo  Editar  Buscar  Opciones  Acerca de

nvo  abr  gdr  ctr  cpr  pgr  e-ram  e-feep  ens  e-pfbas  c-pfbas  e-pfsb8  c-pfsb8  b-feep  min

** PROGRAMA EJEMPLO1_INTFLPTA0 **
** Está contenido en el archivo ejemplo1_intflpta0.asm **
** Por: Antonio Salvá Calleja
** 6/abril/2020

** Este programa ejemplifica el uso de la instancia de
** interrupción 20,denotada Vporta. Véase la tabla 5-2 de la página 64
** del documento MC9S08SH32.PDF.
** El ejemplo presente muestra un caso explícito de la estructura de un
** programa que use interrupciones,usando la instancia de interrupción 20 del MCU.

** Para otras instancias de interrupción el formato es el mismo,lo que cambiaría
** sería la dirección de colocación del vector que corresponda,que se obtiene viendo
** y ajustando lo propio en la tabla 5-2. Además,según sea el caso,habría que hacer
** lo que haya que hacer en los registros
** de configuración y operación del periférico del MCU asociado con la instancia de
** interrupción con la que se trabaje en un momento dado.

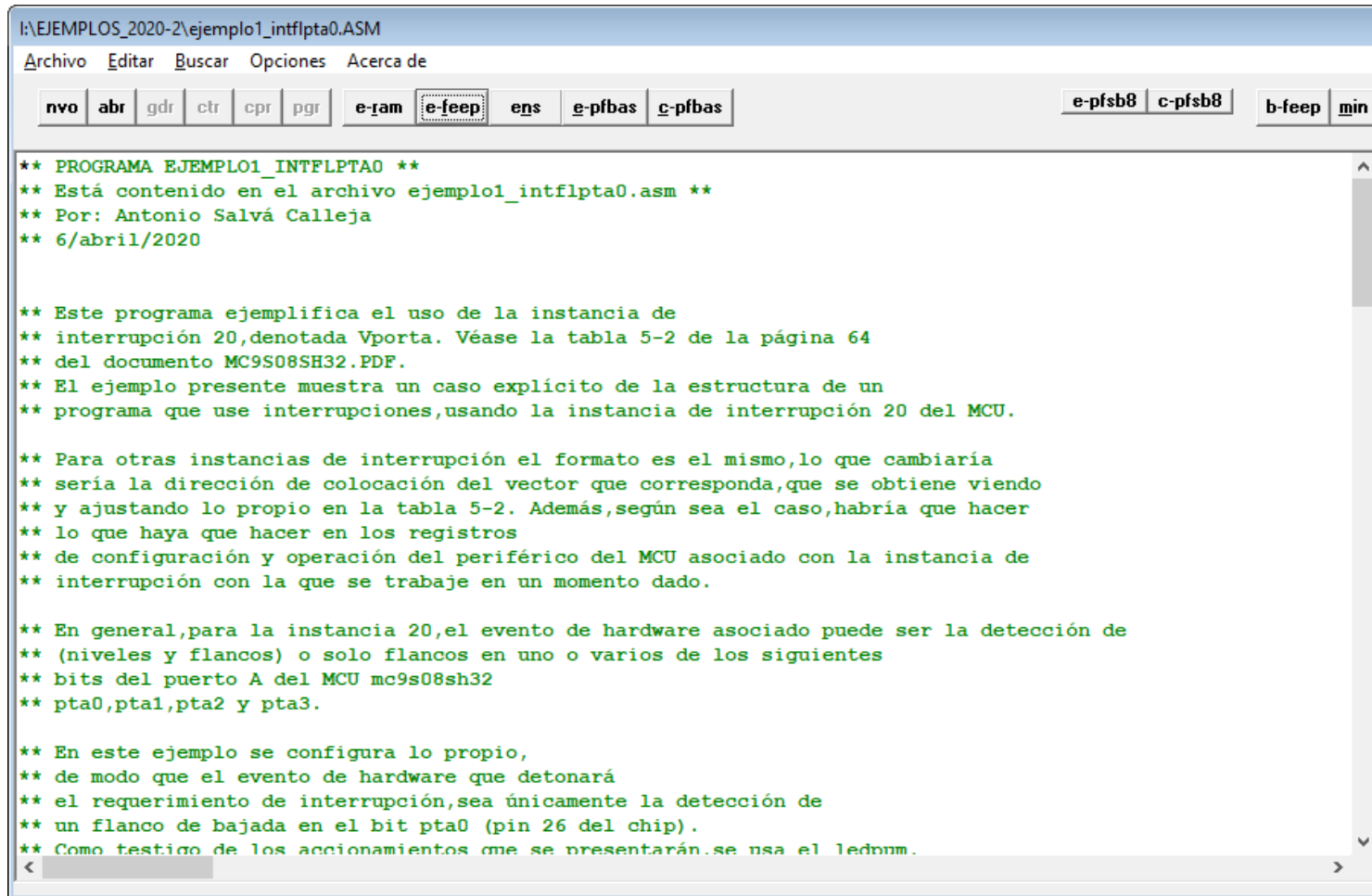
** En general,para la instancia 20,el evento de hardware asociado puede ser la detección de
** (niveles y flancos) o solo flancos en uno o varios de los siguientes
** bits del puerto A del MCU mc9s08sh32
** pta0,pta1,pta2 y pta3.

** En este ejemplo se configura lo propio,
** de modo que el evento de hardware que detonará
** el requerimiento de interrupción,sea únicamente la detección de
** un flanco de bajada en el bit pta0 (pin 26 del chip).
** Como testigo de los accionamientos que se presentarán,se usa el ledpum.
```

## PASOS A SEGUIR PARA EJECUTAR EN LA TARJETA FACIL\_08SH EL PROGRAMA ILUSTRATIVO

### CONTENIDO EN EL ARCHIVO ejemplo1\_intflpta0.asm

**Paso 4:** Oprimir el botón <e-feep>, lo cual hará que el programa contenido en la ventana del editor se cargue y ejecute en la memoria FLASH del usuario, a partir de la dirección \$8000.



```
I:\EJEMPLOS_2020-2\ejemplo1_intflpta0.ASM
Archivo  Editar  Buscar  Opciones  Acerca de
nvo  abr  gdr  ctr  cpr  pgr  e-ram  e-feep  ens  e-pfbas  c-pfbas  e-pfsb8  c-pfsb8  b-feep  min

** PROGRAMA EJEMPLO1_INTFLPTA0 **
** Está contenido en el archivo ejemplo1_intflpta0.asm **
** Por: Antonio Salvá Calleja
** 6/abril/2020

** Este programa ejemplifica el uso de la instancia de
** interrupción 20,denotada Vporta. Véase la tabla 5-2 de la página 64
** del documento MC9S08SH32.PDF.
** El ejemplo presente muestra un caso explícito de la estructura de un
** programa que use interrupciones,usando la instancia de interrupción 20 del MCU.

** Para otras instancias de interrupción el formato es el mismo,lo que cambiaría
** sería la dirección de colocación del vector que corresponda,que se obtiene viendo
** y ajustando lo propio en la tabla 5-2. Además,según sea el caso,habría que hacer
** lo que haya que hacer en los registros
** de configuración y operación del periférico del MCU asociado con la instancia de
** interrupción con la que se trabaje en un momento dado.

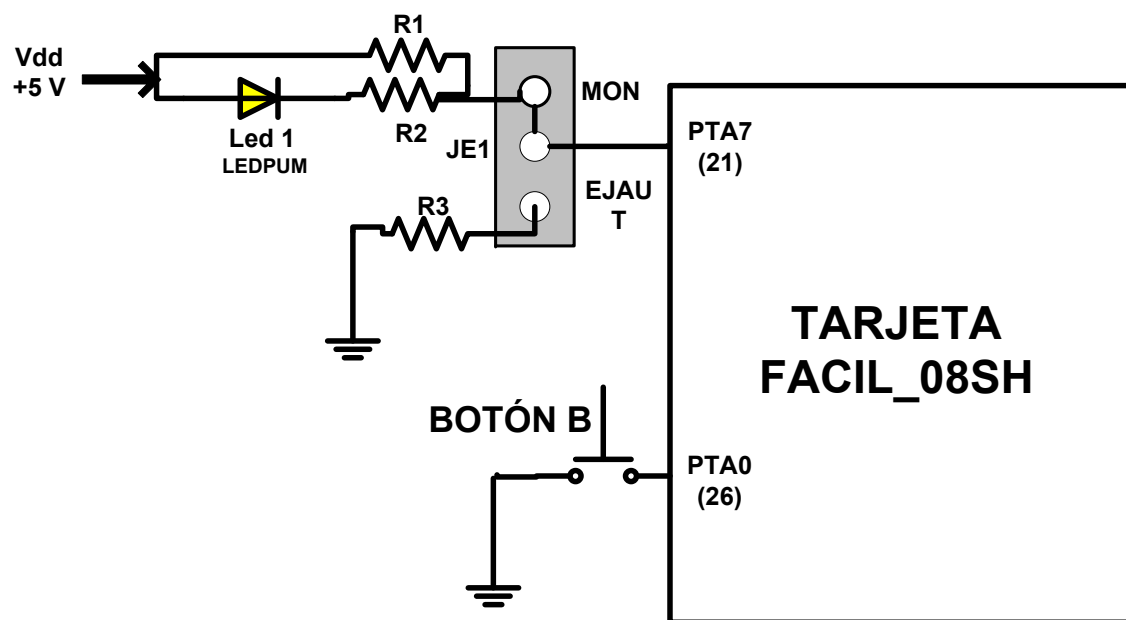
** En general,para la instancia 20,el evento de hardware asociado puede ser la detección de
** (niveles y flancos) o solo flancos en uno o varios de los siguientes
** bits del puerto A del MCU mc9s08sh32
** pta0,pta1,pta2 y pta3.

** En este ejemplo se configura lo propio,
** de modo que el evento de hardware que detonará
** el requerimiento de interrupción,sea únicamente la detección de
** un flanco de bajada en el bit pta0 (pin 26 del chip).
** Como testigo de los accionamientos que se presentarán,se usa el lednum.
```



**PASOS A SEGUIR PARA EJECUTAR EN LA TARJETA FACIL\_08SH EL  
PROGRAMA ILUSTRATIVO  
CONTENIDO EN EL ARCHIVO ejemplo1\_intflpta0.asm**

**Paso 5:** Una vez que el programa se esté ejecutando, se deberá observar un parpadeo del ledpum con la cadencia (apagado un segundo)/(encendido 25 ms).



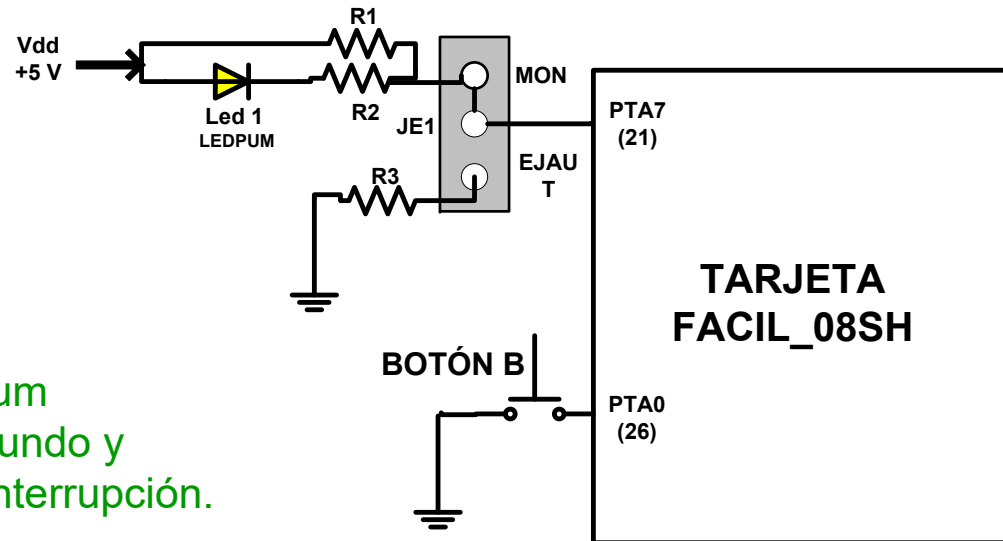
## PASOS A SEGUIR PARA EJECUTAR EN LA TARJETA FACIL\_08SH EL PROGRAMA ILUSTRATIVO

CONTENIDO EN EL ARCHIVO ejemplo1\_intflpta0.asm

**Paso 6:** Oprimir y soltar el botón B, esto hará que se presente un flanco de bajada en el bit pta0, lo cual hará que el MCU pase a ejecutar la rutina de servicio, la cual hará que el ledpum permanezca prendido durante un segundo, después de esto se retorna de la interrupción, regresando el ledpum a la cadencia (apagado un segundo)/(encendido 25 ms). Como referencia didáctica, se muestra aquí la rutina de servicio, cuyo inicio fue denotado por el programador con la etiqueta “servfl:”.

```
servfl:    lda ptasc
           ora #$04
           sta ptasc ; ptaif ← 0

           bclr 7,ptad ;enciende ledpum
           bsr ret1seg ;espera un segundo y
           rti         ;retorna de la interrupción.
```



## PASOS A SEGUIR PARA EJECUTAR EN LA TARJETA FACIL\_08SH EL PROGRAMA ILUSTRATIVO

### CONTENIDO EN EL ARCHIVO ejemplo1\_intflpta0.asm

**Paso 7:** Oprimir y soltar el botón de RESET de la tarjeta FACIL\_08SH, esto hará que ésta regrese a la ejecución del receptor de comandos que se envíen desde PUMMA\_EST, testificándose esto con una cadencia (encendido 30ms)/(apagado 30 ms) en el ledpum. Aquí se podría modificar algo del programa presente en el editor y reejecutarlo oprimiendo el botón <e-feep> para apreciar los cambios efectuados; o bien, abrir otro programa y ejecutarlo siguiendo los pasos 4 a 7 aquí descritos. Considerando que si el programa a ejecutarse ha de cargarse en RAM el botón a oprimir en el paso 4 deberá ser <e-ram>, por otra parte, si el programa ha de ejecutarse en FLASH, el botón a oprimirse en el paso 4 deberá ser <e-feep>; como fue el caso del ejemplo ilustrativo de interrupciones por flanco de bajada recién aquí descrito.

