FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Departamento de Electrónica

66.09 Laboratorio de Microcomputadoras

Proyecto

Controlador de ensayos de *Spark Plasma*

Profesor:	Ing Guillermo Campiglio
Cuatrimestre / Año	1.er cuatrimestre 2008
Turno de clases prácticas:	Miércoles
Jefe de Trabajos Prácticos:	
Docente Guía:	

Alumno			Seguimiento del proyecto								
Apellido	Nombre	Padrón									
Albani	Francisco	84891									
Arbelo Arrocha	Daniel	85583									
Figueroa	Gonzalo	84255									

Obsevaciones: _____

Fecha de aprobación				Firma del J.T.P.

Coloquio	
Nota Final	
Firma Profesor	

Índice

1. Introducción

En el **Laboratorio de Sólidos Amorfos** de la **Facultad de Ingeniería de la Universidad de Buenos Aires**, se producen materiales magnéticos mediante técnicas de solidificación rápida como *melt-spinning* o atomización centrífuga. Los materiales obtenidos por estas técnicas tienen forma de cintas o polvos, necesitando ser posteriormente consolidados en piezas de mayor tamaño.

La consolidación de las piezas masivas en su forma final requiere técnicas de compactación y sinterizado. Dado que los materiales deben ser micro y nano-estructurados, con el objeto de potenciar sus propiedades magnéticas en forma acorde a su aplicación, es altamente deseable tener un control granular y preciso del proceso de consolidación.

En el caso de la compactación de ferromagnetos nanocompuestos de NdFeB[1], que son empleados como imanes permanentes, se requiere que dicha técnica de consolidación no provoque un crecimiento exagerado del tamaño de grano (que en casos típicos ronda las decenas de nanómetros), porque esto redunda en una degradación de sus propiedades magnéticas.

Una técnica acorde con este último objetivo es la compactación con descarga eléctrica. Un esquema del mismo se puede observar en la figura 1. Una densidad de corriente eléctrica muy alta (típicamente de 50 kA/cm^2) provoca la soldadura superficial de las partículas. El efecto responsable de esta soldadura es el calentamiento Joule local, el cual actúa en el breve lapso de tiempo que dura la descarga. Sin embargo, en el interior de las partículas la temperatura no alcanza los valores correspondientes como para promover la difusión atómica que provoca el crecimiento de granos, o si lo hace, su cinética de calentamiento es tan rápida que no permite el crecimiento excesivo de los mismos.

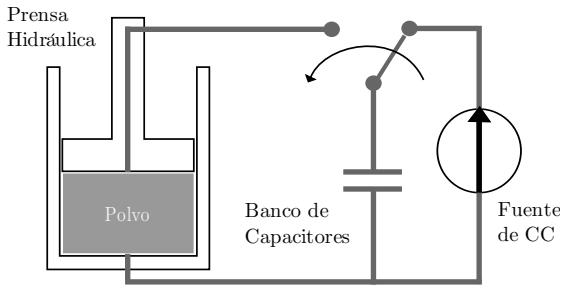


Figura 1: Esquema de un sistema de compactación con descarga eléctrica.

[1] NdFeB: Neodimio Hierro Boro.

2. Descripción del dispositivo

El dispositivo desarrollado tiene como objetivo asistir a los investigadores del **Laboratorio de Sólidos Amorfos** en la ejecución de los ensayos de fundición de materiales magnéticos mediante compactación y descarga eléctrica.

Para alcanzarlo, debe controlar y monitorear los parámetros que definen a un ensayo, y ofrecer al invesigador una interfaz para programar una serie de ellos. Los parámetros son:

- Presión mecánica sobre el material.
- Tiempo de reposo de la muestra bajo presión.
- Tensión de carga del banco de capacitores que provoca la descarga.
- Ventana de tiempo de la descarga sobre la muestra.

La interacción con el usuario se da por medio de una terminal constituida por un teclado y un *display* gráfico a través de los cuales se puede ver el estado de la ejecución de ensayos y configurar sus parámetros.

Por otro lado, una interfaz eléctrica constituida por sensores y *relays* activados digitalmente provee control directo sobre el sistema, reflejando los comandos del operario y adquiriendo los datos que son presentados a éste.

El dispositivo almacena todos los ensayos en una memoria ROM para permitirle al investigador rehacerlos individualmente.

A continuación se presenta un diagrama de bloques que describe superficialmente las relaciones entre los distintos módulos:

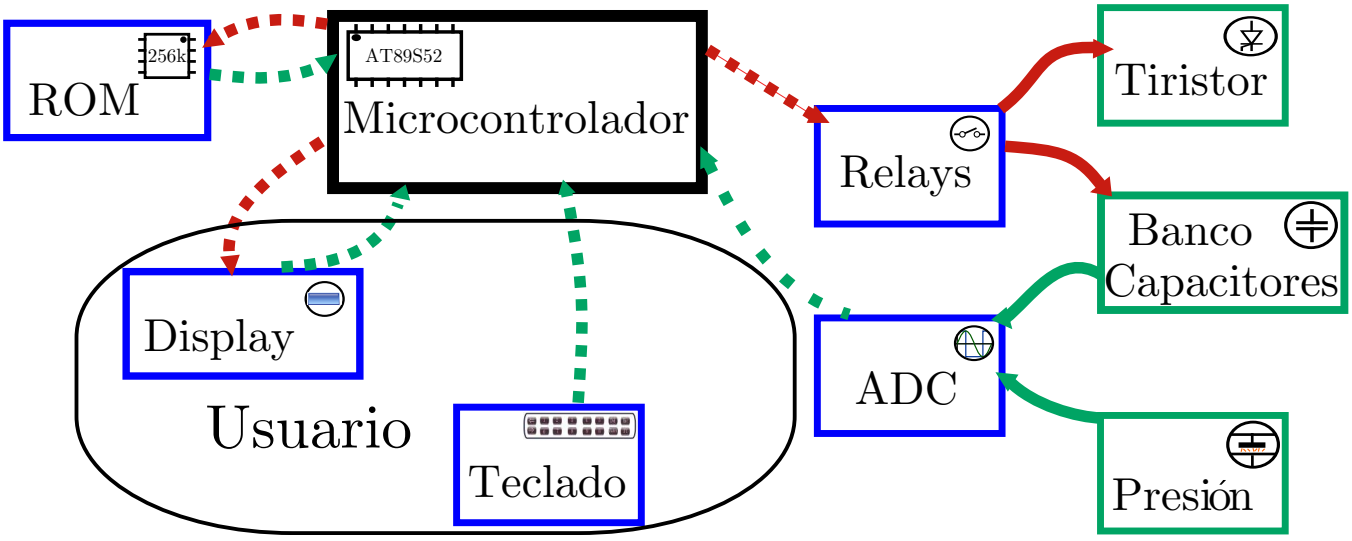


Figura 2: Diagrama en bloques.

3. Hardware

En esta sección se describe como se resolvió técnicamente cada uno de los módulos nombrados anteriormente, terminando con un listado de costos.

3.1. Microcontrolador

Adquisidor de todas las entradas y salidas de datos, manejo de interfases y comunicación. Se implementó con un microcontrolador *AT89S52* de la familia *MCS-51* de Intel.

Algunas de sus características son las siguientes:

- 8 *kbytes* de memoria Flash.
- 256 *bytes* de memoria RAM.
- 4 puertos de 8 líneas cada uno.
- 3 timers de 16-*bits*.

Los motivos que llevaron a elegir este dispositivo, fueron principalmente su simpleza, su fuerte orientación a aplicaciones de control, y su capacidad para almacenar código. Eso permitió almacenar a los 95 caracteres con representación gráfica del Standard ASCII de 7 bits completo en memoria de código.

3.2. Display gráfico

El *display* es el principal dispositivo en lo que a interacción con el usuario respecta. Se utilizó el modelo *WG12864A-PMI-V* del fabricante *Winstar*. Posee un superficie de 128×64 pixeles dividida en dos sectores de 64x64 pixeles, cada uno controlado por un *chip KS0108*.

La memoria RAM del *chip KS0108* almacena los píxeles que son mostrados en el *display* y posibilita direccionar 64 columnas × 64 filas. Las filas están agrupadas en 8 páginas de 8 filas cada una. Estas últimas no son más que "tiras" de *bytes* que son leídos o escritos a través de los terminales DB0-DB7 del *display*.

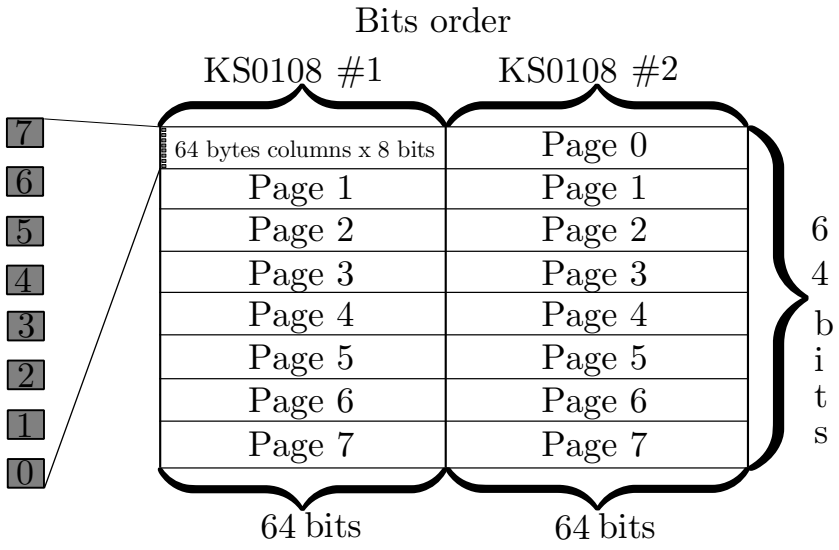


Figura 3: Relación entre la memoria del controlador y su ubicación en la pantalla.

Para mostrar datos en pantalla se deben seguir, a grandes rasgos, los siguientes pasos: seleccionar el sector a escribir, especificar la página, especificar la columna de inicio dentro de la página y, finalmente, escribir los datos. Es importante destacar que el puntero interno de columnas se incrementa cada vez que se realiza una escritura, permitiendo el volcado secuencial de *bytes* sobre una página.

Dentro de la biblioteca `display.asm`, junto a las rutinas necesarias para inicialización y comunicación con el *display*, se incluyeron también otras de más alto nivel. A continuación se puede ver una descripción de cada una de ellas:

- `DISPLAY_PUT_BYTE:` muestra un *byte* (8 píxeles verticales) en una posición específica del *display*.
- `DISPLAY_PUT_CHAR:` muestra los 5 *bytes* que componen a un caracter a partir de ru representación almacenada en memoria de código.
- `DISPLAY_PUT_ASCII:` muestra un caracter a partir de su código ASCII.
- `DISPLAY_PUT_STRING_STATIC:` escribe una cadena ASCII ubicada en memoria de código.
- `DISPLAY_PUT_STRING_DYNAMIC:` escribe una cadena ASCII ubicada en memoria de datos.
- `DISPLAY_PUT_SCREEN_FROM_CODE:` vuelca los 1024 *bytes* que componen una pantalla entera desde memoria de código.

3.3. Teclado

Se diseñó un teclado matricial de 4 filas por 4 columnas, y para su implementación se utilizó un *buffer* *74HC244N* de 8 entradas y 8 salidas como interfaz al microcontrolador, conectado de la siguiente manera:

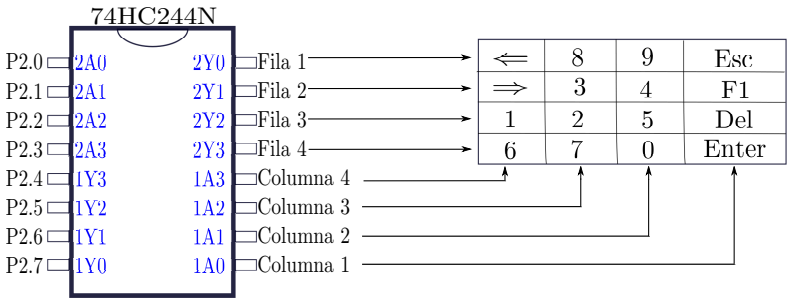


Figura 4: Conexionado del teclado.

Se pensó de tal forma que la presión de una tecla implicara un nivel lógico bajo tanto en su fila como en su columna. De esta manera, el procedimiento de detección de una tecla puede describirse así:

- Enviar un cero lógico **sólo** a la fila de la tecla deseada (alguno de los terminales 2An del *buffer*).
- Si dicha tecla está presionada, el valor lógico se propagará en poco tiempo a uno de los terminales 1An del *buffer*, determinado por la columna a la que la tecla pertenece.
- Esperar un tiempo prudente para que el buffer actualice los valores 1Yn.
- Se toma una lectura de los 4 terminales 1Yn y se chequea si la columna de la tecla deseada tiene el valor lógico cero.

Para simplificar el procedimiento anterior, se escribió una rutina, llamada `KEYPAD_SCAN_CORE:`, que cuando es llamada, prueba en sucesión a las 16 combinaciones válidas. Para simplificar aún más esta tarea, se definió una tabla de *bytes* donde cada uno representa a la máscara correspondiente a cada tecla. En los 4 *bits* menos significativos, se almacena una de las 4 "peticiones" posibles: 1110 para la fila 1, 1101 para la fila 2, 1011 para la fila 3 y 0111 para la fila 4; en los cuatro más significativos, se almacenan cada una de las "respuestas" válidas: 1110 para la columna 4, 1101 para la columna 3, 1011 para la columna 2 y 0111 para la columna 1. De esta manera, se puede usar el mismo *byte* para el momento del envío de la máscara de filas y para el momento de comparación de la máscara de columnas.

El orden de las máscaras en la tabla se hizo coincidir con el orden de la codificación de las teclas, para reducir a una simple iteración de variable la parte del algoritmo que informa cual de las teclas se detectó presionada.

Finalmente, se encapsuló a la rutina anteriormente descripta, dentro de otra preparada para ser utilizada fuera de la biblioteca, llamada `KEYPAD_SCAN:`, que se encarga de realizar tareas de *debouncing* y atrapar la ejecución hasta que la tecla detectada haya sido soltada, para evitar múltiples disparos.

Para complementar, se incluyeron en la biblioteca `keypad.asm` rutinas de más alto nivel para realizar acciones como atrapar la ejecución a la espera de un tecla cualquiera o específica e iterar bucles hasta detectar alguna presionada.

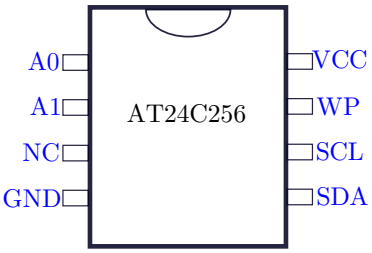
A continuación se presenta la disposición física de las teclas que componen el periférico:

Figura 5: Disposición física del teclado.

3.4. Memoria ROM externa

Otro de los periféricos que conforman el dispositivo, es una memoria ROM serial. Se trata del modelo *AT24C256* de *ATMEL* y se utilizó para almacenar los ensayos que hiciera el usuario y las pantallas gráficas, meramente estéticas.

Este integrado provee 32,768 *bytes* de memoria E^{2}PROM y posibilita la conexión en cascada de tres integrados más para una eventual ampliación de memoria.



Pin	Function
A0 - A1	Address Inputs
SDA	Serial Data
SCL :	Serial Clock Input
WP	Write Protect
NC	No Connect
GND	Ground

3.4.1. Operaciones generales

- ***Clock* y envio de datos:** El pin SDA debe cambiar sólo cuando el pin SCL se encuentra bajo. Ésto indica el envio de datos. El cambio de SDA mientras SCL está en alto se interpreta como *bits* de *start-stop*.
- **Condición de *start*:** Con el SCL en alto y realizando una transición de alto a bajo en el SDA se estableca la condición de *start*.
- **Condición de *stop*:** Con el SCL en alto y realizando una transición de bajo a alto en el SDA se estableca la condición de *stop*. Una vez efectuada la memoria se pone en modo *standby*.
- ***Acknowledge*:** La memoria envia un cero durante el noveno ciclo de *clock* para corroborar que recibió en forma correcta cada uno de los *bytes*.

3.4.2. *Device Address*

Para determinar la función que realiza la memoria, ya sea escritura o lectura, se utiliza un *byte* seguido de la condición de *start*. A1 y A0 son dos *bits* que forman parte de dicho *byte* y que se utilizan para selecionar una de las cuatro memorias que pudieran estar conectadas. Cabe destacar que el *byte* en cuestión es establecido por el fabricante y se denomina *device address*.

MSB

1	0	1	0	0	*A1*	A_0	R/W

LSB

El bit menos significativo determina la operación; si está en alto se realizará una lectura y si está en bajo se realizará una escritura.

3.4.3. Operaciones de escritura

Para realizar una escritura en la ROM se requieren, además de la *device address*, 2 *bytes* que determinan la posición en memoria donde se escribirán los datos. Una vez establecida la secuencia de direcciones se envian los datos, que finaliza con la condición de *stop*.

El modelo *AT24Cxx* permite el envío de hasta 64 *bytes* (una página) de datos antes de la condición de *stop*. Una vez enviada la dirección en memoria, la E^{2}PROM incrementa el *byte* bajo a medida que recibe los *bytes* de datos. Sin embargo, cuando se envian más de 64 *bytes* los datos comienzan a pisarse; ésto se denomina "*roll over*".

3.4.4. Operaciones de lectura

Como se dijo anteriormente la diferencia sustancial entre escribir la ROM y leerla radica en el bit menos significativo de la *device address*. Sin embargo es posible implementar, tres tipos de operaciones para la lectura:

- ***Current address*:** En este caso el contador interno de la ROM mantiene la última dirección de memoria incrementada en uno. Dicha dirección se mantiene siempre y cuando el chip esté alimentado.
- ***Random*:** Se envian a la ROM la *device address* y una dirección de memoria cualquiera simulando una escritura. Luego el microcontrolador debe generar otra condición de *start*. Y, a continuación inicia la lectura del *byte* random.
- ***Sequential*:** Es posible iniciar una lectura secuencial comenzando por cualquiera de las dos operaciones antes mencionadas. Siempre y cuando la E^2PROM reciba un *acknowledge* seguirá incrementando la posición en memoria permitiendo una lectura secuencial.

3.5. Conversor A/D

El conversor analógico-digital es el encargado de servir los datos sensados al microcontrolador, para que éste actúe en consecuencia. Se utilizó para procesar los datos provenientes del transmisor de presión y para controlar la tensión de carga del banco de capacitores.

La función básica del conversor es tomar una señal entre 0 V y 5 V y traducirla a un *byte* comprendido entre 0 y 255 respectivamente.

El modelo que se utilizó fue *ADC0834*, del fabricante *National Semiconductor*. Éste consta de 4 canales que pueden operar de dos maneras distintas:

- ***Single-Ended*:** compara la señal del canal con la referencia común.
- ***Differential*:** compara las señales de dos canales y obtiene la diferencia.

La comunicación entre el microcontrolador y el conversor se establece en forma serial y se utiliza un *byte* especifico, dado por el fabricante para determinar los canales a utilizar y el modo de operación. Para este proyecto se utilizaron dos canales en modo *Single-Ended*. El canal cero se encarga de controlar la carga de los capacitores y tiene asignado el siguiente *byte* para ser iterpretado por el conversor: $CH0 = 10000000b$. El sensor de presión se procesa a través del canal uno y su *byte* de reconocimiento es: $CH1 = 10100000b$.

3.6. Sensor de presión

En su diseño original el dispositivo debía ser capaz de operar en forma completamente desatendida, lo que implicaba la operación automatizada de la prensa por parte del microcontrolador.

Debido a limitaciones presupuestarias ese objetivo debió ser abandonado. Como solución de compromiso se optó por el sensado pasivo de la presión ejercida sobre el material, reportada por pantalla junto con la presión objetivo de la experiencia, para orientar al operario de la prensa manual.

Para poder medir la presión ejercida sobre el material se instaló en esta un sensor de presión modelo *ADZ-SML-20.0* de *Nagano* cuya especificación técnica indica que es capaz de sensar de 0 a 4000 bar con un error de ±0,1 bar, traduciendo lo sensado a una tensión entre 0 V y 10 V.

Adicionalmente se adaptó, mediante un divisor resistivo (Figura 3.6), la tensión de salida del transductor al rango (0 V − 5 V). Una vez adaptada a este rango, la tensión es interpretada por el conversor A/D conectado al microcontrolador, permitiendole a este monitorear el proceso de compactación.

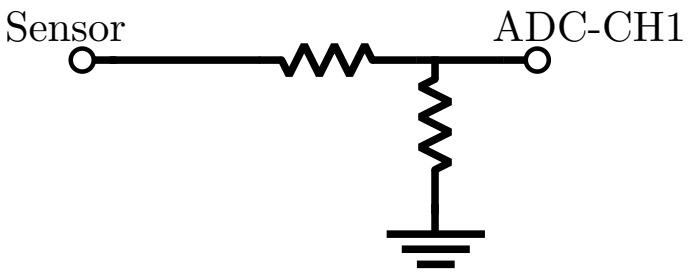


Figura 6: Conexión entre la salida del sensor y el canal 1 del ADC.

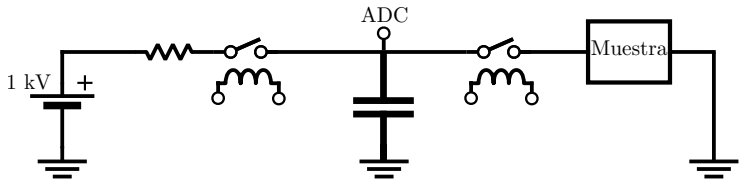
3.7. Relays

Para llevar a cabo el control sobre el ensayo, se necesitó de una interfaz entre el microcontrolador y los circuito de carga y descarga del banco de capacitores. Esta interfaz se compuso combinando *relays* con un *array* de transistores *Darlington*, *ULN2003A*. Este integrado es capaz de entregar la corriente necesaria para activar la bobina de un *relay*.

Transcurrido el tiempo de reposo de la muestra bajo presión, el microcontrolador activa al *relay* que conecta el banco de capacitores a la fuente de 1kV. A partir de ese momento, se espera a que la lectura del conversor A/D determine que se ha alcanzado la tensión necesaria, para luego aislar al banco de la fuente y, mediante otro *relay*, conectar al banco de capacitores con el compartimiento de la muestra.

Si bien es posible realizar un diseño que sólo utilice un *relay* para alternar los dos circuitos, se optó por la implementación de dos por separado de forma tal que la electrónica asociada a la carga los capacitores sea totalmente independiente de la circuitería de descarga.

A continuación, se puede apreciar un esquema superficial de la disposición de los elementos relevantes a lo explicado anteriormente:



3.8. Costos

El hardware que compone al proyecto puede separase en tres categorías, según naturaleza y procedencia:

3.8.1. Hardware Digital

La principal competencia de este informe y a lo que se avoca esta sección, consiste en todo el hardware ubicado en la placa principal del equipo y la placa auxiliar de *relays*.

Nro. de Parte	Descripción	Costo
AT89S52	Microcontrolador	$ 7
74HC138	Decodificador 3 a 8	$ 1,2
ADC0834	Conversor A/D 8 bits	$ 27,60
Winstar - WG12864A-PMI-V	Display Gráfico	$ 150
AT24C256	Memoria ROM	$ 9
HJR 1-2C	Relays	$ 5,7 × 2
74HC244N	Buffer	$ 1,2 × 2
Accesorios	Placa, cables, conectores, resitores, capacitores, etc	$ 40

3.8.2. Hardware Analógico

Una parte del aparejo experimental bajo el control del microcontrolador corresponde a elementos analógicos. Si bien una discusión detallada de estos escapa al ámbito de la materia y el presente informe sus costos se detallan a continuación.

Descripción	Costo
Banco de Capacitores	$ 5000
Tiristor ST700CL	$ 150
Tansformador 220 V- 1 kV- 30 mA	$ 172
Tansmisor de presión	$ 440

3.8.3. Equipamiento preexistente del laboratorio

Existe equipamiento utilizado durante la experiencia, esencial para ésta, que debe contabilizarse como un recurso preexistente del **Laboratorio de Sólidos Amorfos**.

Este es el caso de la prensa hidráulica, que forma parte del patrimonio del Laboratorio desde hace años, volviéndose casi imposible asignarle un costo.

4. Software

4.1. Capas de software

En la figura siguiente, puede apreciarse un diagrama de las capas que separan al usuario del hardware.

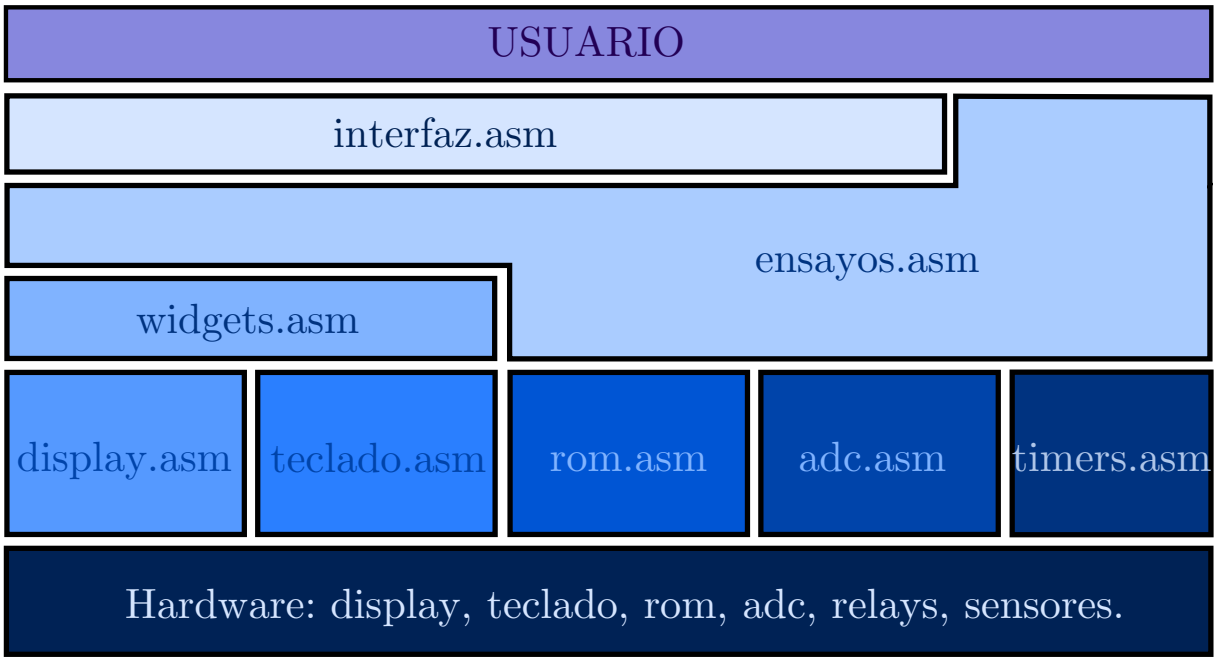


Figura 7: Software en capas.

4.2. Interfaz con el usuario

La interfaz con el usuario fue pensada en un nivel de abstracción más alto en el cual se pudieran ir enunciando, a modo de guión, una a una las acciones a llevar a cabo sin tener la necesidad de adentrarse mucho en los detalles de bajo nivel. Éste ideal se logró en mayor y menor medida en distintas partes de dicha descripción.

Los componentes contenedores de la interfaz son: el **menú principal**, el **anillo de ensayos** y el **menú de edición de parámetros**. Ambos menús fueron implementados utilizando a la rutina genérica `WIDGETS_MENU_INTERACT:`, cuya tarea es dibujar las cadenas de texto de cada opción e interpretar la presión de las teclas especiales, para permitir al usuario navegarlas y elegir la deseada.

Desde el menú principal se pueden realizar las siguientes tareas: ingresar al **anillo de ensayos**, agregar un nuevo ensayo a la lista y, dar inicio a la ejecución en serie de todos los ensayos cargados.

El **anillo de ensayos** no es más que una lista circular que en cada posición muestra en pantalla a todos los parámetros del ensayo seleccionado y ofrece las siguientes opciones para aplicarle: **editar sus parámetros**, **ejecutarlo** y, finalmente, **eliminarlo de la lista**[2]

Tanto desde la opción para agregar un ensayo del menú principal como desde la opción para editar los parámetros de un ensayo, se entrega el control de la aplicación a la rutina `INTERFAZ_MENU_EDITAR_ENSAYO:`[3]. El **menú de edición de parámetros** ofrece una opción por cada uno que cuando es elegida, muestra un cursor a la espera de que el usuario ingrese el valor numérico que quiere asignarle al parámetro. Hay una última opción en este menú, llamada **Grabar ensayo**, que al ser elegida, muestra una pantalla con un resumen de todos los valores ingresados y le permite al usuario confirmar si realmente desea grabarlos o volver a editarlos.

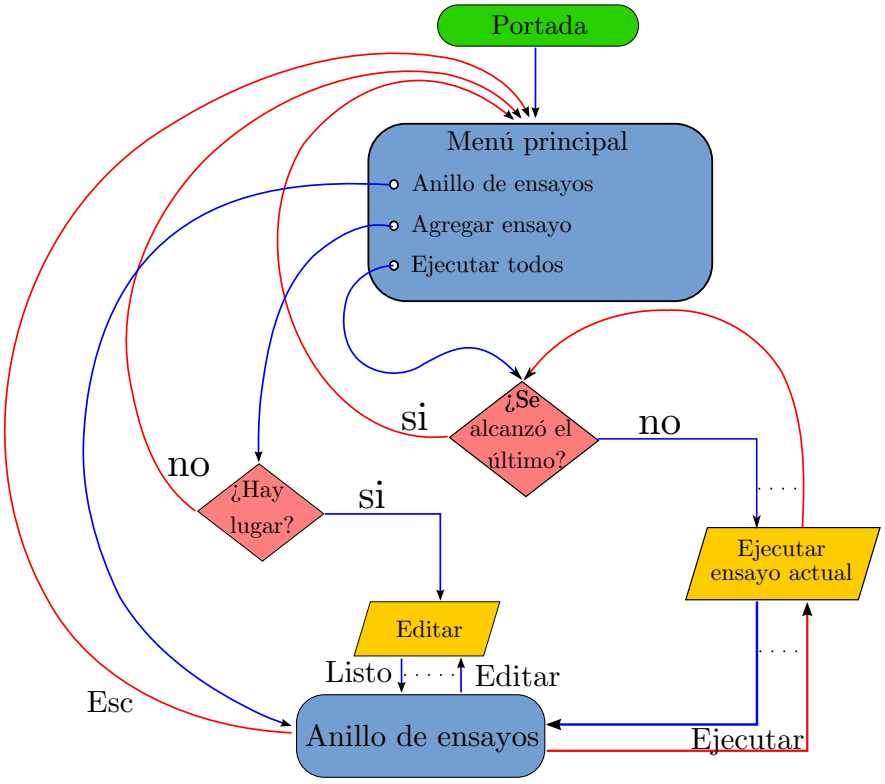


Figura 8: Diagrama de flujo.

[2]Ésto en realidad fue implementado intercambiando al que se desea eliminar con el último, y luego reduciendo la lista.

[3]En el caso de la opción para agregar un ensayo, antes de entregar el control se hacen las verificacione pertinentes para asegurar la disponibilidad de espacio en memoria ROM.

4.3. Administración de ensayos

La administración de los ensayos se compone de un conjunto de rutinas ubicadas en el archivo `ensayos.asm` que permiten: cargar y grabar desde y a la memoria ROM los parámetros de un ensayo y toda la información contextual necesaria[4], eliminar un ensayo de la lista, y ejecutar ensayos en serie o aisladamente.

La estrategia elegida a la hora de organizar los sectores de memoria ROM a utilizar fue la siguiente: se impuso a 255 como cantidad máxima de ensayos, donde cada uno debería poder definirse en un máximo de 8 *bytes*, y se reservaron 8 *bytes* más para la información contextual. De esta manera, se necesitó un total de 2 *kbytes*, lo que equivale a 32 de las 512 páginas de 64 *bytes* de la memoria ROM. Se decidió ubicar este bloque en el final de la memoria, es decir, a partir de la página 480, cuya primera dirección se definió como símbolo bajo el nombre de `ENSAYOS_BASE_ROM_ADDRESS`, y se calculó de la siguiente manera:

$$\left(\boxed{512} - \boxed{32}\right) \times \boxed{64} = 30720 = 0\text{x}7800$$

- Cantidad total de páginas
- Cantidad necesaria
- *Bytes* por página

Una vez definido ese símbolo, definir a la dirección de la información de contexto fue trivial y se obtuvo sumando 255×8 a la dirección anterior.

4.3.1. Almacenamiento de los parámetros de un ensayo

Como se dijo anteriormente, los parámetros que definen un ensayo son: la **tensión de carga** del banco de capacitores, la **presión mecánica** sobre la muestra, el **tiempo de reposo** en el que la muestra debe permanecer bajo presión y el **tiempo de la descarga**.

El valor máximo de tensión al que se pueden cargar los capacitores es 1kV. En principio podría parecer que con 1 *byte* no alcanza para almacenar todos los valores posibles, pero hay una limitación impuesta por el conversor A/D de 8 *bits* que hace que no sea necesario almacenar el valor explícito de tensión; sólo es necesario hacer una conversión que transforme proporcionalmente al intervalo $[0; 1000\text{V}]$ en el intervalo $[0; 255]$.

Un razonamiento análogo se siguió para almacenar el valor de presión mecánica: se almacenó el *byte* que finalmente esperará el conversor AD, previa conversión del intervalo $[0; 20\,\text{Tn}]$.

El **tiempo de reposo**, expresado en segundos, es el único parámetro que requiere ser almacenado en 2 *bytes*. Esto permite intervalos de hasta 18 horas, 12 minutos y 16 segundos, harto suficientes para satisfacer las necesidades de los investigadores que finalmente utilizarán el dispositivo.

El **tiempo de descarga**, expresado en milisegundos, pudo ser almacenado en 1 *byte* ya que 255 es un máximo aceptable para los experimentos.

[4] Actualmente el contexto sólo consiste en la cantidad de ensayos cargados, pero está dentro de los planes a futuro incluir más información a medida que sea necesaria.

4.3.2. Ejecución de ensayos

Al igual que a la hora de diseñar la interfaz, la rutina de ejecución de ensayos se pensó en un nivel más alto en el cual se pudieran ir listando las órdenes y acciones dirigidas al hardware de manera casi guionada, sin entrar demasiado en los detalles de bajo nivel.

Puede apreciarse en la rutina `ENSAYOS_REALIZAR_ACTUAL:`, ubicada en la biblioteca `ensayos.asm`, como, mediante la combinación de macros y rutinas de alto nivel se alcanzó en mayor y menor medida este objetivo. Esta rutina, puede descomponerse superficialmente en las siguientes etapas:

- Carga desde memoria ROM de los parámetros del ensayo a realizar.
- Solicitud por pantalla al operario para que indique cuando la muestra ya está lista en el compartimiento.
- Solicitud por pantalla al operario para que, valiéndose de la prensa hidráulica, alcance el valor de presión mecánica que el ensayo requiere. Se acompaña la información con una barra que indica visualmente el porcentaje alcanzado. El proceso no continua hasta que el operario alcance el valor y lo confirme por teclado.
- Comienzo del tiempo de espera durante el cual la muestra debe permanecer en reposo.
- Inicio de la carga del banco de capacitores.
- Al alcanzarse la tensión requerida por el ensayo, se detiene la carga y se envía la orden de cierre a la llave que cierra el circuito que forman el banco de capacitores con la muestra en el compartimiento.
- Aviso al operario para que retire la muestra.

5. Conclusión

Finalmente y en primera persona, queremos finalizar con algunas apreciaciones sobre el resultado final, las cosas que quedaron en el tintero, las dificultades encontradas y los planes a futuro.

No hubo un punto particular al cual podamos atribuirle las principales dificultades; han estado distribuidas de manera muy uniforme. Comenzamos la materia con un conocimiento prácticamente nulo y cada pequeña nueva pieza de conocimiento nos consumió tiempos apreciables. Prohibirnos hacer las cosas sin antes entenderlas, fue una decisión que se pagó con mucho tiempo y esfuerzo.

Quizá de todas las cosas que quedaron afuera, el no haber podido acceder a una prensa para controlar desatendidamente sea la más importante, pues nos obligó a hacer que el dispositivo dependa mucho de la asistencia humana.

Una vez fundido el material magnético mediante la técnica de *Spark Plasma*, resta magnetizarlo para obtener finalmente un imán. Dentro de los planes a futuro se encuentra la expansión del dispositivo para poder controlar también esa etapa de los ensayos valiéndose de una bobina y posiblemente la misma fuente de tensión.

Comparando los objetivos con el resultado final, podemos decir que hemos llegado a implementar prácticamente toda la lógica necesaria. Inicialmente, también deseábamos finalizar la implantación definitiva junto a la prensa hidráulica, la fuente de 1 kV y el banco de capacitores, pero por falta de tiempo y otras dificultades, no nos fue posible. Más allá de eso, pudimos verificar la correcta comunicación con el sensor de presión y hemos diseñado todos los circuitos de la fuente (realizamos una primera versión para probar a 220 V) y la etapa del disparo mediante un tiristor adecuado para soportar las magnitudes que implica una tensión de 1 kV. Cabe aclarar que, mientras no que tengamos en nuestras manos a los primeros imanes producidos con esta técnica, continuaremos trabajando junto a la gente del **Laboratorio de Sólidos Amorfos**. Éste ha sido simplemente el comienzo.

Buenos Aires,
16 de julio de 2008.

6. Código fuente

6.1. main.asm

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Inicio del programa.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Se incluyen las macros primero:
$INCLUDE(macros.asm)


DSEG        at      020h


CSEG
    ORG     0000H           ; Reset.
    JMP     MAIN

    ORG     000BH
    JMP     TIMERS_T0ISR    ; Timer 0 Interrupt Service Rutine.

    ORG     001BH
    JMP     TIMERS_T1ISR    ; Timer 1 Interrupt Service Rutine.

    ORG     0030H


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Inclusión de librerías:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
$INCLUDE(timers.asm)            ; Control de timers y cronómetros.
$INCLUDE(hardware/decoder.asm)  ; Selección de dispositivos.
$INCLUDE(hardware/adc.asm)      ; Control del ADC.
$INCLUDE(hardware/display.asm)  ; Control del display.
$INCLUDE(hardware/keypad.asm)   ; Control del teclado.
$INCLUDE(hardware/rom.asm)      ; Control de la memoria ROM.
$INCLUDE(widgets.asm)           ; Rutinas para crear accesorios en pantalla.
$INCLUDE(ensayos.asm)           ; Rutinas para manipular ensayos.
$INCLUDE(interfaz.asm)          ; Bucles ad hoc que describen la interfaz.
$INCLUDE(screens.asm)           ; Grabación de pantallas gráficas a ROM.
$INCLUDE(i18n/i18n_sp.asm)      ; Internacionalización: castellano.
$INCLUDE(misc.asm)              ; Rutinas misceláneas.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

DSEG

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Luego de que todas las bibliotecas hayan reservado el espacio para
; todas sus variables, se "reserva" la última posición para poder
; aprovechar el resto de la RAM para la pila:
STACK_POINTER_START:  DS  1

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
CSEG

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MAIN:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Punto de inicio de del programa.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se apagan todos los puertos por seguridad:
    MOV     P0, #00h
    MOV     P1, #00h
    MOV     P2, #00h
    MOV     P3, #00h

    ; Se apunta al stack pointer a la última posición de RAM reservada:
    MOV     SP, #STACK_POINTER_START - 1

    ; Se activan globalmente las interrupciones y se desactivan
    ; todas a la espera de activaciones individuales:
    MOV     IE, #10000000b

    ; Se inicializan los timers:
    CALL    TIMERS_INIT

    ; Se inicializa al display:
    CALL    DISPLAY_INIT

    ; Se muestra una portada con scroll:
    MOV     rom_address_L,   #LOW(SCREENS_LOGOFIUBA_ADDRESS)
    MOV     rom_address_H,  #HIGH(SCREENS_LOGOFIUBA_ADDRESS)
    CALL    SCREENS_DUMP_TO_DISPLAY_FROM_ROM
    CALL    DISPLAY_SCROLL

    ; Se pasa el control de la aplicación al menú principal:
    JMP     INTERFAZ_MENU_PRINCIPAL
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

6.2. display.asm

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutinas de control del dispositivo: Winstar Display WG12864A-PMI-V
;; Dimensiones: 128x64 pixels en dos sectores de 64x64.
;;
;; Interface:
;; ----------
;;
;; ---------------------------------------------------------------------------
;; Pin | Symbol |      Level |                                    Description
;; ---------------------------------------------------------------------------
;;   1      Vss          0 V                                           Ground
;;   2      Vdd        5.0 V                         Supply voltage for logic
;;   3       Vo   (Variable)                        Operating voltage for LCD
;;   4      D/I          H/L                         H: Data, L : Instruction
;;   5      R/W          H/L   H: Read (MPU<-Module), L: Write (MPU->Module)
;;   6        E            H                                    Enable signal
;;   7      DB0          H/L                                       Data bit 0
;;   8      DB1          H/L                                       Data bit 1
;;   9      DB2          H/L                                       Data bit 2
;;  10      DB3          H/L                                       Data bit 3
;;  11      DB4          H/L                                       Data bit 4
;;  12      DB5          H/L                                       Data bit 5
;;  13      DB6          H/L                                       Data bit 6
;;  14      DB7          H/L                                       Data bit 7
;;  15      CS1            L                                Select Column 1~64
;;  16      CS2            L                              Select Column 65~128
;;  17      RES            L                                     Reset signal
;;  18     Vout            -                                 Negative Voltage
;;  19        A            -                Power Supply for LED backlight ( + )
;;  20        K            -                Power Supply for LED backlight ( - )
;; ---------------------------------------------------------------------------
;; ---------------------------------------------------------------------------
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Rutinas y macros públicas:
;; --------------------------
;;                          DISPLAY_MACRO_PUT_ASCII
;;                          DISPLAY_MACRO_PUT_STRING_STATIC
;;                          DISPLAY_MACRO_PUT_STRING_STATIC_INMEDIATE
;;                          DISPLAY_INIT
;;                          DISPLAY_ON
;;                          DISPLAY_OFF
;;                          DISPLAY_CLEAR
;;                          DISPLAY_SCROLL
;;                          DISPLAY_PUT_BYTE
;;                          DISPLAY_PUT_CHAR
;;                          DISPLAY_PUT_ASCII
;;                          DISPLAY_PUT_STRING_STATIC
;;                          DISPLAY_PUT_STRING_DYNAMIC
;;                          DISPLAY_PUT_SCREEN_FROM_CODE
;;
;; Requisitos de esta biblioteca:
;; ------------------------------
;;                              Existencia de una macro llamada
;;                              DISPLAY_MACRO_SELECT que garantice la
;;                              disponibilidad del display.
;;
;;                              Existencia de dos macros llamadas
;;                              START_SENSIBLE_OP y END_SENSIBLE_OP
;;                              para indicarle al exterior el comienzo y fin
;;                              de operaciones de I/O sensibles.
```

```
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Definiciones:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Conexionado:
DISPLAY_RESET                       EQU    P0.2
DISPLAY_SELECT_LEFT_SECTOR          EQU    P0.3
DISPLAY_SELECT_RIGHT_SECTOR         EQU    P0.4
DISPLAY_ENABLE                      EQU    P0.5
DISPLAY_RW_MODE                     EQU    P0.6
DISPLAY_DI_MODE                     EQU    P0.7
DISPLAY_DATABUS                     EQU    P2

; Códigos internos de las instrucciones básicas:
DISPLAY_INSTRUCTION_ON              EQU    00111111b    ; 3F
DISPLAY_INSTRUCTION_OFF             EQU    00111110b    ; 3E
DISPLAY_INSTRUCTION_PAGE_0          EQU    10111000b    ; B8
DISPLAY_INSTRUCTION_COLUMN_0        EQU    01000000b    ; 40
DISPLAY_INSTRUCTION_START_LINE_0    EQU    11000000b    ; C0

; Máscaras:
DISPLAY_STATUS_BUSY_MASK            EQU    10000000b    ; 80
DISPLAY_STATUS_OFF_MASK             EQU    00100000b    ; 20
DISPLAY_STATUS_RESET_MASK           EQU    00010000b    ; 10
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Macros de uso interno:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Activar modo de lectura.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_MACRO_READ_MODE MACRO
    SETB    DISPLAY_RW_MODE
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Activar modo de escritura.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_MACRO_WRITE_MODE MACRO
    CLR     DISPLAY_RW_MODE
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Activar modo de instrucciones.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_MACRO_INST_MODE MACRO
    CLR     DISPLAY_DI_MODE
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Activar modo de datos.
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_MACRO_DATA_MODE MACRO
    SETB    DISPLAY_DI_MODE
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Realizar un strobe.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_MACRO_STROBE_ENABLE MACRO
    SETB    DISPLAY_ENABLE
    NOP
    CLR     DISPLAY_ENABLE
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Enviar el contenido del acumulador como dato.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_MACRO_SEND_DATA MACRO
    ; Informa sobre el inicio de una operación sensible:
    START_SENSIBLE_OP

    ; Activa el modo de escritura de datos:
    DISPLAY_MACRO_WRITE_MODE
    DISPLAY_MACRO_DATA_MODE

    ; Envía al databus el contenido del acumulador y realiza un strobe:
    MOV     DISPLAY_DATABUS, A
    DISPLAY_MACRO_STROBE_ENABLE

    ; Informa sobre el término de una operación sensible:
    END_SENSIBLE_OP
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Enviar el contenido del acumulador como instrucción.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_MACRO_SEND_INSTRUCTION MACRO
    ; Informa sobre el inicio de una operación sensible:
    START_SENSIBLE_OP

    ; Activa el modo de escritura de instrucciones:
    DISPLAY_MACRO_WRITE_MODE
    DISPLAY_MACRO_INST_MODE

    ; Envía al databus el contenido del acumulador y realiza un strobe:
    MOV     DISPLAY_DATABUS, A
    DISPLAY_MACRO_STROBE_ENABLE

    ; Informa sobre el término de una operación sensible:
    END_SENSIBLE_OP
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_MACRO_SET_PAGE  MACRO
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Encapsula a la estructura típica para cambiar de página al puntero
;; del display. Recibe en el acumulador a la página (rango: [0;7]).
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ; Se suma la dirección base definida por el fabricante:
    ADD     A, #DISPLAY_INSTRUCTION_PAGE_0
    DISPLAY_MACRO_SEND_INSTRUCTION
ENDM

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_MACRO_SET_COLUMN  MACRO
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Encapsula a la estructura típica para cambiar de columna al puntero
;; del display. Recibe en el acumulador a la columna (rango: [0;63]).
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ; Se suma la dirección base definida por el fabricante:
    ADD     A, #DISPLAY_INSTRUCTION_COLUMN_0
    DISPLAY_MACRO_SEND_INSTRUCTION
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Macros públicas:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_MACRO_PUT_ASCII MACRO codigo, pagina, columna
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Encapsula a la típica estructura necesaria para mostrar en pantalla
;; una caracter a partir de su código ASCII.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Ejemplo de uso:
;; ---------------
;; DISPLAY_MACRO_PUT_ASCII 'F', 3d, 4d
;; (Muestra en la página 3 y columna 4 al caracter "F".)
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    MOV     display_data, #codigo
    MOV     display_page, #pagina
    MOV     display_column, #columna
    CALL    DISPLAY_PUT_ASCII
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_MACRO_PUT_STRING_STATIC MACRO direccion, pagina, columna
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Encapsula a la típica estructura necesaria para mostrar en pantalla
;; una cadena ubicada en memoria de código.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Ejemplo de uso:
;; ---------------
;; MACRO_PUT_STRING_STATIC LABEL, 3d, 4d
;; (Muestra en la página 3 y columna 4 a la cadena ubicada en "LABEL".)
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    MOV     DPTR, #direccion
    MOV     display_page, #pagina
    MOV     display_column, #columna
```

```
    CALL    DISPLAY_PUT_STRING_STATIC
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_MACRO_PUT_STRING_STATIC_INMEDIATE MACRO cadena, pagina, columna
LOCAL   L1, L2  ; Definición local de las etiquetas para evitar colisiones.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Permite hacer lo mismo que MACRO_PUT_STRING_STATIC, pero sin necesidad de
;; haber definido anteriormente a la cadena en memoria de código.
;; Toma como parámetro un literal y define in situ a la cadena.
;; No es recomendable usarlo para cadenas repetidas, pues se duplica
;; innecesariamente el espacio utilizado.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Ejemplo de uso:
;; ---------------
;; MACRO_PUT_STRING_STATIC_INMEDIATE 'Texto a mostrar', 3d, 4d
;; (Muestra en la página 3 y columna 4 al literal especificado.)
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ; Es necesario saltar a la cadena antes de ejecutar el código:
    SJMP    L2

    ; Definición in situ de la cadena:
L1: DB      &cadena, 0d

    ; Una vez alcanzado este punto, se reutiliza a MACRO_PUT_STRING_STATIC:
L2: DISPLAY_MACRO_PUT_STRING_STATIC L1, pagina, columna
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


DSEG


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Variables de posicionamiento utilizadas como argumentos
;; de las rutinas a las que se les pide ubicar algo en pantalla:
display_page:       DS  1 ; Rango: [0;7]
display_column:     DS  1 ; Rango: [0;127] o [0;19] (depende del contexto)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Variable de propósito general utilizada como argumento
;; de las rutinas que procesan un byte (DISPLAY_PUT_BYTE y DISPLAY_PUT_ASCII):
display_data:       DS  1
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


CSEG


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_ENABLE_LEFT_SECTOR:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para seleccionar al sector izquierdo del display.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    START_SENSIBLE_OP
    CLR     DISPLAY_SELECT_LEFT_SECTOR
    SETB    DISPLAY_SELECT_RIGHT_SECTOR
```

```
    NOP
    END_SENSIBLE_OP
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_ENABLE_RIGHT_SECTOR:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para seleccionar al sector derecho del display.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    START_SENSIBLE_OP
    CLR     DISPLAY_SELECT_RIGHT_SECTOR
    SETB    DISPLAY_SELECT_LEFT_SECTOR
    NOP
    END_SENSIBLE_OP
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_INIT:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina que inicializa al display y lo limpia.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;              DISPLAY_MACRO_SELECT
;;              DISPLAY_MACRO_SEND_INST
;;              DISPLAY_ENABLE_RIGHT_SECTOR
;;              DISPLAY_ENABLE_LEFT_SECTOR
;;              DISPLAY_CLEAR
;;
;; Registros alterados:
;; ___________________
;;                      A, B, display_page
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Primero se ejecuta la macro que garantiza que el display esté disponible:
    DISPLAY_MACRO_SELECT

    ; Se retira al display del estado reset:
    SETB    DISPLAY_RESET

    CALL    DISPLAY_OFF
    CALL    DISPLAY_CLEAR
    CALL    DISPLAY_ON
    RET

DISPLAY_ON:
    ; Se envía la instrucción que prende al display:
    MOV     A, #DISPLAY_INSTRUCTION_ON
    SJMP    DISPLAY_ON_OFF_END
    ; No necesita RET aquí.

DISPLAY_OFF:
    ; Se envía la instrucción que apaga al display:
    MOV     A, #DISPLAY_INSTRUCTION_OFF
```

```
    ; No necesita RET aquí.

DISPLAY_ON_OFF_END:
    ; Finalmente, se envía la instrucción a ambos sectores:
    CALL    DISPLAY_ENABLE_LEFT_SECTOR
    DISPLAY_MACRO_SEND_INSTRUCTION

    CALL    DISPLAY_ENABLE_RIGHT_SECTOR
    DISPLAY_MACRO_SEND_INSTRUCTION

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_CLEAR:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para limpiar completamente a la pantalla.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;               DISPLAY_MACRO_SELECT
;;               DISPLAY_MACRO_SET_PAGE
;;               DISPLAY_MACRO_SET_COLUMN
;;               DISPLAY_MACRO_SEND_DATA
;;               DISPLAY_ENABLE_RIGHT_SECTOR
;;               DISPLAY_ENABLE_LEFT_SECTOR
;;
;;
;; Registros alterados:
;; ____________________
;;                       A, B
;;                       display_page
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Primero se ejecuta la macro que garantiza que el display esté disponible:
    DISPLAY_MACRO_SELECT

    ; Se llama dos veces a la misma rutina privada (DISPLAY_CLEAR_SECTOR)
    ; pero cambiando de sector:
    CALL    DISPLAY_ENABLE_LEFT_SECTOR
    CALL    DISPLAY_CLEAR_SECTOR
    CALL    DISPLAY_ENABLE_RIGHT_SECTOR
    CALL    DISPLAY_CLEAR_SECTOR

    RET

DISPLAY_CLEAR_SECTOR:

    ; Se utiliza a display_page para iterar desde 8 a 1
    ; El rango se desplaza para poder usar DJNZ.
    MOV     display_page, #8d

DISPLAY_CLEAR_LOOP_PAGES:

    ; Antes de seleccionar la página,
    ; debe corregirse el valor:
    MOV     A, display_page
    DEC     A
    DISPLAY_MACRO_SET_PAGE
```

```
    ; Siempre se empieza desde la columna 0:
    CLR     A
    DISPLAY_MACRO_SET_COLUMN

    ; Se utiliza a B para iterar:
    MOV     B, #64d

DISPLAY_CLEAR_LOOP_COLUMNS:
    CLR     A   ; Contenido nulo.
    DISPLAY_MACRO_SEND_DATA
    DJNZ    B, DISPLAY_CLEAR_LOOP_COLUMNS
    DJNZ    display_page, DISPLAY_CLEAR_LOOP_PAGES

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_SCROLL:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para desplazar el contenido de la pantalla desde abajo hacia arriba.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;              DISPLAY_MACRO_SELECT
;;              DISPLAY_MACRO_SEND_INST
;;              DISPLAY_ENABLE_RIGHT_SECTOR
;;              DISPLAY_ENABLE_LEFT_SECTOR
;;
;;
;; Registros alterados:
;; ____________________
;;                      A
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Primero se ejecuta la macro que garantiza que el display esté disponible:
    DISPLAY_MACRO_SELECT

    ; Se utiliza al acumulador para iterar tomando como base al código
    ; de la instrucción que especifica la línea de RAM del display
    ; mostrará al comienzo:
    MOV     A, #DISPLAY_INSTRUCTION_START_LINE_0

DISPLAY_SCROLL_LOOP:

    ; Se envía la instrucción a ambos sectores:
    CALL    DISPLAY_ENABLE_LEFT_SECTOR
    DISPLAY_MACRO_SEND_INSTRUCTION

    CALL    DISPLAY_ENABLE_RIGHT_SECTOR
    DISPLAY_MACRO_SEND_INSTRUCTION

    ; Delay para poder apreciar el efecto:
    CALL    DELAY_TEN_MSEC
    CALL    DELAY_TEN_MSEC
    CALL    DELAY_TEN_MSEC

    ; Se incrementa y compara con la última instrucción que se debe enviar:
    INC     A
    JB      Acc.6, DISPLAY_SCROLL_LOOP
    ; (cuando el sexto bit se apaga, es porque ya se recorrieron
```

```
    ; los 64 valores posibles)

    ; Antes de terminar, se asegura de dejar a la línea 0 como comienzo:
    MOV     A, #DISPLAY_INSTRUCTION_START_LINE_0
    CALL    DISPLAY_ENABLE_LEFT_SECTOR
    DISPLAY_MACRO_SEND_INSTRUCTION
    CALL    DISPLAY_ENABLE_RIGHT_SECTOR
    DISPLAY_MACRO_SEND_INSTRUCTION
RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_PUT_BYTE:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para mostrar un byte a una posición específica del display.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;              DISPLAY_MACRO_SELECT
;;              DISPLAY_MACRO_SET_PAGE
;;              DISPLAY_MACRO_SET_COLUMN
;;              DISPLAY_MACRO_SEND_DATA
;;              DISPLAY_ENABLE_RIGHT_SECTOR
;;              DISPLAY_ENABLE_LEFT_SECTOR
;;
;; Requisitos:
;; ___________
;;              El byte a mostrar en display_data.
;;              display_column entre 0 y 127.
;;              display_page entre 0 y 7.
;;
;; Registros alterados:
;; ____________________
;;                      A, C
;;                      display_column <- display_column + 1
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Primero se ejecuta la macro que garantiza que el display esté disponible:
    DISPLAY_MACRO_SELECT

    ; Se analiza a qué sector del display hay que mostrar al byte:
    MOV     A, display_column
    ; Se utiliza a CJNE sólo para leer luego el Carry (C = A < 64).
    ; Con ($+3) se saltan los 3 bytes que ocupa "CJNE A, #data, rel"
    ; logrando así alcanzar siempre a la instrucción que sigue debajo.
    CJNE    A, #64d, ($+3)
    JC      DISPLAY_PUT_BYTE_LEFT

    ; Si C = 0, corresponde entonces volcar al sector derecho.

    ; Se resta la columna del medio para obtener finalmente
    ; la columna en donde se mostrará:
    SUBB    A, #64d

    CALL    DISPLAY_ENABLE_RIGHT_SECTOR
    SJMP    DISPLAY_PUT_BYTE_FINAL

    ; Si C = 1, corresponde entonces volcar al sector izquierdo:
DISPLAY_PUT_BYTE_LEFT:
    CALL    DISPLAY_ENABLE_LEFT_SECTOR
```

```
DISPLAY_PUT_BYTE_FINAL:

    ; Se llega aquí con la columna en A.
    DISPLAY_MACRO_SET_COLUMN

    MOV     A, display_page
    DISPLAY_MACRO_SET_PAGE

    MOV     A, display_data
    DISPLAY_MACRO_SEND_DATA

    ; Se incrementa para imitar al comportamiento del Display:
    INC     display_column

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_PUT_CHAR:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para mostrar un caracter en pantalla a partir de su codificación
;; en memoria de código (ver tabla—ascii.asm para más información).
;; Se imita a un display de 20x8 caracteres de 8x6 bits.
;; Se deja un margen de cuatro bits en cada lateral para simplificar
;; los algoritmos.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ———————————
;;              DISPLAY_MACRO_SELECT
;;              DISPLAY_MACRO_SET_PAGE
;;              DISPLAY_MACRO_SET_COLUMN
;;              DISPLAY_MACRO_SEND_DATA
;;              DISPLAY_ENABLE_RIGHT_SECTOR
;;              DISPLAY_ENABLE_LEFT_SECTOR
;;
;; Requisitos:
;; ———————————
;;              La dirección del primer byte del caracter en DPTR.
;;              display_column entre 0 y 19
;;              display_page entre 0 y 7
;;
;; Registros alterados:
;; ————————————————————
;;                      A, B, C, DPTR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Primero se ejecuta la macro que garantiza que el display esté disponible:
    DISPLAY_MACRO_SELECT

    ; Se utiliza a B para multiplicar por el ancho de cada caracter:
    MOV     B, #6d

    ; Se analiza a qué sector del display hay que volcar el caracter:
    MOV     A, display_column
    ; Se utiliza a CJNE sólo para leer luego el Carry (C = A < 10).
    ; Con ($+3) se saltan los 3 bytes que ocupa "CJNE A, #data, rel"
    ; logrando así alcanzar siempre a la instrucción que sigue debajo.
    CJNE    A, #10d, ($+3)
    JC      DISPLAY_PUT_CHAR_LEFT
```

```
    ; Si C = 0, corresponde entonces volcar al sector derecho.

    ; Se resta la columna del medio y se multiplica por el ancho
    ; de cada caracter para obtener finalmente la columna en donde
    ; comenzará el volcado:
    SUBB    A, #10d
    MUL     AB

    CALL    DISPLAY_ENABLE_RIGHT_SECTOR
    SJMP    DISPLAY_PUT_CHAR_DUMP

    ; Si C = 1, corresponde entonces volcar al sector izquierdo:
DISPLAY_PUT_CHAR_LEFT:
    ; Se multiplica por el ancho y luego se suma un offset fijo:
    MUL     AB
    ADD     A, #4d
    CALL    DISPLAY_ENABLE_LEFT_SECTOR

DISPLAY_PUT_CHAR_DUMP:

    ; En A se encuentra la columna en donde comienza el volcado.
    DISPLAY_MACRO_SET_COLUMN

    ; La página ha sido especificada desde afuera de la rutina.
    MOV     A, display_page
    DISPLAY_MACRO_SET_PAGE

    ; Se reutiliza a B para iterar los bytes del caracter:
    MOV     B, #5d

DISPLAY_PUT_CHAR_SEND_DATA_LOOP:

    ; Se obtienen uno a uno los bytes del caracter:
    CLR     A
    MOVC    A, @A+DPTR

    ; Se envían a la pantalla:
    DISPLAY_MACRO_SEND_DATA

    INC     DPTR
    DJNZ    B, DISPLAY_PUT_CHAR_SEND_DATA_LOOP

    ; Luego de volcar todos los bytes del caracter, se agrega un byte
    ; que oficia de espacio:
    CLR     A
    DISPLAY_MACRO_SEND_DATA

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_PUT_ASCII:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para mostrar un caracter en pantalla a partir de su código ASCII.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;              DISPLAY_PUT_CHAR
;;
```

```
;; Requisitos:
;; ------------
;;              El código ASCII en display_data. Rango: (ver en tabla_ascii.asm
;;                                                       los implementados).
;;              display_column entre 0 y 19
;;              display_page entre 0 y 7
;;
;; Registros alterados:
;; ---------------------
;;                      A, B, C, DPTR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    MOV     A, display_data

    ; Dirección del primer caracter implementado:
    MOV     DPTR, #TABLA_ASCII_BASE

    ; Se debe restar TABLA_ASCII_START pues la primera parte del standard ASCII
    ; no tiene representación gráfica. En consecuencia, la iteración que
    ; reposiciona a DPTR debe hacerse menos veces.
    CLR     C
    SUBB    A, #TABLA_ASCII_START
    ; Si la resta da cero, no es necesario reposicionar al DPTR:
    JZ      DISPLAY_PUT_ASCII_NO_OFFSET
DISPLAY_PUT_ASCII_DPTR_OFFSET:
    INC     DPTR
    INC     DPTR
    INC     DPTR
    INC     DPTR
    INC     DPTR
    DJNZ    Acc, DISPLAY_PUT_ASCII_DPTR_OFFSET

DISPLAY_PUT_ASCII_NO_OFFSET:
    ; Se llama a DISPLAY_PUT_CHAR una vez que el DPTR ha sido posicionado
    ; en la dirección del caracter deseado:
    CALL    DISPLAY_PUT_CHAR
DISPLAY_PUT_ASCII_END:
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_PUT_STRING_STATIC:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para mostrar una cadena ASCII ubicada en memoria de código.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ------------
;;              DISPLAY_PUT_ASCII
;;
;; Requisitos:
;; ------------
;;              La ubicación del primer caracter en DPTR.
;;              display_column
;;              display_page
;;
;; Registros alterados:
;; ---------------------
;;                      display_column <- última columna utilizada.
;;                      display_data
;;                      A
```

```
;;                      DPTR <- dirección del fin de línea.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    CLR     A
    MOVC    A, @A+DPTR  ; A <- Código ASCII del caracter.

    ; Si el código es 0, se está simbolizando un fin de línea,
    ; por lo que no hay que hacer nada:
    JZ      DISPLAY_PUT_STRING_STATIC_END

    ; Se resguarda la dirección de la cadena.
    ; porque DISPLAY_PUT_ASCII sobrescribe a DPTR:
    PUSH    DPL
    PUSH    DPH

    MOV     display_data, A
    CALL    DISPLAY_PUT_ASCII
    INC     display_column

    ; Se recupera la dirección de la cadena:
    POP     DPH
    POP     DPL

    ; Se llama a sí misma pero con el DPTR incrementado.
    INC     DPTR
    JMP     DISPLAY_PUT_STRING_STATIC

DISPLAY_PUT_STRING_STATIC_END:
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_PUT_STRING_DYNAMIC:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para mostrar una cadena ASCII ubicada en memoria de datos interna.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;              DISPLAY_PUT_ASCII
;;
;; Requisitos:
;; ___________
;;              La ubicación del primer caracter en R0.
;;
;; Registros alterados:
;; ____________________
;;                      display_column <- última columna utilizada.
;;                      display_data
;;                      A
;;                      R0 <- dirección del fin de línea.
;;                      DPTR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    MOV     A, @R0  ; A <- Código ASCII del caracter.

    ; Si el código es 0, se está simbolizando un fin de línea,
    ; por lo que no hay que hacer nada:
    JZ      DISPLAY_PUT_STRING_DYNAMIC_END

    MOV     display_data, A
    CALL    DISPLAY_PUT_ASCII
    INC     display_column
```

```
    ; Se llama a sí misma pero con R0 incrementado.
    INC     R0
    JMP     DISPLAY_PUT_STRING_DYNAMIC

DISPLAY_PUT_STRING_DYNAMIC_END:
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_PUT_SCREEN_FROM_CODE:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para volcar una pantalla entera desde memoria de código.
;; Se empieza volcando al sector izquierdo los primeros 512 bytes
;; desde la página 0 hasta la 7 y desde la columna 0 hasta la 63.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;              DISPLAY_OFF
;;              DISPLAY_ON
;;              DISPLAY_PUT_BYTE
;;
;; Requisitos:
;; ___________
;;              En DPTR la dirección del primero de los 1024 bytes.
;;
;; Registros alterados:
;; ____________________
;;                      DPTR <- dirección del último byte.
;;                      A, B, display_column, display_page
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    CALL    DISPLAY_OFF

    ; Debido a que hay que volcar 512 bytes a cada sector por separado,
    ; se utiliza a B para indicarle a la subrutina interna en cuál de
    ; los dos sectores debe iterar:
    MOV     B, #64d
    CALL    DISPLAY_PUT_SCREEN_FROM_CODE_SECTOR
    MOV     B, #128d
    CALL    DISPLAY_PUT_SCREEN_FROM_CODE_SECTOR

    CALL    DISPLAY_ON
    RET

DISPLAY_PUT_SCREEN_FROM_CODE_SECTOR:
    MOV     display_page, #0d
DISPLAY_PUT_SCREEN_FROM_CODE_LOOP_PAGES:
    ; Si B =  64, entonces display_column = 0.
    ; Si B = 128, entonces display_column = 64.
    MOV     A, B
    CLR     C
    SUBB    A, #64d
    MOV     display_column, A
DISPLAY_PUT_SCREEN_FROM_CODE_LOOP_COLUMNS:
    CLR     A
    MOVC    A, @A + DPTR
    MOV     display_data, A
    CALL    DISPLAY_PUT_BYTE
    INC     DPTR
```

```
    MOV     A, display_column
    CJNE    A, B, DISPLAY_PUT_SCREEN_FROM_CODE_LOOP_COLUMNS

    INC     display_page
    MOV     A, display_page
    CJNE    A, #8d, DISPLAY_PUT_SCREEN_FROM_CODE_LOOP_PAGES

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TABLA_ASCII_BASE:
$INCLUDE(tabla_ascii.asm)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

6.3. tabla-ascii.asm

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Símbolos de la tabla ASCII.
;; Cada uno ocupa 5 bytes.
;; Para simplificar los algoritmos, es necesario que no haya "huecos".
;;
;; Formato:
;; ┌─────┬──────┐
;;     |  Byte
;; ┌─────┼──────┐
;; Bit | 12345
;; ┌─────┼──────┐
;;   0 | 11111
;;   1 | 10000
;;   2 | 10000
;;   3 | 11110
;;   4 | 10000
;;   5 | 10000
;;   6 | 10000
;;   7 | 00000 ──> la última fila se usa como espacio vertical.
;; └─────┴──────┘
;;
;; 1: 01111111
;; 2: 00001001
;; 3: 00001001
;; 4: 00001001
;; 5: 00000001

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; DEFINICIONES:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; Mediante este valor, se indica el código del primer caracter implementado.
;; Permite que las rutinas sepan cuanto deben desplazar a DPTR para alcanzar
;; el caracter deseado.
TABLA_ASCII_START   EQU 32d

CSEG
CHR_032:    DB  00000000b, 00000000b, 00000000b, 00000000b, 00000000b   ; Space
CHR_033:    DB  00000000b, 00000000b, 10111111b, 00000000b, 00000000b   ; !
CHR_034:    DB  00000000b, 00000111b, 00000000b, 00000111b, 00000000b   ; "
CHR_035:    DB  00100100b, 01111110b, 00100100b, 01111110b, 00100100b   ; #
CHR_036:    DB  00100110b, 01001001b, 01111111b, 01001001b, 00110010b   ; $
CHR_037:    DB  01001100b, 00101100b, 00010000b, 01101000b, 01100100b   ; %
CHR_038:    DB  00110000b, 01001110b, 01001001b, 01001010b, 00110000b   ; &
CHR_039:    DB  00000000b, 00000000b, 00000111b, 00000000b, 00000000b   ; '
CHR_040:    DB  00000000b, 00000000b, 00011100b, 00100010b, 01000001b   ; (
CHR_041:    DB  01000001b, 00100010b, 00011100b, 00000000b, 00000000b   ; )
CHR_042:    DB  00101010b, 00011100b, 00111110b, 00011100b, 00101010b   ; *
CHR_043:    DB  00001000b, 00001000b, 00111110b, 00001000b, 00001000b   ; +
CHR_044:    DB  10000000b, 01110000b, 00000000b, 00000000b, 00000000b   ; ,
CHR_045:    DB  00001000b, 00001000b, 00001000b, 00001000b, 00001000b   ; —
CHR_046:    DB  01100000b, 01100000b, 00000000b, 00000000b, 00000000b   ; .
CHR_047:    DB  01100000b, 00110000b, 00011000b, 00001100b, 00000110b   ; /
CHR_048:    DB  00000000b, 00111110b, 01001001b, 01000101b, 00111110b   ; 0
CHR_049:    DB  00000000b, 01000010b, 01111111b, 01000000b, 00000000b   ; 1
CHR_050:    DB  00000000b, 01100010b, 01010001b, 01001001b, 01000110b   ; 2
CHR_051:    DB  00000000b, 00100010b, 01001001b, 01001001b, 00110110b   ; 3
CHR_052:    DB  00000000b, 00001111b, 00001000b, 00001000b, 01111111b   ; 4
CHR_053:    DB  00000000b, 00100111b, 01000101b, 01000101b, 00111001b   ; 5
CHR_054:    DB  00000000b, 00111110b, 01001001b, 01001001b, 00110010b   ; 6
CHR_055:    DB  00000001b, 00000001b, 01111001b, 00000101b, 00000011b   ; 7
CHR_056:    DB  00000000b, 00110110b, 01001001b, 01001001b, 00110110b   ; 8
CHR_057:    DB  00000000b, 00100110b, 01001001b, 01001001b, 00111110b   ; 9
```

```
CHR_058:    DB  01101100b, 01101100b, 00000000b, 00000000b, 00000000b   ; :
CHR_059:    DB  10001100b, 01101100b, 00000000b, 00000000b, 00000000b   ; ;
CHR_060:    DB  00001000b, 00010100b, 00100010b, 01000001b, 00000000b   ; <
CHR_061:    DB  00010100b, 00010100b, 00010100b, 00010100b, 00010100b   ; =
CHR_062:    DB  00000000b, 01000001b, 00100010b, 00010100b, 00001000b   ; >
CHR_063:    DB  00000110b, 01010001b, 00001001b, 00000110b, 00000000b   ; ?
CHR_064:    DB  00111110b, 01000001b, 01011101b, 01010101b, 00101110b   ; @
CHR_065:    DB  01111110b, 00001001b, 00001001b, 00001001b, 01111110b   ; A
CHR_066:    DB  01111111b, 01001001b, 01001001b, 01001001b, 00110110b   ; B
CHR_067:    DB  00111110b, 01000001b, 01000001b, 01000001b, 00100010b   ; C
CHR_068:    DB  01111111b, 01000001b, 01000001b, 01000001b, 00111110b   ; D
CHR_069:    DB  01111111b, 01001001b, 01001001b, 01001001b, 01000001b   ; E
CHR_070:    DB  01111111b, 00001001b, 00001001b, 00001001b, 00000001b   ; F
CHR_071:    DB  00111110b, 01000001b, 01001001b, 01001001b, 00111010b   ; G
CHR_072:    DB  01111111b, 00001000b, 00001000b, 00001000b, 01111111b   ; H
CHR_073:    DB  00000000b, 00000000b, 01111111b, 00000000b, 00000000b   ; I
CHR_074:    DB  00000000b, 00110000b, 01000000b, 00111111b, 00000000b   ; J
CHR_075:    DB  01111111b, 00001000b, 00010100b, 00100010b, 01000001b   ; K
CHR_076:    DB  01111111b, 01000000b, 01000000b, 01000000b, 01000000b   ; L
CHR_077:    DB  01111111b, 00000010b, 00000100b, 00000010b, 01111111b   ; M
CHR_078:    DB  01111111b, 00000100b, 00001000b, 00010000b, 01111111b   ; N
CHR_079:    DB  00111110b, 01000001b, 01000001b, 01000001b, 00111110b   ; O
CHR_080:    DB  01111111b, 00001001b, 00001001b, 00001001b, 00000110b   ; P
CHR_081:    DB  00111110b, 01000001b, 01010001b, 01100001b, 00111110b   ; Q
CHR_082:    DB  01111111b, 00001001b, 00011001b, 00100110b, 01000000b   ; R
CHR_083:    DB  00100110b, 01001001b, 01001001b, 01001001b, 00110010b   ; S
CHR_084:    DB  00000001b, 00000001b, 01111111b, 00000001b, 00000001b   ; T
CHR_085:    DB  00111111b, 01000000b, 01000000b, 01000000b, 00111111b   ; U
CHR_086:    DB  00011111b, 00100000b, 01000000b, 00100000b, 00011111b   ; V
CHR_087:    DB  01111111b, 00100000b, 00010000b, 00100000b, 01111111b   ; W
CHR_088:    DB  01100011b, 00010100b, 00001000b, 00010100b, 01100011b   ; X
CHR_089:    DB  00000011b, 00000100b, 01111000b, 00000100b, 00000011b   ; Y
CHR_090:    DB  01100001b, 01010001b, 01001001b, 01000101b, 01000011b   ; Z

CHR_091:            ; [
DB  00000000b
DB  00000000b
DB  01111111b
DB  01000001b
DB  01000001b

CHR_092:            ; \
DB  00000110b
DB  00001100b
DB  00011000b
DB  00110000b
DB  01100000b

CHR_093:            ; ]
DB  01000001b
DB  01000001b
DB  01111111b
DB  00000000b
DB  00000000b

CHR_094:            ; ^
DB  00000100b
DB  00000010b
DB  00000001b
DB  00000010b
DB  00000100b

CHR_095:            ; _
DB  01000000b
DB  01000000b
DB  01000000b
```

```
DB  01000000b
DB  01000000b

CHR_096:        ; `
DB  00000010b
DB  00000100b
DB  00000000b
DB  00000000b
DB  00000000b

CHR_097:        ; a
DB  00000000b
DB  00100000b
DB  01010100b
DB  01010100b
DB  01111000b

CHR_098:        ; b
DB  00000000b
DB  01111110b
DB  01001000b
DB  01001000b
DB  00110000b

CHR_099:        ; c
DB  00000000b
DB  00111000b
DB  01000100b
DB  01000100b
DB  00101000b

CHR_100:        ; d
DB  00000000b
DB  00110000b
DB  01001000b
DB  01001000b
DB  01111110b

CHR_101:        ; e
DB  00000000b
DB  00111000b
DB  01010100b
DB  01010100b
DB  01011000b

CHR_102:        ; f
DB  00000000b
DB  00010000b
DB  01111100b
DB  00010010b
DB  00000010b

CHR_103:        ; g
DB  00000000b
DB  10011000b
DB  10100100b
DB  10100100b
DB  01111100b

CHR_104:        ; h
DB  00000000b
DB  01111110b
DB  00010000b
DB  00001000b
DB  01110000b
```

```
CHR_105:        ; i
DB  00000000b
DB  00000000b
DB  01111010b
DB  00000000b
DB  00000000b

CHR_106:        ; j
DB  00000000b
DB  10000000b
DB  01111010b
DB  00000000b
DB  00000000b

CHR_107:        ; k
DB  00000000b
DB  01111110b
DB  00010000b
DB  00101000b
DB  01000100b

CHR_108:        ; l
DB  00000000b
DB  00000000b
DB  00000000b
DB  01111111b
DB  00000000b


CHR_109:        ; m
DB  01111100b
DB  00000100b
DB  01111000b
DB  00000100b
DB  01111000b

CHR_110:        ; n
DB  00000000b
DB  01111100b
DB  00001000b
DB  00000100b
DB  01111000b

CHR_111:        ; o
DB  00000000b
DB  00111000b
DB  01000100b
DB  01000100b
DB  00111000b

CHR_112:        ; p
DB  00000000b
DB  11111100b
DB  01000100b
DB  01000100b
DB  00111000b

CHR_113:        ; q
DB  00000000b
DB  00111000b
DB  01000100b
DB  01000100b
DB  11111100b

CHR_114:        ; r
DB  00000000b
```

```
DB  01111100b
DB  00001000b
DB  00000100b
DB  00000100b

CHR_115:        ; s
DB  00000000b
DB  01001000b
DB  01010100b
DB  01010100b
DB  00100100b

CHR_116:        ; t
DB  00000000b
DB  00000100b
DB  01111110b
DB  01000100b
DB  00000000b

CHR_117:        ; u
DB  00000000b
DB  00111100b
DB  01000000b
DB  00100000b
DB  01111100b

CHR_118:        ; v
DB  00000000b
DB  00011100b
DB  01100000b
DB  01100000b
DB  00011100b

CHR_119:        ; w
DB  01111100b
DB  01000000b
DB  00111100b
DB  01000000b
DB  00111100b

CHR_120:        ; x
DB  01000100b
DB  00101000b
DB  00010000b
DB  00101000b
DB  01000100b

CHR_121:        ; y
DB  10000000b
DB  10011100b
DB  01100000b
DB  00100000b
DB  00011100b

CHR_122:        ; z
DB  01000100b
DB  01100100b
DB  01010100b
DB  01001100b
DB  01000100b
CHR_123:        ; {
DB  00000000b
DB  00000000b
DB  00001000b
DB  00110110b
DB  01000001b
```

```
CHR_124:          ; |
DB   00000000b
DB   00000000b
DB   01111111b
DB   00000000b
DB   00000000b

CHR_125:          ; }
DB   01000001b
DB   00110110b
DB   00001000b
DB   00000000b
DB   00000000b

CHR_126:          ; ~
DB   00010000b
DB   00001000b
DB   00010000b
DB   00010000b
DB   00001000b

; Misceláneos:
CHR_EXCLO:       ; ¡
DB   00000000b
DB   00000000b
DB   11111101b
DB   00000000b
DB   00000000b

CHR_PREGO:       ; ¿
DB   00000000b
DB   01100000b
DB   10010000b
DB   10001010b
DB   01100000b

CHR_CURSOR:
DB   01111111b
DB   00000000b
DB   00000000b
DB   00000000b
DB   00000000b

CHR_DIAMOND:
DB   00001000b
DB   00011100b
DB   00111110b
DB   00011100b
DB   00001000b

CHR_RIGHT_ARROW:
DB   00001000b
DB   00001000b
DB   00111110b
DB   00011100b
DB   00001000b

CHR_LEFT_ARROW:
DB   00001000b
DB   00011100b
DB   00111110b
DB   00001000b
DB   00001000b

CHR_CAPACITOR:
```

```
DB  00001000b
DB  01111111b
DB  00000000b
DB  01111111b
DB  00001000b
```

6.4. keypad.asm

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutinas de control del teclado matricial 4x4.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Observaciones:
;; --------------
;;
;; Se utilizó un buffer 74HC244N como interfaz al microcontrolador,
;; conectado de la siguiente manera:
;;
;;          ------------------
;;         |     74HC244N     |
;;         |                  |
;; P2.0 --| 2A0 --|>-- 2Y0 |-- Fila 1
;; P2.1 --| 2A1 --|>-- 2Y1 |-- Fila 2
;; P2.2 --| 2A2 --|>-- 2Y2 |-- Fila 3
;; P2.3 --| 2A3 --|>-- 2Y3 |-- Fila 4
;; P2.4 --| 1Y3 --<|-- 1A3 |-- Columna 4
;; P2.5 --| 1Y2 --<|-- 1A2 |-- Columna 3
;; P2.6 --| 1Y1 --<|-- 1A1 |-- Columna 2
;; P2.7 --| 1Y0 --<|-- 1A0 |-- Columna 1
;;         |                  |
;;          ------------------
;;
;; Cuando un botón es presionado, la fila y la columna a la que
;; pertenece se pone en cero.
;;
;; Esquemático:
;; ------------
;;                              |---------------|---------------|
;;                              | Col1 | Col2 | Col3 | Col4 |
;;                              |------|------|------|------|
;;                              | P2.7 | P2.6 | P2.5 | P2.4 |
;;              |---------------|------|------|------|------|
;;              | Fil1 | P2.0   |  ESC |  9   |  8   |  <-  |
;;              | Fil2 | P2.1   |   F1 |  4   |  3   |  ->  |
;;              | Fil3 | P2.2   |  DEL |  5   |  2   |  1   |
;;              | Fil4 | P2.3   | ENTER|  0   |  7   |  6   |
;;              |------|--------|------|------|------|------|
;;
;; Disposición de los botones:
;; ---------------------------
;;
;;                     ------------------------------------
;;                    |      ------------------------      |
;;                    ||<-| |                        | |Esc||
;;                    |     |                        |     |
;;                    ||->| |                        |  |F1||
;;                    |      ------------------------      |
;;                    | |1|  |2|  |3|   |4|   |5|    |Del||
;;                    | |6|  |7|  |8|   |9|   |0|  |Enter||
;;                     ------------------------------------
;;
;;
;;
;; Requisitos de esta biblioteca:
;; ------------------------------
;;                              Existencia de una macro definida bajo el nombre
;;                              KEYPAD_MACRO_SELECT que garantice la
;;                              disponibilidad del teclado.
;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Definiciones:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Para no depender de las conexiones entre las columnas—filas y el buffer,
;; se definen las siguientes equivalencias para asignar el peso de cada columna
;; en función del bit del puerto:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
KEYPAD_F1       EQU     1d      ; Bit 0
KEYPAD_F2       EQU     2d      ; Bit 1
KEYPAD_F3       EQU     4d      ; Bit 2
KEYPAD_F4       EQU     8d      ; Bit 3
KEYPAD_C1       EQU     128d    ; Bit 7
KEYPAD_C2       EQU     64d     ; Bit 6
KEYPAD_C3       EQU     32d     ; Bit 5
KEYPAD_C4       EQU     16d     ; Bit 4

; Códigos que representan a las teclas no numéricas:
KEYPAD_RIGHT    EQU     10d
KEYPAD_LEFT     EQU     11d
KEYPAD_ENTER    EQU     12d
KEYPAD_HELP     EQU     13d
KEYPAD_DEL      EQU     14d
KEYPAD_ESC      EQU     15d
KEYPAD_NONE     EQU     16d
; Las teclas numéricas llevan su número como código.

KEYPAD_DATABUS  EQU     P2

; Aquí se indica la mitad de la cantidad de CM's deseados para debouncing.
KEYPAD_DEBOUNCE_DEEPNESS    EQU     255d
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Macros:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Simplifica la captura de la ejecución a la espera de una tecla en particular.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
KEYPAD_MACRO_WAIT_KEY   MACRO   codigo
    MOV     A, #codigo
    CALL    KEYPAD_SCAN_WAIT_KEY
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


DSEG


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Posición de memoria para comunicarle al exterior
; el código de la tecla presionada:
keypad_pressed_key:     DS      1
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
CSEG


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
KEYPAD_DEBOUNCE_WAIT:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina que realiza un delay por software para debouncing.
;; Está pensada para ser utilizada sólo por KEYPAD_SCAN.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Registros afectados:
;; ____________________
;;                      A
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    MOV     A, #KEYPAD_DEBOUNCE_DEEPNESS
    DJNZ    Acc, $  ; Esta operación lleva 2 ciclos de máquina.
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
KEYPAD_SCAN:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina que realiza un escaneo en KEYPAD_DATABUS
;; y devuelve en keypad_pressed_key el código de la tecla apretada.
;; Si ninguna está apretada, devuelve KEYPAD_NONE.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;              KEYPAD_DEBOUNCE_WAIT
;;              KEYPAD_SCAN_CORE
;;
;; Registros afectados:
;; ____________________
;;                      A, DPTR, keypad_pressed_key
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Primero se ejecuta la macro que garantiza la disponibilidad del teclado:
    KEYPAD_MACRO_SELECT

    ; Se obtiene una primera lectura:
    CALL    KEYPAD_SCAN_CORE

    ; Etapa de debouncing:

    ; Se guarda una copia de la primera lectura:
    PUSH    keypad_pressed_key
    ; Se espera un tiempo prudente:
    CALL    KEYPAD_DEBOUNCE_WAIT
    ; Se realiza una segunda lectura:
    CALL    KEYPAD_SCAN_CORE
    ; Se recupera la copia en el acumulador:
```

```
    POP     Acc
    ; Se las compara y si son iguales, se considera un éxito:
    CJNE    A, keypad_pressed_key, KEYPAD_SCAN_FAIL

   ; Fin de la etapa de debouncing.

    ; Para evitar múltiples lecturas idénticas, se atrapa la ejecución
    ; hasta que se verifique que la tecla ha sido soltada:
    PUSH    keypad_pressed_key      ; Se guarda una copia.
KEYPAD_SCAN_TRAP:
    ; Se realiza una lectura y se sigue iterando hasta
    ; que equivalga a ninguna tecla:
    CALL    KEYPAD_SCAN_CORE
    MOV     A, keypad_pressed_key
    CJNE    A, #KEYPAD_NONE, KEYPAD_SCAN_TRAP
    POP     keypad_pressed_key
    RET

KEYPAD_SCAN_FAIL:
    ; En caso de que la etapa de debouncing decida que fue una falsa alarma,
    ; se informa que no se ha detectado tecla alguna:
    MOV     keypad_pressed_key, #KEYPAD_NONE
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
KEYPAD_SCAN_CORE:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Corazón de la lectura de teclas.
;; Está pensada para ser llamada sólo por KEYPAD_SCAN.
;; Itera a través de la tabla de las máscaras que representan a cada tecla de
;; tal manera que cuando alguna coincide con la lectura de KEYPAD_DATABUS,
;; finaliza con el código de la tecla guardado en keypad_pressed_key.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Registros afectados:
;; ____________________
;;                      A, C, DPTR, keypad_pressed_key
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ; Se comienza probando con la primera máscara:
    MOV     DPTR, #KEYPAD_MASK_0

    ; Se inicializa a keypad_pressed_key en 255 para que luego del bucle
    ; quede con el valor correcto gracias a los sucesivos incrementos:
    MOV     keypad_pressed_key, #255d
KEYPAD_SCAN_LOOP:

    ; Se chequea para ver si se alcanzó el final de la tabla de máscaras.
    MOV     A, keypad_pressed_key
    CLR     C
    SUBB    A, #16d
    ; Si ya se recorrió toda la tabla sin éxito debe abandonar el bucle.
    JZ      KEYPAD_SCAN_CORE_END

    ; Se lee la máscara:
    CLR     A
    MOVC    A, @A+DPTR

    ; Se envía la máscara:
```

```
    MOV     KEYPAD_DATABUS, A

    ; Se le da tiempo al buffer para que actualice el valor:
    NOP

    ; Se apunta a la siguiente máscara:
    INC     DPTR

    ; Se incrementa keypad_pressed_key para que cuando se detecte
    ; finalmente la tecla apretada, quede con el valor de su código:
    INC     keypad_pressed_key

    ; Si no coincide, entonces itera nuevamente para probar con la próxima:
    CJNE    A, KEYPAD_DATABUS, KEYPAD_SCAN_LOOP
KEYPAD_SCAN_CORE_END:
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
KEYPAD_SCAN_WAIT:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Frena la ejecución hasta que se presione alguna tecla.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;              KEYPAD_SCAN
;;
;; Registros afectados:
;; ____________________
;;                      A, DPTR, keypad_pressed_key
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    CALL    KEYPAD_SCAN
    MOV     A, keypad_pressed_key
    CJNE    A, #KEYPAD_NONE, ($+3)
    JNC     KEYPAD_SCAN_WAIT
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
KEYPAD_SCAN_JMP:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina temeraria pensada para poder iterar un bucle arbitrario
;; hasta que se presione alguna tecla.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;              KEYPAD_SCAN
;;
;; Requisitos:
;; ___________
;;              La dirección de inicio del bucle en DPTR.
;;
```

```
;; Registros afectados:
;; ———————————————————
;;                      A, DPTR, keypad_pressed_key
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Antes de llamar a KEYPAD_SCAN, se guarda al DPTR pues contiene
    ; la dirección en memoria del inicio del bucle:
    PUSH    DPL
    PUSH    DPH

    ; Se fija si hay alguna tecla presionada:
    CALL    KEYPAD_SCAN

    ; Recupera a DPTR:
    POP     DPH
    POP     DPL

    MOV     A, keypad_pressed_key

    ; Si hay alguna tecla apretada, se finaliza:
    CJNE    A, #KEYPAD_NONE, KEYPAD_SCAN_JMP_RET

    ; Si no hay ninguna tecla apretada, se debe saltar a la dirección del
    ; bucle, pero como esta rutina ha sido llamada con CALL, es necesario
    ; antes retirar del stack a los dos bytes de la dirección de retorno,
    ; pues no se va a volver con RET:
    POP     Acc
    POP     Acc
    ; Finalmente, se realiza el salto a la dirección del bucle:
    CLR     A
    JMP     @A + DPTR

KEYPAD_SCAN_JMP_RET:
    ; Si había una tecla apretada, se retorna normalmente:
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
KEYPAD_SCAN_WAIT_KEY:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Frena la ejecución hasta que se presione una tecla específica.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Requisitos:
;; ———————————
;;              El código de la tecla deseada en el acumulador.
;;
;; Depende de:
;; ———————————
;;              KEYPAD_SCAN
;;
;; Registros afectados:
;; ———————————————————
;;                      A, DPTR, keypad_pressed_key
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    MOV     B, A
    CALL    KEYPAD_SCAN
```

```
    MOV     A, keypad_pressed_key
    CJNE    A, B, ($+5)
    SJMP    KEYPAD_SCAN_WAIT_KEY_END
    MOV     A, B
    SJMP    KEYPAD_SCAN_WAIT_KEY
KEYPAD_SCAN_WAIT_KEY_END:
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Tabla de máscaras de petición y respuesta.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Formato:
;; ________
;;          4 bits de petición / 4 bits de respuesta
;;          C1 C2 C3 C4        / F4 F3 F2 F1
;;
;; Ejemplo:
;; ________
;;          11101011 para detectar la tecla ubicada en Fila 3 y Columna 4.
;;
;; Para no depender la significación de cada elemento de la matriz,
;; se define en función de las asignaciones de los terminales.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
KEYPAD_MASK_0:       DB    LOW (255d - KEYPAD_F4 - KEYPAD_C2)
KEYPAD_MASK_1:       DB    LOW (255d - KEYPAD_F3 - KEYPAD_C4)
KEYPAD_MASK_2:       DB    LOW (255d - KEYPAD_F3 - KEYPAD_C3)
KEYPAD_MASK_3:       DB    LOW (255d - KEYPAD_F2 - KEYPAD_C3)
KEYPAD_MASK_4:       DB    LOW (255d - KEYPAD_F2 - KEYPAD_C2)
KEYPAD_MASK_5:       DB    LOW (255d - KEYPAD_F3 - KEYPAD_C2)
KEYPAD_MASK_6:       DB    LOW (255d - KEYPAD_F4 - KEYPAD_C4)
KEYPAD_MASK_7:       DB    LOW (255d - KEYPAD_F4 - KEYPAD_C3)
KEYPAD_MASK_8:       DB    LOW (255d - KEYPAD_F1 - KEYPAD_C3)
KEYPAD_MASK_9:       DB    LOW (255d - KEYPAD_F1 - KEYPAD_C2)
KEYPAD_MASK_RIGHT:   DB    LOW (255d - KEYPAD_F2 - KEYPAD_C4)
KEYPAD_MASK_LEFT:    DB    LOW (255d - KEYPAD_F1 - KEYPAD_C4)
KEYPAD_MASK_ENTER:   DB    LOW (255d - KEYPAD_F4 - KEYPAD_C1)
KEYPAD_MASK_F1:      DB    LOW (255d - KEYPAD_F2 - KEYPAD_C1)
KEYPAD_MASK_DEL:     DB    LOW (255d - KEYPAD_F3 - KEYPAD_C1)
KEYPAD_MASK_ESC:     DB    LOW (255d - KEYPAD_F1 - KEYPAD_C1)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Por ejemplo: la tecla que representa al CERO, está en Fila 4 y Columna 2.
;;
;; La resta entre 255 y dos valores, equivale a la negación de la suma
;; entre los valores.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

6.5. rom.asm

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutinas de control del dispositivo: Serial EEPROM — AT24C256 Atmel.
;; 256 kbits de memoria organizados en 512 páginas de 64 bytes cada una.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Rutinas públicas:
;; ____________________
;;                    ROM_WRITE_PAGE_FROM_CODE
;;                    ROM_READ_PAGE_TO_BUFFER
;;                    ROM_INC_ADDRESS_64
;;                    ROM_ERASE_PAGE
;;                    ROM_ERASE
;;                    ROM_READ_TO_BANK
;;                    ROM_WRITE_FROM_BANK
;;
;; Requisitos de esta biblioteca:
;; ______________________________
;;                               Existencia de una macro llamada
;;                               ROM_MACRO_SELECT que garantice la
;;                               disponibilidad del display.
;;
;;                               Existencia de dos macros llamadas
;;                               START_SENSIBLE_OP y END_SENSIBLE_OP
;;                               para indicarle al exterior el comienzo y fin
;;                               de operaciones de I/O sensibles.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Definiciones:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ROM_CLK             EQU     P2.0        ; Señal de Clock para la ROM.
ROM_SDA             EQU     P2.1        ; Datos a ROM (Full Duplex).
ROM_ADDRESS_MASK    EQU     0A0H        ; Device address para EEPROMs AT24Cxx.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


DSEG

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
rom_address_H:      DS      1           ; Posición de memoria ROM (byte alto)
rom_address_L:      DS      1           ; Posición de memoria ROM (byte bajo)

; Buffer para almacenar 64 bytes que van o vienen hacia o desde la ROM:
rom_page_buffer:    DS      64
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

CSEG


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ROM_BEGIN:
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Envia la condición de start a la ROM (Flanco descendente en SDA mientras CLK
;; se mantiene alto).
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Registros afectados:
;; ---------------------
;;                       C <- flag de error, 0 es OK.
;;                       ROM_CLK <- 0
;;                       ROM_SDA <- 0
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se ejecuta la macro que avisa al exterior sobre el comienzo
    ; de una operacion de I/O sensible:
    START_SENSIBLE_OP

    SETB    ROM_SDA
    SETB    ROM_CLK
    ; Si SDA o CLK estan en bajo entonces, error de bus.
    JNB     ROM_SDA, ROM_BEGIN_ERROR
    JNB     ROM_CLK, ROM_BEGIN_ERROR
    ; Se indica que no hay error.
    CLR     C
    CLR     ROM_SDA
    ; Tiempo de procesamiento.
    NOP
    NOP
    NOP
    NOP
    NOP
    CLR     ROM_CLK
    RET
ROM_BEGIN_ERROR:
    ; Se indica que hay error.
    SETB    C

    ; Se ejecuta la macro que avisa al exterior sobre el fin
    ; de una operacion de I/O sensible:
    END_SENSIBLE_OP

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ROM_END:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Envia la condición stop a la ROM (Flanco descendente en SDA mientras CLK se
;; mantiene alto).
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Registros afectados:
;; ---------------------
;;                       C <- flag de error, 0 es OK.
;;                       ROM_CLK <- 1
;;                       ROM_SDA <- 1
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se ejecuta la macro que avisa al exterior sobre el comienzo
    ; de una operacion de I/O sensible:
    START_SENSIBLE_OP

    ; SDA en bajo.
    CLR     ROM_SDA
    ; Tiempo de procesamiento.
    NOP
    NOP
    ; CLK en alto.
    SETB    ROM_CLK
    ; Tiempo de procesamiento.
    NOP
    NOP
    NOP
    NOP
    NOP
    ; SDA en alto.
    SETB    ROM_SDA

    ; Se ejecuta la macro que avisa al exterior sobre el fin
    ; de una operacion de I/O sensible:
    END_SENSIBLE_OP

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ROM_SEND_ACK:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Acepta los datos recibidos.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Requisitos:
;; ___________
;;              ROM_CLK en bajo.
;;
;; Registros afectados:
;; ____________________
;;                      ROM_CLK <- 0
;;                      ROM_SDA <- 0
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se ejecuta la macro que avisa al exterior sobre el comienzo
    ; de una operacion de I/O sensible:
    START_SENSIBLE_OP

    ; SDA en bajo -> bit de acknowledge.
    CLR     ROM_SDA
    ; Tiempo de procesamiento.
    NOP
    NOP
    ; CLK en alto.
    SETB    ROM_CLK
    ; Tiempo de procesamiento.
    NOP
    NOP
```

```
    NOP
    NOP
    ; CLK en bajo.
    CLR     ROM_CLK

    ; Se ejecuta la macro que avisa al exterior sobre el fin
    ; de una operacion de I/O sensible:
    END_SENSIBLE_OP

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ROM_SEND_NACK:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rechaza los datos recibidos.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Requisitos:
;; ___________
;;              ROM_CLK en bajo.
;;
;; Registros afectados:
;; ____________________
;;                       ROM_CLK <- 1
;;                       ROM_SDA <- 1
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se ejecuta la macro que avisa al exterior sobre el comienzo
    ; de una operacion de I/O sensible:
    START_SENSIBLE_OP

    ; SDA en alto -> bit de NO acknowledge.
    SETB    ROM_SDA
    ; Tiempo de procesamiento.
    NOP
    NOP
    ; CLK en alto.
    SETB    ROM_CLK
    ; Tiempo de procesamiento.
    NOP
    NOP
    NOP
    nop
    ; CLK en bajo.
    CLR     ROM_CLK

    ; Se ejecuta la macro que avisa al exterior sobre el fin
    ; de una operacion de I/O sensible:
    END_SENSIBLE_OP

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ROM_SHIFT_IN_BYTE:
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Recibe un byte en forma serial (MSB primero) via SDA y lo deja en A.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Requisitos:
;; ___________
;;              ROM_CLK en bajo.
;;
;; Registros afectados:
;; ____________________
;;                      A <- byte leído.
;;                      ROM_CLK <- 0
;;                      ROM_SDA
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se ejecuta la macro que avisa al exterior sobre el comienzo
    ; de una operacion de I/O sensible:
    START_SENSIBLE_OP

    ; Se inicializa SDA.
    SETB    ROM_SDA
    ; Se resguarda B.
    PUSH    B
    ;Se usa B como iterador — 8 bits son los que se envian.
    MOV     B, #8d
ROM_SHIFT_IN_BYTE_AUX:
    ; Tiempo de procesamiento.
    NOP
    NOP
    NOP
    ; CLK en alto.
    SETB    ROM_CLK
    ; Tiempo de procesamiento.
    NOP
    NOP
    ; Se copia SDA en C para formar el bit.
    MOV     C, ROM_SDA
    RLC     A
    ; CLK en bajo.
    CLR     ROM_CLK
    DJNZ    B, ROM_SHIFT_IN_BYTE_AUX

    ; Se restaura B
    POP     B

    ; Se ejecuta la macro que avisa al exterior sobre el fin
    ; de una operacion de I/O sensible:
    END_SENSIBLE_OP

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ROM_SHIFT_OUT_BYTE:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Recibe un byte en A, al que emite en forma serial, con el MSB primero,
;; por SDA.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Requisitos:
;; ___________
;;                      El byte a enviar en el acumulador.
;;                      ROM_SDA y ROM_CLK en estado bajo.
;;
;; Registros afectados:
;; ____________________
;;                      ROM_CLK <- 0
;;                      ROM_SDA
;;                      C <- flag de error. 0 es OK.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se ejecuta la macro que avisa al exterior sobre el comienzo
    ; de una operacion de I/O sensible:
    START_SENSIBLE_OP

    ; Se resguarda B.
    PUSH    B
    ; Se utiliza como iterador -> cantidad de bits.
    MOV     B, #8d

ROM_SHIFT_OUT_BYTE_AUX:
    ; Bit mas alto de A al carry.
    RLC     A
    MOV     ROM_SDA, C

    SETB    ROM_CLK
    ; Tiempo de procesamiento.
    NOP
    NOP
    NOP
    NOP
    CLR     ROM_CLK
    DJNZ    B, ROM_SHIFT_OUT_BYTE_AUX

   ; SDA en alto para el acknowledge.
    SETB    ROM_SDA
    ; Tiempo de procesamiento.
    NOP
    NOP

    ; CLK en alto.
    SETB    ROM_CLK
    ; Tiempo de procesamiento.
    NOP
    NOP
    NOP
    NOP
    MOV     C, ROM_SDA
    ; CLK en bajo.
    CLR     ROM_CLK

    ; Se restaura B.
    POP     B

    ; Se ejecuta la macro que avisa al exterior sobre el fin
    ; de una operacion de I/O sensible:
    END_SENSIBLE_OP

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ROM_WAIT_BUSY:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina bloqueante, espera que la ROM responda. Usa a C (carry) como flag de
;; error, 0 es OK.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;                      ROM_SHIFT_OUT_BYTE
;;
;; Registros afectados:
;; ____________________
;;                      C -> flag de error. 0 es OK.
;;                      A
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    CALL    ROM_BEGIN
    JC      ROM_WAIT_BUSY

    MOV     A,#ROM_ADDRESS_MASK
    CLR     ACC.0
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_WAIT_BUSY
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ROM_WRITE_PAGE_FROM_CODE:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Escribe en la ROM externa 64 bytes ubicados en la memoria de código del
;; microcontrolador.
;; Los datos se escriben en en la ROM a partir de la dirección indicada por
;; rom_address_H y rom_address_L (HI y LO), ocupando una página completa.
;; Recibe en DPTR la dirección del CSEG que contiene los datos a copiar.
;; En caso de error, aborta la escritura usando el carry como flag de error,
;; 0 es OK.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Depende de:
;; ___________
;;                      ROM_BEGIN
;;                      ROM_END
;;                      ROM_SHIFT_OUT_BYTE
;; Requisitos:
;; ___________
;;                      rom_address_L
;;                      rom_address_H
;;
;; Registros afectados:
;; ____________________
;;                      DPTR
;;                      A
;;                      C
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
    ; Se selecciona el dispositivo.
    ROM_MACRO_SELECT
    ; Se resguarda B.
    PUSH    B
    ; Se utiliza como iterador -> cantidad de bytes.
    MOV     B, #64
    ;; Condición de Start.
    CALL    ROM_BEGIN
    JC      ROM_WRITE_PAGE_FROM_CODE_END
    ;; Sale si hay error.

    ;; Se envia la Device Address (WRITE).
    MOV     A,#ROM_ADDRESS_MASK
    CLR     acc.0
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_WRITE_PAGE_FROM_CODE_ERROR
    ;; Sale si hay error.

    ;; Se envia la dirección de la página a escribir.
    MOV     A, rom_address_H
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_WRITE_PAGE_FROM_CODE_ERROR
    ;; Salimos si hay error.
    MOV     A, rom_address_L
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_WRITE_PAGE_FROM_CODE_ERROR
    ;; Sale si hay error.

ROM_WRITE_PAGE_FROM_CODE_LOOP:
    ; Se copia en A el byte de CSEG.
    CLR     A
    movc    A, @A+DPTR
    ;Se envia el byte.
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_WRITE_PAGE_FROM_CODE_ERROR
    ;; Salimos si hay error.

    INC     DPTR
    DJNZ    B, ROM_WRITE_PAGE_FROM_CODE_LOOP

    ; Se indica que no hay errores.
    CLR     C
    ; Condición de stop.
    CALL    ROM_END
    ; Se espera a que de desocupe la ROM.
    CALL    ROM_WAIT_BUSY
    JMP     ROM_WRITE_PAGE_FROM_CODE_END

ROM_WRITE_PAGE_FROM_CODE_ERROR:
    CALL    ROM_END
ROM_WRITE_PAGE_FROM_CODE_END:
    ; Se resatura B
    POP     B
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ROM_READ_PAGE_TO_BUFFER:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Lee una página (64 bytes) de la ROM externa y los copia a rom_page_buffer
;; definido en RAM interna.
;; La dirección inicial de la ROM externa viene via rom_address_H y
```

```
;; rom_address_L (HI y LO).
;; Recibe en DPTR la dirección del CSEG que contiene los datos a copiar.
;; En caso de error, usa el carry como flag de error, 0 es OK.
;; Hace un write falso para indicar la dirección en la ROM y de ahí lee en forma
;; secuencial.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;                      ROM_BEGIN
;;                      ROM_END
;;                      ROM_SEND_ACK
;;                      ROM_SEND_NACK
;;                      ROM_SHIFT_IN_BYTE
;;                      ROM_SHIFT_OUT_BYTE
;; Requisitos:
;; ___________
;;                      rom_address_L
;;                      rom_address_H
;;
;; Registros afectados:
;; ____________________
;;                      rom_page_buffer
;;                      A
;;                      C
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ; Se selecciona el dispositivo.
    ROM_MACRO_SELECT
    ; Condición de start para setear la posición desde donde se lee.
    CALL   ROM_BEGIN
    JC     ROM_READ_PAGE_TO_BUFFER_END
    ; Sale si hay error.

    ; Se envia la Device Address (WRITE).
    MOV    A,#ROM_ADDRESS_MASK
    CLR    ACC.0
    CALL   ROM_SHIFT_OUT_BYTE
    JC     ROM_READ_PAGE_TO_BUFFER_ERROR
    ; Sale si hay error.

    ; Se envia la dirección de la página a 'escribir'.
    MOV    A, rom_address_H
    CALL   ROM_SHIFT_OUT_BYTE
    JC     ROM_READ_PAGE_TO_BUFFER_ERROR
    ; Sale si hay error.
    MOV    A, rom_address_L
    CALL   ROM_SHIFT_OUT_BYTE
    JC     ROM_READ_PAGE_TO_BUFFER_ERROR
    ; Sale si hay error.

    ; Condición de start para leer.
    CALL   ROM_BEGIN
    JC     ROM_READ_PAGE_TO_BUFFER_ERROR
    ; Sale si hay error.

    ; Se enviamos la Device Address (READ).
    MOV    A, #ROM_ADDRESS_MASK
    SETB   acc.0
    CALL   ROM_SHIFT_OUT_BYTE
    JC     ROM_READ_PAGE_TO_BUFFER_ERROR
    ; Sale si hay error.

    ; Se resguarda B.
```

```
    PUSH    B
    ; Se resguarda R0.
    MOV     B,R0
    ; Se mueve a R0 la posición en CSEG del buffer.
    MOV     R0, #rom_page_buffer

ROM_READ_PAGE_TO_BUFFER_LOOP:
    ; Se lee en A.
    CALL    ROM_SHIFT_IN_BYTE
    ; Se envia el byte al buffer.
    MOV     @R0, A
    ; Se copia en A el puntero.
    MOV     A, R0
    SUBB    A,#rom_page_buffer

    ; Ahora A es cantidad de bytes escritos menos uno.
    CJNE    A, #64d, NOT_THE_LAST_BYTE
    ; Si es el último no se hace acknowledge.
    CALL    ROM_SEND_NACK
    JMP     ROM_READ_PAGE_TO_BUFFER_FINISH
NOT_THE_LAST_BYTE:
    ; Se anvia el acknowledge.
    CALL    ROM_SEND_ACK
    INC     R0
    JMP     ROM_READ_PAGE_TO_BUFFER_LOOP
ROM_READ_PAGE_TO_BUFFER_FINISH:
    MOV     R0,B                        ;Se restaura R0.
    POP     B                           ;Se restaura B.
    CLR     C                           ;No hubo errores.
ROM_READ_PAGE_TO_BUFFER_ERROR:
    CALL    ROM_END
ROM_READ_PAGE_TO_BUFFER_END:
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ROM_INC_ADDRESS_64:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para incrementar la dirección de memoria en 64 bytes
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    CLR     C
    MOV     A,rom_address_L
    ADD     A,#64d
    JNC     ROM_INC_ADDRESS_64_EXIT
    INC     rom_address_H
ROM_INC_ADDRESS_64_EXIT:
    MOV     rom_address_L, A
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ROM_ERASE_PAGE:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para borrar una página de 64 bytes.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
```

```
;; Depende de:
;; ___________
;;              ROM_BEGIN
;;              ROM_SHIFT_OUT_BYTE
;;              ROM_END
;;              ROM_WAIT_BUSY
;;
;; Requisitos:
;; ___________
;;              rom_address_L
;;              rom_address_H
;;
;; Registros alterados:
;; ____________________
;;                      A
;;                      C
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ROM_MACRO_SELECT
    CALL    ROM_BEGIN
    JC      ROM_ERASE_PAGE_END

    MOV     A, #ROM_ADDRESS_MASK
    CLR     Acc.0
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_ERASE_PAGE_ERROR

    MOV     A, rom_address_H
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_ERASE_PAGE_ERROR

    MOV     A, rom_address_L
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_ERASE_PAGE_ERROR

    PUSH    B
    MOV     B, #64
ROM_ERASE_PAGE_LOOP:
    CLR     A
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_ERASE_PAGE_ERROR
    DJNZ    B, ROM_ERASE_PAGE_LOOP

    CLR     C   ;No hay errores.
    CALL    ROM_END
    CALL    ROM_WAIT_BUSY
    JMP     ROM_ERASE_PAGE_END
ROM_ERASE_PAGE_ERROR:
    CALL    ROM_END
ROM_ERASE_PAGE_END:
    POP     B

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ROM_ERASE:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para borrar completamente el contenido de la memoria ROM.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;
;; Depende de:
;; ___________
;;               ROM_ERASE_PAGE
;;               ROM_INC_ADDRESS_64
;;
;; Requisitos:
;; ___________
;;
;; Registros alterados:
;; ____________________
;;                       B, A, C
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ROM_MACRO_SELECT
    MOV     rom_address_L, #00H
    MOV     rom_address_H, #00H
ROM_ERASE_LOOP:
    CALL    ROM_ERASE_PAGE
    CALL    ROM_INC_ADDRESS_64
    MOV     A, rom_address_H
    CJNE    A, #080H, ROM_ERASE_LOOP
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ROM_WRITE_FROM_BANK:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para grabar los 8 bytes del banco de registros seleccionado a ROM.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;               ROM_MACRO_SELECT
;;               ROM_BEGIN
;;               ROM_SHIFT_OUT_BYTE
;;               ROM_END
;;               ROM_WAIT_BUSY
;;
;; Requisitos:
;; ___________
;;
;; Registros alterados:
;; ____________________
;;                       A
;;                       C <- Informa si hubo errores. 1 = Fracaso, 0 = Éxito.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Primero se ejecuta la macro que garantiza que la rom esté disponible:
    ROM_MACRO_SELECT

    ; Se envía la señal de inicio:
    CALL    ROM_BEGIN
    ; En el carry se informa si se produjo un error:
    JC      ROM_WRITE_FROM_BANK_ERROR

    ; Se envía la Device Address (WRITE).
    MOV     A, #ROM_ADDRESS_MASK
    CLR     Acc.0
    CALL    ROM_SHIFT_OUT_BYTE
```

```
    ; En el carry se informa si se produjo un error:
    JC      ROM_WRITE_FROM_BANK_ERROR

    ; Se envía la dirección de la página a escribir:
    ; Primero la parte alta:
    MOV     A, rom_address_H
    CALL    ROM_SHIFT_OUT_BYTE
    ; En el carry se informa si se produjo un error:
    JC      ROM_WRITE_FROM_BANK_ERROR

    ; Luego la parte baja:
    MOV     A, rom_address_L
    CALL    ROM_SHIFT_OUT_BYTE
    ; En el carry se informa si se produjo un error:
    JC      ROM_WRITE_FROM_BANK_ERROR

    ; Inicio del envío de los registros del banco:

    MOV     A, R0
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_WRITE_FROM_BANK_ERROR

    MOV     A, R1
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_WRITE_FROM_BANK_ERROR

    MOV     A, R2
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_WRITE_FROM_BANK_ERROR

    MOV     A, R3
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_WRITE_FROM_BANK_ERROR

    MOV     A, R4
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_WRITE_FROM_BANK_ERROR

    MOV     A, R5
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_WRITE_FROM_BANK_ERROR

    MOV     A, R6
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_WRITE_FROM_BANK_ERROR

    MOV     A, R7
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_WRITE_FROM_BANK_ERROR

    ; Si no hubo errores, se limpia el Carry para informarlo:
    CALL    ROM_END
    CALL    ROM_WAIT_BUSY
    CLR     C
    RET

    ; Si hubo errores, se prende el Carry para informarlo:
ROM_WRITE_FROM_BANK_ERROR:
    CALL    ROM_END
    SETB    C
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ROM_READ_TO_BANK:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para leer 8 bytes ROM y guardarlos en
;; el banco de registros seleccionado.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;              ROM_MACRO_SELECT
;;              ROM_BEGIN
;;              ROM_SHIFT_OUT_BYTE
;;              ROM_SHIFT_IN_BYTE
;;              ROM_END
;;              ROM_WAIT_BUSY
;;
;; Requisitos:
;; ___________
;;              La parte alta de la dirección en rom en rom_address_H.
;;              La parte baja de la dirección en rom en rom_address_L.
;; Registros alterados:
;; ____________________
;;                      A
;;                      C <- Informa si hubo errores. 1 = Fracaso, 0 = Éxito.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Primero se ejecuta la macro que garantiza que la rom esté disponible:
    ROM_MACRO_SELECT

    ; Se envía la señal de inicio:
    CALL    ROM_BEGIN
    ; En el carry se informa si se produjo un error:
    JC      ROM_READ_TO_BANK_ERROR

    ; Se envía la Device Address (WRITE).
    MOV     A, #ROM_ADDRESS_MASK
    CLR     ACC.0
    CALL    ROM_SHIFT_OUT_BYTE
    ; En el carry se informa si se produjo un error:
    JC      ROM_READ_TO_BANK_ERROR

    ; Se envía la dirección de la página a escribir:
    ; Primero la parte alta:
    MOV     A, rom_address_H
    CALL    ROM_SHIFT_OUT_BYTE
    ; En el carry se informa si se produjo un error:
    JC      ROM_READ_TO_BANK_ERROR

    ; Luego la parte baja:
    MOV     A, rom_address_L
    CALL    ROM_SHIFT_OUT_BYTE
    ; En el carry se informa si se produjo un error:
    JC      ROM_READ_TO_BANK_ERROR

    ; Se envía la señal de inicio:
    CALL    ROM_BEGIN
    JC      ROM_READ_TO_BANK_ERROR

    ; Se envía la Device Address (READ).
    MOV     A, #ROM_ADDRESS_MASK
    SETB    acc.0
    CALL    ROM_SHIFT_OUT_BYTE
    JC      ROM_READ_TO_BANK_ERROR
```

```
    ; Inicio de la lectura de bytes:
    CALL    ROM_SHIFT_IN_BYTE
    MOV     R0,    A
    CALL    ROM_SEND_ACK

    CALL    ROM_SHIFT_IN_BYTE
    MOV     R1,    A
    CALL    ROM_SEND_ACK

    CALL    ROM_SHIFT_IN_BYTE
    MOV     R2,    A
    CALL    ROM_SEND_ACK

    CALL    ROM_SHIFT_IN_BYTE
    MOV     R3,    A
    CALL    ROM_SEND_ACK

    CALL    ROM_SHIFT_IN_BYTE
    MOV     R4,    A
    CALL    ROM_SEND_ACK

    CALL    ROM_SHIFT_IN_BYTE
    MOV     R5,    A
    CALL    ROM_SEND_ACK

    CALL    ROM_SHIFT_IN_BYTE
    MOV     R6,    A
    CALL    ROM_SEND_ACK

    CALL    ROM_SHIFT_IN_BYTE
    MOV     R7,    A
    CALL    ROM_SEND_NACK           ;Al último no hacemos ACK.

    ; Si no hubo errores, se limpia el Carry para informarlo:
    CLR     C

    ; Si hubo errores, se deja al Carry prendido para informarlo:
ROM_READ_TO_BANK_ERROR:
    CALL    ROM_END
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

6.6. adc.asm

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutinas de control del dispositivo: A/D Converter — ADC0834ACN
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Rutinas públicas:
;; ——————————————————
;;                    ADC_CONVERSION
;;
;; Macros públicas:
;; ——————————————————
;;                    ADC_MACRO_READ_CHANNEL_0
;;                    ADC_MACRO_READ_CHANNEL_1
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Definiciones:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ADC_CLK             EQU     P2.0    ; Señal de Clock ADC.
ADC_DATA_INPUT      EQU     P2.1    ; Datos: Micro -> ADC.
ADC_DATA_OUTPUT     EQU     P2.2    ; Datos: ADC -> Micro.
ADC_SARS            EQU     P2.3    ; Señal en proceso.

;;                          123 -> Bits que determinan el canal.
ADC_CHANNEL0        EQU     10000000b
ADC_CHANNEL1        EQU     11000000b
ADC_CHANNEL2        EQU     10100000b
ADC_CHANNEL3        EQU     11100000b
         ; +-
ADC_CHANNEL01       EQU     00000000b
ADC_CHANNEL10       EQU     01000000b
ADC_CHANNEL23       EQU     00100000b
ADC_CHANNEL32       EQU     01100000b

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Macros:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Simplifica la lectura del canal 0. Devuelve en el acumulador el byte leído.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ADC_MACRO_READ_CHANNEL_0    MACRO
    MOV     A, #ADC_CHANNEL0
    CALL    ADC_CONVERSION
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Simplifica la lectura del canal 1. Devuelve en el acumulador el byte leído.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ADC_MACRO_READ_CHANNEL_1    MACRO
    MOV     A, #ADC_CHANNEL1
    CALL    ADC_CONVERSION
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Simplifica la lectura del canal 2. Devuelve en el acumulador el byte leído.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ADC_MACRO_READ_CHANNEL_2    MACRO
    MOV     A, #ADC_CHANNEL2
    CALL    ADC_CONVERSION
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Simplifica la lectura del canal 3. Devuelve en el acumulador el byte leído.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ADC_MACRO_READ_CHANNEL_3    MACRO
    MOV     A, #ADC_CHANNEL3
    CALL    ADC_CONVERSION
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


CSEG


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutinas:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ADC_CONVERSION:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;              ADC_MACRO_SELECT
;;              ADC_PULSE
;;
;; Requisitos:
;; ____________
;;              El canal deseado en el acumulador.
;;              (Ver Definiciones de ADC_CHANNEL0 a 4.)
;;
;; Registros afectados:
;; ____________________
;;                      A <- byte leído. 0 equivale a 0 Volts, 255 a 5 Volts.
;;                      C <- flag de error, 0 es OK.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Primero se ejecuta la macro que garantiza que el ADC está disponible:
    ADC_MACRO_SELECT
    CLR     ADC_CLK
    ; Se envía el bit de Start:
    SETB    ADC_DATA_INPUT
    CALL    ADC_PULSE

    ; Se usa B como iterador para mandar los tres bits que determinan
    ; el canal multiplexado:
    PUSH    B
    MOV     B, #3d
```

```
    ; Se establece el canal:

ADC_CONVERSION_SHIFT_IN:
    ; Se envían los tres bits del canal al carry para establecer
    ; la comunicación serial:
    RLC     A
    MOV     ADC_DATA_INPUT, C
    CALL    ADC_PULSE
    JB      ADC_SARS, ($+3)
    SETB    C
    DJNZ    B, ADC_CONVERSION_SHIFT_IN


    ; Acontinuación se guarda la conversión en A.

    ; Se usa nuevamente B como iterador. Esta vez B=8 ya que
    ; son 8 bits los que se deben almacenar.
    MOV     B, #8d

ADC_CONVERSION_SHIFT_OUT:
    CALL    ADC_PULSE
    MOV     C,ADC_DATA_OUTPUT
    ; Se guardan uno a uno los bits en A:
    RLC     A
    ; Tiempo para el ADC y el BUFFER:
    NOP
    NOP
    NOP
    NOP
    JNB     ADC_SARS, ($+3)
    SETB    C

    DJNZ    B, ADC_CONVERSION_SHIFT_OUT

    POP     B
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ADC_PULSE:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina que envía efectivamente los datos al microcontrolador.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    SETB    ADC_CLK
    NOP     ; Delay debido al BUFFER.
    CLR     ADC_CLK
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

6.7. decoder.asm

```
DECA    EQU P0.1        ; Entrada A del decodificador.
DECB    EQU P0.0        ; Entrada B del decodificador.
DECC    EQU P1.7        ; Entrada C del decodificador.

DECODER_DISPLAY     EQU     10000000b
DECODER_KEYPAD      EQU     01000000b
DECODER_ROM         EQU     11000000b
DECODER_ADC         EQU     00000000b

DSEG

; Si ninguna interrupción va a cambiar de dispositivo, se puede
; evitar usar esta variable.
decoder_selected_device:    DS  1

CSEG

DECODER_SELECT_DEVICE:
      START_SENSIBLE_OP
      PUSH  Acc
      CLR   C
      MOV   A, decoder_selected_device
      RLC   A
      MOV   DECA, C
      RLC   A
      MOV   DECB, C
      RLC   A
      MOV   DECC, C
      POP   Acc

      NOP
      NOP
      NOP
      NOP
      END_SENSIBLE_OP
      RET
```

6.8. timers.asm

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutinas de timers y manejo de tiempos.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Registro TMOD:
;; ──────────────
;; 0: Timer 0 M0 bit
;; 1: Timer 0 M1 bit
;; 2: Timer 0 Clock source select bit (External(1)/On-chip-clock(0)).
;; 3: Timer 0 Gate bit.
;; 4: Timer 1 M0 bit
;; 5: Timer 1 M1 bit
;; 6: Timer 1 Clock source select bit (External(1)/On-chip-clock(0)).
;; 7: Timer 1 Gate bit.
;;
;; Timer modes:
;; ────────────
;; M1   M0  MODE    Description
;;  0    0     0    13-bit timer mode.
;;  0    1     1    16-bit timer mode.
;;  1    0     2    8-bit auto-reload mode.
;;  1    1     3    Split timer mode.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Definiciones:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Base de tiempo del Timer 0 en milisegundos:
TIMER0_BASE_MS      EQU     10d
; Valor inicial de la cuenta regresiva del Timer 0.
TIMER0_COUNT_START  EQU     100d

; Base de tiempo del Timer 1 en milisegundos:
TIMER1_BASE_MS      EQU     1d

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


DSEG


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
timer0_count_down:  DS      1   ; Valor de la cuenta regresiva del Timer 0.

;; Segundero de 2 bytes. 2^16 segundos = 18:12:16 (horas:minutos:segundos).
timers_seconds_L:   DS      1   ; Byte bajo del segundero.
timers_seconds_H:   DS      1   ; Byte alto del segundero.

; \()/
timers_msecs:       DS      1   ; milisegundero.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


CSEG
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TIMERS_INIT:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina de inicialización de timers.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Está pensada para ser llamada al momento de inicialización del sistema.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    CLR     TR0 ; Por seguridad, se deshabilitan los 2 timers y sus
    CLR     TR1
    CLR     TF0 ; interrupciones asociadas.
    CLR     TF1

    ; Se carga el valor inicial de la cuenta regresiva del Timer 0:
    MOV     timer0_count_down, #TIMER0_COUNT_START

    ; Se inicializa el segundero:
    MOV     timers_seconds_L, #0d
    MOV     timers_seconds_H, #0d

    MOV     timers_msecs, #0d

    MOV     TMOD, #00010001b ; Ambos en modo 16-bit.

    ; Se carga el iterador del Timer 0:
    MOV     TL0, #LOW (-(1000d * TIMER0_BASE_MS))
    MOV     TH0, #HIGH(-(1000d * TIMER0_BASE_MS))

    ; Se carga el iterador del Timer 1:
    MOV     TL1, #LOW (-(1000d * TIMER1_BASE_MS))
    MOV     TH1, #HIGH(-(1000d * TIMER1_BASE_MS))

    ; ON/OFF de los Timers y sus interrupciones asociadas:
    CLR     ET0
    CLR     ET1
    CLR     TR0
    CLR     TR1

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TIMERS_T0ISR:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina de servicio de la interrupción del overflow del Timer 0.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Se dispara en intervalos de duración dictados por TIMER0_BASE_MS.
;;
;; En cada llamada, decrementa la cuenta regresiva (timer0_count_down)
;; para saber cuando incrementar a las variables que almacenan la cantidad
;; de segundos pasados desde que el timer 0 fue puesto funcionar.
;;
;; Registros afectados:
;; ---------------------
;;                  timer0_count_down
;;                  timers_seconds_L
;;                  timers_seconds_H
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    CLR     TR0 ; Se detiene a si mismo antes de la recarga.

    ; Recarga del timer:
    ; Se suma la cantidad de ciclos de máquina de la rutina
    ; para aumentar la precisión.
    MOV     TL0, #LOW (-(1000d * TIMER0_BASE_MS) + 8d)
    MOV     TH0, #HIGH(-(1000d * TIMER0_BASE_MS) + 8d)

    ; Se decrementa la cuenta regresiva del segundero:
    DJNZ    timer0_count_down, TIMERS_T0ISR_END
    ; Si la cuenta llegó a cero, hay que incrementar un segundo.

    PUSH    PSW     ; Se salva porque se necesita usar el Carry.

    ; Se utiliza XCH para evitar usar PUSH/POP y para ahorrar un MOV:
    XCH     A, timers_seconds_L
    ADD     A, #1d  ; No se usa INC pues no informa sobre overflows.

    ; Si C = 1 hubo overflow 255 -> 0, entonces
    ; hay que incrementar también al byte alto:
    JNC     TIMERS_T0ISR_NO_OV
    INC     timers_seconds_H
TIMERS_T0ISR_NO_OV:
    XCH     A, timers_seconds_L ; Se restauran las variables.
    POP     PSW

    ; Se reinicia la cuenta regresiva:
    MOV     timer0_count_down, #TIMER0_COUNT_START

TIMERS_T0ISR_END:
    SETB    TR0 ; Se enciende a si mismo nuevamente.
    RETI
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TIMERS_T1ISR:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina de servicio de la interrupción del overflow del Timer 1.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Se dispara en intervalos de duración dictados por TIMER1_BASE_MS.
;;
;; Registros afectados:
;; --------------------
;;                      timers_msec
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se detiene a si mismo antes de la recarga:
    CLR     TR1

    ; Recarga del timer:
    ; Se suma la cantidad de ciclos de máquina de la rutina
    ; para aumentar la precisión.
    MOV     TL1, #LOW (-(1000d * TIMER1_BASE_MS) + 7d)
    MOV     TH1, #HIGH(-(1000d * TIMER1_BASE_MS) + 7d)

    INC     timers_msecs

TIMERS_T1ISR_END:
```

```
    ; Se enciende a si mismo nuevamente:
    SETB    TR1
    RETI
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TIMERS_START_SECONDS_CRON:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina de iniciación del cronómetro de segundos.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Al ser llamada, reinicia el segundero de dos bytes, habilita al Timer 0 y
;; su interrupción asociada.
;;
;; Registros posiblemente afectados:
;; ---------------------------------
;       timer0_count_down   <- #TIMER0_COUNT_START
;;      timers_seconds_L    <- 0
;;      timers_seconds_H    <- 0
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ; Se deshabilita temporalmente al Timer 0 y su interrupción asociada:
    CLR     ET0
    CLR     TR0

    ; Reiniciación de valores:
    MOV     timer0_count_down, #TIMER0_COUNT_START
    MOV     timers_seconds_L, #0d
    MOV     timers_seconds_H, #0d
    MOV     TL0, #LOW (-(1000d * TIMER0_BASE_MS))
    MOV     TH0, #HIGH(-(1000d * TIMER0_BASE_MS))

    ; Se habilita al Timer 0 y su interrupción asociada:
    SETB    ET0
    SETB    TR0
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TIMERS_STOP_SECONDS_CRON:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina de detención del cronómetro de segundos.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Al ser llamada, detiene al Timer 0 y su interrupción asociada.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ; Se deshabilita temporalmente al Timer 0 y su interrupción asociada:
    CLR     ET0
    CLR     TR0
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TIMERS_START_MSECS_CRON:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina de iniciación del cronómetro de milisegundos.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Al ser llamada, reinicia el milisegundero, habilita al Timer 1 y
;; su interrupción asociada.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ; Se deshabilita temporalmente al Timer 1 y su interrupción asociada:
    CLR     ET1
    CLR     TR1

    ; Reiniciación de valores:
    MOV     timers_msecs, #0d
    MOV     TL1, #LOW (-(1000d * TIMER1_BASE_MS))
    MOV     TH1, #HIGH(-(1000d * TIMER1_BASE_MS))

    ; Se habilita al Timer 1 y su interrupción asociada:
    SETB    ET1
    SETB    TR1
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TIMERS_STOP_MSECS_CRON:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina de detención del cronómetro de milisegundos.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Al ser llamada, detiene al Timer 1 y su interrupción asociada.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ; Se deshabilita temporalmente al Timer 1 y su interrupción asociada:
    CLR     ET1
    CLR     TR1
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

6.9. i18n-sp.asm

```
i18n_Load_sample:
DB  'Cargue la muestra'
DB  0

i18n_Enter_to_continue:
DB  'ENTER para continuar'
DB  0

i18n_Enter_to_accept:
DB  'ENTER para aceptar'
DB  0

i18n_Enter_to_edit:
DB  'Enter para editar'
DB  0

i18n_Esc_to_cancel:
DB  'ESC para cancelar'
DB  0

i18n_Enter_for_edit:
DB  'ENTER: editar'
DB  0

i18n_F1_for_excute:
DB  'F1:    realizar'
DB  0

i18n_DEL_for_delete:
DB  'DEL:   eliminar'
DB  0

i18n_Reach_preassure:
DB  'Alcance la presion'
DB  0

i18n_Settling_sample:
DB  'Muestra en reposo'
DB  0

i18n_Charging_capacitors:
DB  'Cargando capacitores'
DB  0

i18n_Spark:
DB  127d,'Spark!'
DB  0

i18n_Take_sample:
DB  'Retire la muestra'
DB  0

i18n_Loading_data:
DB  'Cargando datos...'
DB  0

i18n_Saving_data:
DB  'Guardando datos...'
DB  0

i18n_Title_Confirm_data:
DB  '-=Confirmar datos=-'
```

```
DB  0

i18n_Erasing_ROM:
DB  'Borrando ROM'
DB  0

i18n_ROM_erased:
DB  'ROM borrada'
DB  0

i18n_Title_Test_nnn_mmm:
DB  '-=ENSAYO nnn/mmm=-'
DB  0

i18n_All_tests_completed:
DB  'Ensayos completados'
DB  0

i18n_Out_of_space:
DB  'No hay mas espacio.'
DB  0

i18n_Voltage:
DB  'Tension'
DB  0

i18n_Preassure:
DB  'Presion'
DB  0

i18n_Settling_time:
DB  'T. reposo'
DB  0

i18n_Discharge_time:
DB  'T. descarga'
DB  0

i18n_Delete_message_A:
DB  128d,'Intercambiar por el'
DB  0
i18n_Delete_message_B:
DB  'ultimo de la lista?'
DB  0
i18n_Delete_message_C:
DB  '(y reducir la lista)'
DB  0

i18n_Updating_data:
DB  'Actualizando datos'
DB  0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Texto de los menús.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Formato:
;; --------
;; MENUS_{NOMBRE DEL MENU}_LABELS:
;; DB  'TITULO'  , 0d
;; DB  'OPCION 0', 0d
;; DB  'OPCION 1', 0d
;; DB  'OPCION 2', 0d
;; DB  'OPCION 3', 0d
;; DB  'OPCION 4', 0d
;; DB  'OPCION 5', 0d
;; DB  MENUS_EOM
```

```
;;
;; El título no puede superar los 20 caracteres.
;; Las opciones no pueden superar los 18 caracteres, pues se reservan los 2
;; primeros para mostrar un indicador de selección.
;; Recordar que la pantalla tiene 8 páginas. De esta forma, se utiliza la
;; primera para el título, desde la segunda hasta la séptima para mostrar
;; un máximo de 6 opciones, y la última se reserva para un eventual mensaje
;; de ayuda.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

i18n_Main_menu_labels:
DB  '-=MENU PRINCIPAL=-', 0d
DB  'Anillo de ensayos', 0d
DB  'Agregar ensayo', 0d
DB  'Ejecutar todos', 0d
DB  'Extras', 0d
DB  WIDGETS_MENU_EOM

i18n_Edit_menu_labels:
DB  '-=EDITAR ENSAYO=-', 0d
DB  'Tension de carga',0d
DB  'Presion',0d
DB  'Tiempo de reposo',0d
DB  'Tiempo de descarga',0d
DB  'Grabar ensayo',0d
DB  WIDGETS_MENU_EOM

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

6.10. widgets.asm

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutinas para crear accesorios (widgets) en la pantalla, utilizando las
;; librerías de control específicas de display y teclado.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Rutinas y macros públicas:
;; ──────────────────────────
;;                          WIDGETS_MENU_INTERACT
;;                          WIDGETS_PUT_INT
;;                          WIDGETS_PROMPT
;;                          WIDGETS_PROGRESS_BAR
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Definiciones:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; El siguiente valor se utiliza para simbolizar el fin de un menú.
WIDGETS_MENU_EOM   EQU     1d
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Macros:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Simplifica la estructura necesaria para mostrar la representación decimal
;; de un número en pantalla.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
WIDGETS_MACRO_PUT_INT   MACRO   numero_L, numero_H, pagina, columna
    MOV     R2, numero_L
    MOV     R1, numero_H
    MOV     display_page, #pagina
    MOV     display_column, #columna
    CALL    WIDGETS_PUT_INT
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Simplifica la estrucura necesaria para situar un prompt en pantalla y
;; obtener los 2 bytes del número ingresado.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
WIDGETS_MACRO_PROMPT    MACRO   destino_L, destino_H, pagina, columna
    MOV     display_page, #pagina
    MOV     display_column, #columna
    CALL    WIDGETS_PROMPT
    MOV     destino_L, widgets_buffer
    MOV     destino_H, widgets_buffer + 1d
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Simplifica la estructura necesaria para construir una barra de progreso
;; a partir de un valor objetivo y un valor actual.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
WIDGETS_MACRO_PROGRESS_BAR_DIFF  MACRO   actual, objetivo, pagina
    MOV     A, #255d
    MOV     B, objetivo
    DIV     AB
    MOV     B, actual
    MUL     AB

    MOV     display_page, #pagina
    MOV     display_data, A
    CALL    WIDGETS_PROGRESS_BAR
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


DSEG


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Variable para saber qué opción del menú está seleccionada:
widgets_menu_option:    DS      1

; Variable para saber cuantas opciones tiene el menú mostrado:
widgets_menu_length:    DS      1

; Variable para almacenar los 6 bytes necesarios en las conversiones de dígitos
; a bytes y viceversa:
widgets_buffer:         DS      6
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


CSEG


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
WIDGETS_SHOW_MENU:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para mostrar un menú en pantalla.
;; Está pensada para ser utilizada por WIDGETS_MENU_INTERACT.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;              DISPLAY_PUT_STRING_STATIC
;;              DISPLAY_PUT_CHAR
;;
;; Requisitos:
;; ___________
;;              La dirección del texto del menú en DPTR.
;;
;; Registros alterados:
;; ____________________
;;                      display_page
;;                      display_column
;;                      widgets_menu_length
;;                      A
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
    ; Primero se muestra el título (primera cadena).
    MOV     display_page, #0d
    MOV     display_column, #0d
    CALL    DISPLAY_PUT_STRING_STATIC

    MOV     widgets_menu_length, #0d
    INC     DPTR
    INC     display_page

WIDGETS_SHOW_MENU_LOOP:
    ; Se muestran todas las opciones.
    MOV     display_column, #2d
    CALL    DISPLAY_PUT_STRING_STATIC
    INC     DPTR
    INC     display_page
    INC     widgets_menu_length
    CLR     A
    MOVC    A,@A+DPTR
    ; Hasta que se alzanza el fin del menú.
    CJNE    A, #WIDGETS_MENU_EOM, WIDGETS_SHOW_MENU_LOOP

    ; Finalmente se muestra el indicador de selección.
    MOV     display_page, widgets_menu_option
    INC     display_page
    MOV     display_column, #0d
    MOV     DPTR, #CHR_RIGHT_ARROW
    CALL    DISPLAY_PUT_CHAR

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
WIDGETS_MENU_INTERACT:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para mostrar un menú e interactuar con el teclado.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;              WIDGETS_SHOW_MENU
;;              DISPLAY_CLEAR
;;              KEYPAD_SCAN_WAIT
;;
;; Requisitos:
;; ___________
;;              La dirección del texto del menú en DPTR.
;;
;; Registros alterados:
;; ____________________
;;                      widgets_menu_option
;;                      A
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; El indicador de selección comienza siempre en la primera opción:
    MOV     widgets_menu_option, #0d

WIDGETS_MENU_INTERACT_REDRAW:
    CALL    DISPLAY_CLEAR

    PUSH    DPL     ; Se salva DPTR pues WIDGETS_SHOW_MENU y KEYPAD_SCAN_WAIT
```

```
    PUSH    DPH     ; lo alteran.

    ; Se llama a la rutina que muestra las opciones en pantalla con
    ; el indicador de selección al lado del elemento del menú seleccionado:
    CALL    WIDGETS_SHOW_MENU
    CALL    KEYPAD_SCAN_WAIT

    POP     DPH     ; Se recupera a DPTR.
    POP     DPL

    ; La estructura siguiente es equivalente a el típico SWITCH
    ; de los lenguajes de nivel más alto.

    ; Switch (keypad_pressed_key):
    MOV     A, keypad_pressed_key

    ; Case KEYPAD_RIGHT:
    CJNE    A, #KEYPAD_RIGHT, ($+3+2)
    SJMP    WIDGETS_MENU_INTERACT_RIGHT
    ; Case KEYPAD_LEFT:
    CJNE    A, #KEYPAD_LEFT, ($+3+2)
    SJMP    WIDGETS_MENU_INTERACT_LEFT
    ; Case KEYPAD_ENTER:
    CJNE    A, #KEYPAD_ENTER, ($+3+2)
    SJMP    WIDGETS_MENU_INTERACT_ENTER
    ; Case KEYPAD_ESC:
    CJNE    A, #KEYPAD_ESC, ($+3+2)
    SJMP    WIDGETS_MENU_INTERACT_ESC
    ; Default:
    SJMP    WIDGETS_MENU_INTERACT_REDRAW

    ; A continuación, se definen las acciones de las teclas especiales:
WIDGETS_MENU_INTERACT_RIGHT:
    ; Si se presiona esta tecla, hay que seleccionar a la opción de abajo.
    INC     widgets_menu_option
    MOV     A, widgets_menu_option

    ; Si no se alcanzó a la última opción, se vuelve a dibujar el menú
    ; pero con la selección incrementada:
    CJNE    A, widgets_menu_length, WIDGETS_MENU_INTERACT_REDRAW

    ; Si se alcanzó el final, se vuelve hacia arriba (topología circular).
    MOV     widgets_menu_option, #0d
    SJMP    WIDGETS_MENU_INTERACT_REDRAW
WIDGETS_MENU_INTERACT_LEFT:
    ; Si se presiona esta tecla, hay que seleccionar a la opción de abajo.
    DEC     widgets_menu_option
    MOV     A, widgets_menu_option

    ; Si no se alcanzó a la primera opción, se vuelve a dibujar el menú
    ; pero con la selección decrementada:
    CJNE    A, #255d, WIDGETS_MENU_INTERACT_REDRAW
    ; (Se compara contra 255d porque si se decrementa a 0 hay overflow.)

    ; Si se alcanzó el principio, reaparece por abajo (topología circular).
    MOV     widgets_menu_option, widgets_menu_length
    DEC     widgets_menu_option
    SJMP    WIDGETS_MENU_INTERACT_REDRAW

    ; Actualmente no hay necesidad de diferenciar si se abandona el menú
    ; con ENTER o ESC pues la rutina exterior se encarga de asignar a cada
    ; tecla una acción distinta.
WIDGETS_MENU_INTERACT_ENTER:
WIDGETS_MENU_INTERACT_ESC:
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
WIDGETS_PUT_INT:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para mostrar en pantalla un número entero
;; representable por 1 o 2 bytes. Si es de 1 byte, muestra 3 dígitos;
;; si es de 2, muestra 5.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ----------
;;             HEX2ASCII
;;             DISPLAY_PUT_STRING_DYNAMIC
;;
;; Requisitos:
;; ----------
;;             Parte baja del número en R2.
;;             Parte alta del número en R1.
;;             display_column entre 0 y 19
;;             display_page entre 0 y 7
;;
;; Registros alterados:
;; -------------------
;;                     A, B, R0
;;                     widgets_buffer
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se salva la parte alta para luego decidir si mostrar 3 o 5 dígitos:
    MOV     A, R1
    PUSH    Acc

    ; R2 <-- parte baja.
    ; R1 <-- parte alta.
    CALL    HEX2ASCII

    ; Decenas de millares:
    MOV     widgets_buffer + 0d, R3
    ; Millares:
    MOV     widgets_buffer + 1d, R4
    ; Centenas:
    MOV     widgets_buffer + 2d, R5
    ; Decenas:
    MOV     widgets_buffer + 3d, R6
    ; Unidades
    MOV     widgets_buffer + 4d, R7

    ; Se apunta R0 al comienzo de la cadena y se la muestra en pantalla:

    ; Se recupera la parte alta:
    POP     Acc

    JNZ     WIDGETS_PUT_INT16
    ; Si es nula, sólo se deben mostrar los últimos 3:
    MOV     R0, #(widgets_buffer + 2d)
    SJMP    WIDGETS_PUT_INT_SHOW

    ; Si no es nula, se deben mostrar los 5 dígitos:
WIDGETS_PUT_INT16:
    MOV     R0, #(widgets_buffer + 0d)

WIDGETS_PUT_INT_SHOW:
```

```
    ; Se marca el fin de línea en el sexto byte del buffer:
    MOV     widgets_buffer + 5d, #0d

    CALL    DISPLAY_PUT_STRING_DYNAMIC


    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
WIDGETS_PROMPT:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para obtener un número desde el teclado.
;; Muestra un cursor y espera a que el usuario ingrese
;; un número y presione ENTER.
;; Permite borrar caracter por caracter.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;              DISPLAY_PUT_CHAR
;;              KEYPAD_SCAN_WAIT
;;              BCD2HEX
;;              DISPLAY_PUT_ASCII
;;
;; Requisitos:
;; ___________
;;              display_column entre 0 y 19
;;              display_page entre 0 y 7
;;
;; Registros alterados:
;; ____________________
;;                      A, B, c
;;                      R1
;;                      widgets_buffer + 0 <- parte baja del número ingresado.
;;                      widgets_buffer + 1 <- parte alta del número ingresado.
;;                      widgets_buffer + 2 <- 0
;;                      widgets_buffer + 3 <- 0
;;                      widgets_buffer + 4 <- 0
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Limpieza del buffer:
    MOV     R1, #widgets_buffer + 5
    MOV     @R1, #0d
    DEC     R1
    MOV     @R1, #0d
    DEC     R1
    MOV     @R1, #0d
    DEC     R1
    MOV     @R1, #0d
    DEC     R1
    MOV     @R1, #0d
    DEC     R1
    MOV     @R1, #0d

    ; Se muestra un símbolo para guiar al usuario:
    MOV     DPTR, #CHR_062
    CALL    DISPLAY_PUT_CHAR
    INC     display_column

WIDGETS_PROMPT_CURSOR:
```

```
    ; Se muestra el cursor en la columna donde aparecerá el próximo dígito:
    MOV     DPTR, #CHR_CURSOR
    CALL    DISPLAY_PUT_CHAR

WIDGETS_PROMPT_WAIT:
    ; Se espera la presión de alguna tecla:
    CALL    KEYPAD_SCAN_WAIT

    ; Chequeo de teclas especiales:

    ; La estructura siguiente es equivalente a el típico SWITCH
    ; de los lenguajes de nivel más alto:

    ; Switch (keypad_pressed_key):
    MOV     A, keypad_pressed_key

    ; Case KEYPAD_ESC:
    CJNE    A, #KEYPAD_ESC, ($+5)
    SJMP    WIDGETS_PROMPT_KEY_ESC
    ; Case KEYPAD_ENTER:
    CJNE    A, #KEYPAD_ENTER, ($+5)
    SJMP    WIDGETS_PROMPT_KEY_ENTER
    ; Case KEYPAD_DEL:
    CJNE    A, #KEYPAD_DEL, ($+5)
    SJMP    WIDGETS_PROMPT_KEY_DEL
    ; Default:
    SJMP    WIDGETS_PROMPT_KEY_NUM

WIDGETS_PROMPT_KEY_DEL:
    ; Chequea que no se encuentre en la primera posición:

    MOV     A, R1
    CJNE    R1, #widgets_buffer, ($+3+2)
    SJMP    WIDGETS_PROMPT_CURSOR

    ; Borra el cursor en la posición actual:
    MOV     DPTR, #CHR_032
    CALL    DISPLAY_PUT_CHAR
    ; Borra el último caracter:
    DEC     display_column
    MOV     DPTR, #CHR_032
    CALL    DISPLAY_PUT_CHAR
    ; Corrije el resultado:
    MOV     @R1, #0d
    DEC     R1

    JMP     WIDGETS_PROMPT_CURSOR

WIDGETS_PROMPT_KEY_ESC:

    ; Limpieza del buffer:
    MOV     R1, #widgets_buffer + 5
    MOV     @R1, #0d
    DEC     R1
    MOV     @R1, #0d
    DEC     R1
    MOV     @R1, #0d
    DEC     R1
    MOV     @R1, #0d
    DEC     R1
    MOV     @R1, #0d
    DEC     R1
    MOV     @R1, #0d

    RET
```

```
WIDGETS_PROMPT_KEY_ENTER:
    ; La conversión es llevada a cabo por la rutina BCD2HEX.

    ; Se salva la información para poder recuperar el banco de registros:
    PUSH    PSW
    ; Se alterna l banco de registros 1 para que BCD2HEX no altere al actual:
    MACRO_SELECT_BANK_1

    ; Pre-condiciones de la rutina de conversión:
    ; Decenas de millares:
    MOV     R5, widgets_buffer + 0d
    ; Millares
    MOV     R4, widgets_buffer + 1d
    ; Centenas:
    MOV     R3, widgets_buffer + 2d
    ; Decenas
    MOV     R2, widgets_buffer + 3d
    ; Unidades:
    MOV     R1, widgets_buffer + 4d

    CALL    BCD2HEX
    ; BCD2HEX devuelve:
    ; R3 <- parte baja del byte.
    ; R2 <- parte alta del byte.

    MOV     widgets_buffer + 0d, R3
    MOV     widgets_buffer + 1d, R2

    ; Se recupera el banco de registros anterior:
    POP     PSW

    MOV     widgets_buffer + 2d, #0d
    MOV     widgets_buffer + 3d, #0d
    MOV     widgets_buffer + 4d, #0d

    RET

WIDGETS_PROMPT_KEY_NUM:

    ; Chequea si se superó el límite de 5 dígitos (2-bytes):
    ; Salva al valor del acumulador:
    XCH     A, B
    ; Calcula la diferencia entre la dirección base y
    ; la dirección a la que apunta R1:
    MOV     A, R1
    CLR     C
    SUBB    A, #widgets_buffer
    ; Si es 5, vuelve a la parte de espera de teclas:
    CJNE    A, #5d, ($+3+3)
    LJMP    WIDGETS_PROMPT_WAIT

    ; Si no es 5, entonces debe mostrar el nuevo dígito y
    ; actualizar el buffer:

    ; Recupera al acumulador salvado previamente:
    XCH     A, B

    ; Se almacena el nuevo dígito en el buffer y se desplaza el puntero:
    MOV     @R1, A
    INC     R1
    ; Se suma el código ASCII del cero y se muestra en pantalla:
    ADD     A, #'0'
    MOV     display_data, A
    CALL    DISPLAY_PUT_ASCII

    ; Se incrementa la columna y se regresa a la espera de más dígitos:
```

```
    INC     display_column
    JMP     WIDGETS_PROMPT_CURSOR

    ; La instrucción de retorno (RET) se alcanza en
    ; WIDGETS_PROMPT_KEY_ENTER o WIDGETS_PROMPT_KEY_ESC.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
WIDGETS_PROGRESS_BAR:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para mostrar una preciosa barra de progreso en pantalla,
;; con terminaciones cuidadosamente redondeadas.
;;
;; La longitud es de 64 píxeles y la posición es en el centro del display.
;; Sólo se puede especificar la página.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ———————————
;;              DISPLAY_PUT_BYTE
;;
;; Requisitos:
;; ———————————
;;              El "porcentaje" en display_data. Rango: [0;255].
;;                    0 corresponde a   0%
;;                  255 corresponde a 100%
;;
;;              display_page entre 0 y 7
;;
;; Registros alterados:
;; ————————————————————
;;                      A, B, display_column
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se salva pues se la necesta para DISPLAY_PUT_BYTE
    ; y no se desea que esta RUTINA pise al valor:
    PUSH    display_data

    ; Se divide por 4 porque la barra mide 64 píxeles:
    MOV     A, display_data
    MOV     B, #4d
    DIV     AB

    ; Se salva a A:
    XCH     A, B

    ; Primero se dibuja el borde izquierdo:
    MOV     display_data, #00111100b
    MOV     display_column, #(63d - (64d / 2d))
    CALL    DISPLAY_PUT_BYTE

    ; Se recupera a A:
    MOV     A, B
    ; Si es 0, entonces sólo resta dibujar toda la barra vacía:
    JZ      WIDGETS_PROGRESS_BAR_0
    ; Si no es 0, hay que dibujar la parte de la barra rellena:
    MOV     display_data, #01111110b
WIDGETS_PROGRESS_BAR_LOOP_1:
    ; Este bucle dibuja la parte rellena.
```

```
    CALL    DISPLAY_PUT_BYTE
    DJNZ    B, WIDGETS_PROGRESS_BAR_LOOP_1

    ; Si se alcanzó el final, no hace falta completar con una parte vacía:
    MOV     A, display_column
    CJNE    A, #(64d + (64d / 2d)), ($+5)
    SJMP    WIDGETS_PROGRESS_BAR_100

    ; Si no se alcanzó el final, hay que completar con la parte vacía:
WIDGETS_PROGRESS_BAR_0:
    MOV     display_data, #01000010b
WIDGETS_PROGRESS_BAR_LOOP_2:
    ; Este bucle dibuja la parte vacía.
    CALL    DISPLAY_PUT_BYTE
    MOV     A, display_column
    CJNE    A, #(64d + (64d / 2d)), WIDGETS_PROGRESS_BAR_LOOP_2

    ; Finalmente, se dibuja el borde derecho:
WIDGETS_PROGRESS_BAR_100:
    MOV     display_data, #00111100b
    CALL    DISPLAY_PUT_BYTE

    POP     display_data
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

6.11. ensayos.asm

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutinas para manipular ensayos.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;      ENSAYOS_CARGAR_CONTEXTO
;;      ENSAYOS_GRABAR_CONTEXTO
;;      ENSAYOS_CARGAR
;;      ENSAYOS_GRABAR
;;      ENSAYOS_REALIZAR_SERIE
;;      ENSAYOS_REALIZAR_ACTUAL
;;      ENSAYOS_ELIMINAR
;;      ENSAYOS_CONVERTIR_TENSION_ADC2HUMAN
;;      ENSAYOS_CONVERTIR_TENSION_HUMAN2ADC
;;      ENSAYOS_CONVERTIR_PRESION_ADC2HUMAN
;;      ENSAYOS_CONVERTIR_PRESION_HUMAN2ADC
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Definiciones:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Dirección en memoria ROM del primer byte del primer ensayo:
ENSAYOS_BASE_ROM_ADDRESS        EQU ((512d — 32d) * 64d)
; Se define a una distancia del final que permite almacenar hasta 255 ensayos
; de 8 bytes c/u y deja 8 bytes al final para guardar información de contexto.

; La dirección inmediatamente siguiente se reserva para 8 bytes de información
; de contexto:
ENSAYOS_CONTEXTO_ROM_ADDRESS    EQU ENSAYOS_BASE_ROM_ADDRESS + (255d* 8d)

; Puerto de conexión con el relay que permite
; la carga del banco de capacitores:
ENSAYOS_RELAY_BANCO_CAPACITORES EQU     P3.2
; SETB para cerrar la llave, CLR para abrirla.

; Puerto de conexión con el relay que activa el circuito del tiristor:
ENSAYOS_RELAY_TIRISTOR          EQU     P3.4
; SETB para activar, CLR para desactivar.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Macros:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Encapsula la lectura del canal del ADC conectado al punto de medición
;; de la tensión del banco de capacitores.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ENSAYOS_MACRO_LEER_TENSION    MACRO
    ADC_MACRO_READ_CHANNEL_0
    MOV     ensayos_adc_tension, A
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Encapsula la lectura del canal del ADC conectado al sensor de presión.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ENSAYOS_MACRO_LEER_PRESION    MACRO
    ADC_MACRO_READ_CHANNEL_1
    MOV     ensayos_adc_presion, A
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


DSEG


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Variables para almacenar los parámetros de un ensayo:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ensayos_tension:            DS  1
ensayos_presion:            DS  1
ensayos_tiempo_reposo_L:    DS  1
ensayos_tiempo_reposo_H:    DS  1
ensayos_tiempo_descarga:    DS  1
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Variables para almacenar la información de contexto:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ensayos_actual:             DS  1   ; Rango [0;255]
; Significado: indicador del ensayo seleccionado. El primer ensayo de la lista
; se identifica con 0.

ensayos_ultimo:             DS  1   ; Rango [0;255]
; Significado: indicador del último ensayo cargado y válido.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Variables auxiliares para almacenar las lecturas del ADC:
ensayos_adc_tension:        DS  1
ensayos_adc_presion:        DS  1
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


CSEG


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ENSAYOS_MOSTRAR_CABECERA:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Muestra en la primera página del display una cadena de la forma:
;;              -=ENSAYO 002/015=-
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Title_Test_nnn_mmm,    0d, 1d
    WIDGETS_MACRO_PUT_INT               ensayos_actual, #0d,        0d, 10d
    WIDGETS_MACRO_PUT_INT               ensayos_ultimo, #0d,        0d, 14d
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ENSAYOS_CARGAR_CONTEXTO:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Carga desde la ROM la información de contexto.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    MOV     rom_address_L,  #LOW(ENSAYOS_CONTEXTO_ROM_ADDRESS)
    MOV     rom_address_H, #HIGH(ENSAYOS_CONTEXTO_ROM_ADDRESS)

    ; Se salva PSW para poder luego volver al banco de registros correcto:
    PUSH    PSW

    ; Se selecciona el banco de registros 3:
    MACRO_SELECT_BANK_3

    ; La siguiente rutina lee 8 bytes y los devuelve en el banco seleccionado:
    CALL    ROM_READ_TO_BANK

    ; A continuación se obtienen los valores:
    MOV     ensayos_ultimo,   R0
    ; (actualmente sólo se usa el primer byte para almacenar a ensayos_ultimo)

    ; Se recupera PSW para dejar seleccionado al banco de registros correcto:
    POP     PSW
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ENSAYOS_GRABAR_CONTEXTO:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Graba en memoria ROM todas las variables de contexto:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    MOV     rom_address_L,  #LOW(ENSAYOS_CONTEXTO_ROM_ADDRESS)
    MOV     rom_address_H, #HIGH(ENSAYOS_CONTEXTO_ROM_ADDRESS)

    ; Se salva PSW para poder luego volver al banco de registros correcto:
    PUSH    PSW

    ; Se selecciona el banco de registros 3:
    MACRO_SELECT_BANK_3

    ; (actualmente sólo se usa el primer byte para almacenar a ensayos_ultimo)
    MOV     R0, ensayos_ultimo

    ; Se llama a la rutina que guarda en memoria ROM
    ; los 8 bytes del banco de registros seleccionado:
    CALL    ROM_WRITE_FROM_BANK

    ; Se recupera PSW para dejar seleccionado al banco de registros correcto:
    POP     PSW
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ENSAYOS_CARGAR:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Obtiene de memoria ROM a los parámetros del ensayo seleccionado:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; La dirección se obtiene mediante la siguiente aritmética:
```

```
    ; Cada ensayo ocupa 8 bytes:
    MOV     A, #8d
    MOV     B, ensayos_actual
    ; Multiplicando se obtiene la dirección del primer byte del ensayo actual:
    MUL     AB

    ; Se suma ese offset a la dirección base de los ensayos:

    ADD     A, #LOW(ENSAYOS_BASE_ROM_ADDRESS)
    MOV     rom_address_L, A

    MOV     A, B
    ADDC    A, #HIGH(ENSAYOS_BASE_ROM_ADDRESS)
    MOV     rom_address_H, A

    ; Se salva PSW para poder luego volver al banco de registros correcto:
    PUSH    PSW

    ; Se selecciona el banco de registros 3:
    MACRO_SELECT_BANK_3

    ; La siguiente rutina lee 8 bytes y los devuelve en el banco seleccionado:
    CALL    ROM_READ_TO_BANK

    MOV     ensayos_tension,            R0
    MOV     ensayos_presion,            R1
    MOV     ensayos_tiempo_reposo_L,    R2
    MOV     ensayos_tiempo_reposo_H,    R3
    MOV     ensayos_tiempo_descarga,    R4

    ; Se recupera PSW para dejar seleccionado al banco de registros correcto:
    POP     PSW
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ENSAYOS_GRABAR:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Graba en memoria ROM a los parámetros del ensayo seleccionado:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ; La dirección se obtiene mediante la siguiente aritmética:

    ; Cada ensayo ocupa 8 bytes:
    MOV     A, #8d
    MOV     B, ensayos_actual
    ; Multiplicando se obtiene la dirección del primer byte del ensayo actual:
    MUL     AB

    ; Se suma ese offset a la dirección base de los ensayos:

    ADD     A, #LOW(ENSAYOS_BASE_ROM_ADDRESS)
    MOV     rom_address_L, A

    MOV     A, B
    ADDC    A, #HIGH(ENSAYOS_BASE_ROM_ADDRESS)
    MOV     rom_address_H, A

    ; Se salva PSW para poder luego volver al banco de registros correcto:
    PUSH    PSW

    ; Se selecciona el banco de registros 3:
    MACRO_SELECT_BANK_3
```

```
    MOV     R0, ensayos_tension
    MOV     R1, ensayos_presion
    MOV     R2, ensayos_tiempo_reposo_L
    MOV     R3, ensayos_tiempo_reposo_H
    MOV     R4, ensayos_tiempo_descarga
    ; Se llama a la rutina que guarda en memoria ROM
    ; los 8 bytes del banco de registros seleccionado:
    CALL    ROM_WRITE_FROM_BANK

    ; Se recupera PSW para dejar seleccionado al banco de registros correcto:
    POP     PSW
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ENSAYOS_CONVERTIR_TENSION_ADC2HUMAN:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para convertir la lectura del canal del ADC conectado al punto
;; de medición de la tensión del banco de capacitores en un número de 2 bytes
;; expresado en Volts.
;;
;; Se transforma proporcionalmente al intervalo [0;255] en [0;1000].
;;
;; La cuenta que se hace es:    human = adc·1000/255 = adc·200/51
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;              MUL16
;;              DIV16
;;
;; Requisitos:
;; ___________
;;              La variable en el rango del ADC en el Acumulador.
;;
;; Registros alterados:
;; ____________________
;;                      Registros del banco seleccionado
;;                      A <- Parte baja del resultado de 2 bytes.
;;                      B <- Parte alta del resultado de 2 bytes.
;;                      C
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Factor 1:
    ; La parte alta es siempre nula:
    MOV     R6, #0d
    ; La parte baja viene en el acumulador:
    MOV     R7, A

    ; Factor 2:
    MOV     R4, #HIGH(200d)
    MOV     R5,  #LOW(200d)
    CALL    MUL16
    ; El resultado de 4 bytes queda en R0/R1/R2/R3.

    ; Divisor:
    ; Parte alta:
    MOV     A, R2
    MOV     R1, A
```

```
    ; Parte baja:
    MOV     A, R3
    MOV     R0, A

    ; Dividendo:
    MOV     R3, #HIGH(51d)
    MOV     R2,  #LOW(51d)
    CALL    DIV16
    ; El resultado de 2 bytes queda en R3/R2.

    ; Se lo devuelve en B/A:
    MOV     A, R2
    MOV     B, R3

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ENSAYOS_CONVERTIR_TENSION_HUMAN2ADC:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para convertir una tensión expresada en Volts y almacenada en 2 bytes
;; al byte equivalente para el ADC.
;;
;; Se transforma proporcionalmente al intervalo [0;1000] en [0;255].
;;
;; La cuenta que se hace es:    adc = human·255/1000 = human·51/200
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ___________
;;              MUL16
;;              DIV16
;;
;; Requisitos:
;; ___________
;;              La parte alta de la tensión en B.
;;              La parte baja de la tensión en A.
;;
;; Registros alterados:
;; ____________________
;;                      Registros del banco seleccionado
;;                      A <- byte para el ADC.
;;                      B, C
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Factor 1:
    ; La parte alta viene en B:
    MOV     R6, B
    ; La parte baja viene en A:
    MOV     R7, A

    ; Factor 2:
    MOV     R4, #HIGH(51d)
    MOV     R5,  #LOW(51d)
    CALL    MUL16
    ; El resultado de 4 bytes queda en R0/R1/R2/R3.

    ; Divisor:
    ; Parte alta:
    MOV     A, R2
```

```
    MOV     R1, A
    ; Parte baja:
    MOV     A, R3
    MOV     R0, A

    ; Dividendo:
    MOV     R3, #HIGH(200d)
    MOV     R2,  #LOW(200d)
    CALL    DIV16
    ; El resultado de 2 bytes queda en R3/R2.

    ; Se devuelve en A a la parte baja del resultado:
    MOV     A, R2

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ENSAYOS_CONVERTIR_PRESION_ADC2HUMAN:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para convertir la lectura del canal del ADC conectado al sensor
;; de presión en un número de 2 bytes expresado en kilogramos.
;;
;; Se transforma proporcionalmente al intervalo [0;255] en [0;20000].
;;
;; La cuenta que se hace es:   human = adc·20000/255 = adc·4000/51
;;
;; Pero para evitar problemas de redondeo se hace en tres etapas:
;;          adc·40  ——————>  /51  ——————>  ·100
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ———————————
;;              DIV16
;;
;; Requisitos:
;; ———————————
;;              La variable en el rango del ADC en el Acumulador.
;;
;; Registros alterados:
;; ————————————————————
;;                      Registros del banco seleccionado
;;                      A <— Parte baja del resultado de 2 bytes.
;;                      B <— Parte alta del resultado de 2 bytes.
;;                      C
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se multiplica al valor del adc que viene en A por 40:
    MOV     B, #40d
    MUL     AB

    ; Divisor:
    ; Parte alta:
    MOV     R1, B
    ; Parte baja:
    MOV     R0, A

    ; Dividendo:
    MOV     R3, #HIGH(51d)
    MOV     R2,  #LOW(51d)
```

```
    CALL    DIV16
    ; El resultado de 2 bytes queda en R3/R2.

    ; Se multiplica por 100 a la parte baja del cociente:
    MOV     A, R2
    MOV     B, #100d
    MUL     AB
    ; A <- parte baja del producto.
    ; B <- parte alta del producto.

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ENSAYOS_CONVERTIR_PRESION_HUMAN2ADC:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para convertir una presión expresada en kilogramos
;; y almacenada en 2 bytes al byte equivalente para el ADC.
;;
;; Se transforma proporcionalmente al intervalo [0;20000] en [0;255].
;;
;; La cuenta que se hace es:    adc = human·255/20000 = human·51/4000
;;
;; Pero para evitar problemas de redondeo se hace en tres etapas:
;;          human/40  ———>  ·51  ———>  /100
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Depende de:
;; ———————————
;;              MUL16
;;              DIV16
;;
;; Requisitos:
;; ———————————
;;              La parte alta de la presión en B.
;;              La parte baja de la presión en A.
;;
;; Registros alterados:
;; ————————————————————
;;                      Registros del banco seleccionado
;;                      A <- byte para el ADC.
;;                      B, C
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Divisor:
    ; Parte alta:
    MOV     R1, B
    ; Parte baja:
    MOV     R0, A

    ; Dividendo:
    MOV     R3, #HIGH(40d)
    MOV     R2,  #LOW(40d)
    CALL    DIV16
    ; El resultado de 2 bytes queda en R3/R2.

    ; Factor 1:
    ; Parte alta:
    MOV     A, R3
    MOV     R6, A
```

```
    ; Parte baja:
    MOV     A, R2
    MOV     R7, A

    ; Factor 2:
    MOV     R4, #HIGH(51d)
    MOV     R5,  #LOW(51d)
    CALL    MUL16
    ; El resultado de 4 bytes queda en R0/R1/R2/R3.

    ; Divisor:
    ; Parte alta:
    MOV     A, R2
    MOV     R1, A
    ; Parte baja:
    MOV     A, R3
    MOV     R0, A

    ; Dividendo:
    MOV     R3, #HIGH(100d)
    MOV     R2,  #LOW(100d)
    CALL    DIV16
    ; El resultado de 2 bytes queda en R3/R2.

    ; Se devuelve la parte bajar del cociente:
    MOV     A, R2

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ENSAYOS_REALIZAR_SERIE:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para ejecutar los ensayos en sucesión.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se carga el contexto:
    CALL    ENSAYOS_CARGAR_CONTEXTO

    ; Se comienza por el ensayo 0:
    MOV     ensayos_actual, #0d

ENSAYOS_REALIZAR_SERIE_CYCLE:

    ; Se llama a la rutina que ejecuta al ensayo seleccionado:
    CALL    ENSAYOS_REALIZAR_ACTUAL

    ; Si actual = último, entonces termina:
    MOV     A, ensayos_actual
    CJNE    A, ensayos_ultimo, ($+3+2)
    SJMP    ENSAYOS_REALIZAR_SERIE_FIN

    ; Si actual != último, entonces itera al siguiente:
    INC     ensayos_actual
    SJMP    ENSAYOS_REALIZAR_SERIE_CYCLE

ENSAYOS_REALIZAR_SERIE_FIN:
    CALL    DISPLAY_CLEAR

    ; Se informa que todos los ensayos han finalizado:
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_All_tests_completed,   3d, 0d
```

```
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Enter_to_continue,     7d, 0d

    KEYPAD_MACRO_WAIT_KEY               KEYPAD_ENTER

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ENSAYOS_REALIZAR_ACTUAL:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para ejecutar al ensayo seleccionado:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se cargan las variables con la información del ensayo a realizar:
    CALL    ENSAYOS_CARGAR

    CALL    DISPLAY_CLEAR

    ; Se informa el ensayo actual en la cabecera:
    CALL    ENSAYOS_MOSTRAR_CABECERA

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Primero se le pide al operador que confirme que la muestra ya está lista
;; en el compartimiento:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Load_sample,       3d, 0d
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Enter_to_continue, 7d, 0d

    KEYPAD_MACRO_WAIT_KEY               KEYPAD_ENTER

    CALL    DISPLAY_CLEAR

    ; Se informa el ensayo actual en la cabecera:
    CALL    ENSAYOS_MOSTRAR_CABECERA

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Luego se espera a que el operario alcance con la prensa el valor de presión
;; necesitado. Se muestran en pantalla la lectura del canal del ADC conectado
;; al sensor de presión y el valor que se debe alcanzar:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Reach_preassure,   2d, 0d
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Enter_to_continue, 7d, 0d

    ; Se realiza la conversión [0;255] ---> [0;20000]:
    MOV     A, ensayos_presion
    CALL    ENSAYOS_CONVERTIR_PRESION_ADC2HUMAN
    ; Se muestra la presión objetivo:
    WIDGETS_MACRO_PUT_INT   A, B, 6d, 7d

ENSAYOS_REALIZAR_ACTUAL_PRESION_LOOP:

    ; Se toma una lectura del ADC:
    ENSAYOS_MACRO_LEER_PRESION
    ; El resultado queda en ensayos_adc_presion.


    ; Se realiza la conversión [0;255] ---> [0;20000]:
    MOV     A, ensayos_adc_presion
    CALL    ENSAYOS_CONVERTIR_PRESION_ADC2HUMAN
    ; Se muestra el valor actual de presión:
    WIDGETS_MACRO_PUT_INT   A, B, 4d, 7d
```

```
    ; Se construye una barra de progreso a partir
    ; del valor actual y el valor objetivo:
    WIDGETS_MACRO_PROGRESS_BAR_DIFF ensayos_adc_presion, ensayos_presion, 5d

    ; Se llama a una rutina que permite iterar un bucle arbitrario
    ; hasta que se presione alguna tecla:
    MOV     DPTR, #ENSAYOS_REALIZAR_ACTUAL_PRESION_LOOP
    CALL    KEYPAD_SCAN_JMP

    CALL    DISPLAY_CLEAR

    ; Se informa el ensayo actual en la cabecera:
    CALL    ENSAYOS_MOSTRAR_CABECERA

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; A continuación se espera que pase el tiempo en el que la muestra debe
;; dejar reposar bajo presión.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Settling_sample,       2d, 0d

    ; Se muestra la cantidad de segundos a esperar:
    WIDGETS_MACRO_PUT_INT ensayos_tiempo_reposo_L,ensayos_tiempo_reposo_H,6d,7d

    ; Se inicia el cronómetro de segundos:
    CALL    TIMERS_START_SECONDS_CRON

ENSAYOS_REALIZAR_ACTUAL_REPOSO_LOOP:

    ; Se muestra el tiempo transcurrido:
    WIDGETS_MACRO_PUT_INT timers_seconds_L, timers_seconds_H, 4d, 7d

    ; Se itera hasta que se alcance el tiempo necesario:
ENSAYOS_REALIZAR_ACTUAL_REPOSO_RETRY:
    MOV     A, timers_seconds_H
    MOV     R6, timers_seconds_L
    CJNE    A, timers_seconds_H, ENSAYOS_REALIZAR_ACTUAL_REPOSO_RETRY
    MOV     R7, A
    MOV     A, R6
    CJNE    A, ensayos_tiempo_reposo_L, ENSAYOS_REALIZAR_ACTUAL_REPOSO_LOOP
    MOV     A, R7
    CJNE    A, ensayos_tiempo_reposo_H, ENSAYOS_REALIZAR_ACTUAL_REPOSO_LOOP

    ; Se detiene al cronómetro de segundos:
    CALL    TIMERS_STOP_SECONDS_CRON
    CALL    DISPLAY_CLEAR
    ; Se informa el ensayo actual en la cabecera:
    CALL    ENSAYOS_MOSTRAR_CABECERA
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Luego se comienza la carga de los capacitores. Se muestra en pantalla
;; la lectura del canal del ADC conectado al punto de medición de tensión.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Charging_capacitors,   2d, 0d

    ; Se realiza la conversión [0;255] ---> [0;1000]:
    MOV     A, ensayos_tension
    CALL    ENSAYOS_CONVERTIR_TENSION_ADC2HUMAN
    ; Se muestra el valor objetivo:
    WIDGETS_MACRO_PUT_INT   A , B, 6d, 7d
    ; Por seguridad, se asegura que el banco de capacidores esté desconectado
    ; de la segunda etapa del circuito:
    CLR     ENSAYOS_RELAY_TIRISTOR
    ; Se cierra la llave que permite la carga:
    SETB    ENSAYOS_RELAY_BANCO_CAPACITORES

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ENSAYOS_REALIZAR_ACTUAL_TENSION_LOOP:

    ; Por seguridad, se asegura en cada iteración que la llave esté cerrada:
    SETB    ENSAYOS_RELAY_BANCO_CAPACITORES

    ENSAYOS_MACRO_LEER_TENSION

    ; Se realiza la conversión [0;255] ---> [0;1000]:
    MOV     A, ensayos_adc_tension
    CALL    ENSAYOS_CONVERTIR_TENSION_ADC2HUMAN
    ; Se muestra la tensión medida:
    WIDGETS_MACRO_PUT_INT   A, B, 4d, 7d

    ; Se muestra una barra de progreso:
    WIDGETS_MACRO_PROGRESS_BAR_DIFF ensayos_adc_tension, ensayos_tension, 5d

    ; Se compara la lectura del ADC con el objetivo:
    MOV     A, ensayos_adc_tension
    CJNE    A, ensayos_tension, ($+3)
    ; C = actual < objetivo.
    JC      ENSAYOS_REALIZAR_ACTUAL_TENSION_LOOP


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se detiene la carga del banco de capacitores:
    CLR     ENSAYOS_RELAY_BANCO_CAPACITORES

    CALL    DELAY_ONE_SECOND
    CALL    DELAY_ONE_SECOND

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se activa el circuito del tiristor para descargar los capacitores
    ; sobre la muestra:
    SETB    ENSAYOS_RELAY_TIRISTOR

    ; Se inicia al cronómetro de milisegundos:
    CALL    TIMERS_START_MSECS_CRON

ENSAYOS_REALIZAR_ACTUAL_TIEMPO_DESCARGA_LOOP:

    MOV     A, timers_msecs
    CJNE    A, ensayos_tiempo_descarga, ($+3)
    ; C = actual < objetivo.
    JC      ENSAYOS_REALIZAR_ACTUAL_TIEMPO_DESCARGA_LOOP

    CLR     ENSAYOS_RELAY_TIRISTOR

    ; Se detiene al cronómetro de milisegundos:
    CALL    TIMERS_STOP_MSECS_CRON

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    CALL    DISPLAY_CLEAR
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Spark, 3d, 3d

    CALL    DELAY_ONE_SECOND
    CALL    DELAY_ONE_SECOND

    CALL    DISPLAY_CLEAR
```

```
    ; Se informa el ensayo actual en la cabecera:
    CALL    ENSAYOS_MOSTRAR_CABECERA

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Antes de terminar, se espera que el operario retire la muestra
;; del compartimiento.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Take_sample,       3d, 0d
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Enter_to_continue, 7d, 0d

    KEYPAD_MACRO_WAIT_KEY               KEYPAD_ENTER

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ENSAYOS_ELIMINAR:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; "Elimina" un ensayo de la lista. En realidad lo que hace es intercambiarlo
;; por el último y reducir la lista.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se carga de ROM la información de contexto, para conocer
    ; la posición del último:
    CALL    ENSAYOS_CARGAR_CONTEXTO

    ; Si sólo hay un ensayo en la lista, se ignora la llamada:
    MOV     A, ensayos_ultimo
    JZ      ENSAYOS_ELIMINAR_END

    ; Si se quizo eliminar al último, sólo se necesita
    ; decrementar a ensayos_ultimo:
    CJNE    A, ensayos_actual, ($+3+2)
    SJMP    ENSAYOS_ELIMINAR_ULTIMO

    ; Se salva la posición del ensayo que se desea eliminar:
    PUSH    ensayos_actual

    ; Se mueve hasta el final y carga en RAM al último ensayo:
    MOV     ensayos_actual, ensayos_ultimo
    CALL    ENSAYOS_CARGAR

    ; Se sobrescribe al actual por el último:
    POP     ensayos_actual
    CALL    ENSAYOS_GRABAR

ENSAYOS_ELIMINAR_ULTIMO:

    ; Y finalmente decrementa la posición del último y graba el contexto:
    DEC     ensayos_ultimo
    CALL    ENSAYOS_GRABAR_CONTEXTO

ENSAYOS_ELIMINAR_END:
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

6.12. interfaz.asm

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Descripción de la interfaz con el usuario.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Las "rutinas" que aquí se definen, no deben ser llamadas nunca con CALL.
;; Fueron concebidas para tomar un control temprano de la ejecución y luego
;; guiar todo el comportamiento.
;;
;; El punto de inicio recomendado es INTERFAZ_MENU_PRINCIPAL y debe ser
;; alcanzado con una instrucción del tipo JMP.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;      INTERFAZ_MENU_PRINCIPAL
;;      INTERFAZ_MENU_EDITAR_ENSAYO
;;      INTERFAZ_ANILLO_ENSAYOS
;;      INTERFAZ_ANILLO_ENSAYOS_CYCLE
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

CSEG


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INTERFAZ_MENU_PRINCIPAL:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    CALL    DISPLAY_CLEAR

    ; Se pasa el control a la rutina que muestra el texto de las opciones
    ; de un menú y permite recorrerlas:
    MOV     DPTR, #i18n_Main_menu_labels
    CALL    WIDGETS_MENU_INTERACT
    ; Al volver, se obtiene la opción elegida en widgets_menu_option
    ; y la tecla presionada en keypad_pressed_key.

    ; Si se salió del menú presionando ESC, se vuelve a mostrar a sí mismo:
    MOV     A, keypad_pressed_key
    CJNE    A, #KEYPAD_ESC, ($+3+2)
    SJMP    INTERFAZ_MENU_PRINCIPAL

    ; La estructura siguiente es equivalente a el típico SWITCH
    ; de los lenguajes de nivel más alto:

    ; Switch (widgets_menu_option):
    MOV     A, widgets_menu_option

    ; Case 0:
    CJNE    A, #0d, ($+5)
    SJMP    INTERFAZ_MENU_PRINCIPAL_OPTION_0
    ; Case 1:
    CJNE    A, #1d, ($+5)
    SJMP    INTERFAZ_MENU_PRINCIPAL_OPTION_1
    ; Case 2:
    CJNE    A, #2d, ($+5)
    SJMP    INTERFAZ_MENU_PRINCIPAL_OPTION_2
    ; Case 3:
```

```
    CJNE    A, #3d, ($+5)
    SJMP    INTERFAZ_MENU_PRINCIPAL_OPTION_3
    ; Default:
    SJMP    INTERFAZ_MENU_PRINCIPAL

INTERFAZ_MENU_PRINCIPAL_OPTION_0:
    ; La opción 0 muestra al anillo de ensayos:
    JMP     INTERFAZ_ANILLO_ENSAYOS

INTERFAZ_MENU_PRINCIPAL_OPTION_1:
    ; La opción 1 permite agregar un ensayo a la lista.

    ; Primero se carga desde ROM la información necesaria para saber
    ; si queda lugar para agregar otro ensayo:
    CALL    ENSAYOS_CARGAR_CONTEXTO

    ; Si el último ensayo no está en la posición 255, entonces hay lugar:
    MOV     A, ensayos_ultimo
    CJNE    A, #255d, INTERFAZ_MENU_PRINCIPAL_OPTION_1_HAY_ESPACIO

    ; Si el último ensayo está en la posición 255, entonces no hay más lugar:
    CALL    DISPLAY_CLEAR

    ; Se muestra un texto informando sobre el incidente:
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Out_of_space,      3d, 0d
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Enter_to_continue, 7d, 0d

    ; Se espera a que el usuario presione ENTER
    ; antes de volver al menú principal:
    KEYPAD_MACRO_WAIT_KEY   KEYPAD_ENTER
    JMP     INTERFAZ_MENU_PRINCIPAL

INTERFAZ_MENU_PRINCIPAL_OPTION_1_HAY_ESPACIO:
    ; Se incrementa la posición del último ensayo,
    ; se lo apunta y se llama a la rutina que graba el contexto
    ; para actualizar el valor de ensayos_ultimo en ROM:
    INC     ensayos_ultimo
    MOV     ensayos_actual, ensayos_ultimo
    CALL    ENSAYOS_GRABAR_CONTEXTO

    ; Se inicializan los valores del nuevo experimento:
    MOV     ensayos_tension,            #0d
    MOV     ensayos_presion,            #0d
    MOV     ensayos_tiempo_reposo_L,    #0d
    MOV     ensayos_tiempo_descarga,    #0d

    ; Se pasa el control a la interfaz que permite editar
    ; los parámetros del ensayo seleccionado, que en este caso
    ; es el que acaba de crearse:
    JMP     INTERFAZ_MENU_EDITAR_ENSAYO

INTERFAZ_MENU_PRINCIPAL_OPTION_2:
    ; Se llama a la rutina que realiza la serie de ensayos
    ; almacenador en ROM:
    CALL    ENSAYOS_REALIZAR_SERIE

    ; Al finalizar, se vuelve a mostrar el menú principal:
    JMP     INTERFAZ_MENU_PRINCIPAL

INTERFAZ_MENU_PRINCIPAL_OPTION_3:
    ; Menú loco:
    JMP     INTERFAZ_MENU_EXTRAS
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INTERFAZ_MENU_EDITAR_ENSAYO:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    CALL    DISPLAY_CLEAR

    ; Se pasa el control a la rutina que muestra el texto de las opciones
    ; de un menú y permite recorrerlas:
    MOV     DPTR, #i18n_Edit_menu_labels
    CALL    WIDGETS_MENU_INTERACT
    ; Al volver, se obtiene la opción elegida en widgets_menu_option
    ; y la tecla presionada en keypad_pressed_key.

    ; Si se salió del menú presionando ESC, se cancela la edición
    ; y se pasa el control al ciclo del anillo de ensayos.
    ; Es importante notar que al no seguir la estructura de CALL/RET,
    ; se volverá al ciclo del anillo de ensayos
    ; independientemente de la ubicación anterior.
    MOV     A, keypad_pressed_key
    CJNE    A, #KEYPAD_ESC, ($+3+3)
    LJMP    INTERFAZ_ANILLO_ENSAYOS_CYCLE

    ; La estructura siguiente es equivalente a el típico SWITCH
    ; de los lenguajes de nivel más alto:

    ; Switch (widgets_menu_option):
    MOV     A, widgets_menu_option

    ; Case 0:
    CJNE    A, #0d, ($+3+2)
    SJMP    INTERFAZ_MENU_EDITAR_ENSAYO_OPTION_0
    ; Case 1:
    CJNE    A, #1d, ($+3+2)
    SJMP    INTERFAZ_MENU_EDITAR_ENSAYO_OPTION_1
    ; Case 2:
    CJNE    A, #2d, ($+3+2)
    SJMP    INTERFAZ_MENU_EDITAR_ENSAYO_OPTION_2
    ; Case 3:
    CJNE    A, #3d, ($+3+2)
    SJMP    INTERFAZ_MENU_EDITAR_ENSAYO_OPTION_3
    ; Case 4:
    CJNE    A, #4d, ($+3+2)
    SJMP    INTERFAZ_MENU_EDITAR_ENSAYO_OPTION_4
    ; Default:
    JMP     INTERFAZ_MENU_EDITAR_ENSAYO

    ; La estructura de las opciones 0 a 3 es idéntica y consiste en
    ; llamar a la rutina que lee dígitos desde el teclado y los
    ; transforma en números.
INTERFAZ_MENU_EDITAR_ENSAYO_OPTION_0:
    ; Lectura de la tensión:
    WIDGETS_MACRO_PROMPT    A, B,                       7d, 6d
    ; A <- parte baja del valor de 2 bytes ingresado en Volts.
    ; B <- parte alta del valor de 2 bytes ingresado en Volts.
    CALL    ENSAYOS_CONVERTIR_TENSION_HUMAN2ADC
    MOV     ensayos_tension, A

    JMP     INTERFAZ_MENU_EDITAR_ENSAYO
INTERFAZ_MENU_EDITAR_ENSAYO_OPTION_1:
    ; Lectura de la presión:
    WIDGETS_MACRO_PROMPT    A, B,                       7d, 6d
    ; A <- parte baja del valor de 2 bytes ingresado en kilogramos.
    ; B <- parte alta del valor de 2 bytes ingresado en kilogramos.
    CALL    ENSAYOS_CONVERTIR_PRESION_HUMAN2ADC
```

```
    MOV     ensayos_presion, A
    JMP     INTERFAZ_MENU_EDITAR_ENSAYO
INTERFAZ_MENU_EDITAR_ENSAYO_OPTION_2:

    WIDGETS_MACRO_PROMPT ensayos_tiempo_reposo_L, ensayos_tiempo_reposo_H,7d,6d

    JMP     INTERFAZ_MENU_EDITAR_ENSAYO
INTERFAZ_MENU_EDITAR_ENSAYO_OPTION_3:

    WIDGETS_MACRO_PROMPT    ensayos_tiempo_descarga, Acc, 7d, 6d

    JMP     INTERFAZ_MENU_EDITAR_ENSAYO


INTERFAZ_MENU_EDITAR_ENSAYO_OPTION_4:
    ; Esta opción muestra en pantalla todos los datos ingresados
    ; y da la oportunidad de confirmarlos o descartarlos.

    CALL    DISPLAY_CLEAR

    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Title_Confirm_data,    0d, 0d

    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Voltage,               1d, 0d
    MOV     A, ensayos_tension
    CALL    ENSAYOS_CONVERTIR_TENSION_ADC2HUMAN
    WIDGETS_MACRO_PUT_INT               A, B,                       1d, 15d

    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Preassure,             2d, 0d
    MOV     A, ensayos_presion
    CALL    ENSAYOS_CONVERTIR_PRESION_ADC2HUMAN
    WIDGETS_MACRO_PUT_INT               A, B,                       2d, 15d

    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Settling_time,         3d, 0d
    WIDGETS_MACRO_PUT_INT ensayos_tiempo_reposo_L,ensayos_tiempo_reposo_H,3d,15d

    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Discharge_time,        4d, 0d
    WIDGETS_MACRO_PUT_INT               ensayos_tiempo_descarga, #0d, 4d, 15d

    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Enter_to_accept,       6d, 0d
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Esc_to_cancel,         7d, 0d


INTERFAZ_MENU_EDITAR_ENSAYO_WAIT_ENTER:

    ; Se detiene la ejecución hasta que se presione alguna tecla:
    CALL    KEYPAD_SCAN_WAIT

    ; Si se presiona ESC es porque no se aceptan los datos
    ; y se quiere editarlos:
    MOV     A, keypad_pressed_key
    CJNE    A, #KEYPAD_ESC, ($+3+3)
    LJMP    INTERFAZ_MENU_EDITAR_ENSAYO

    ; Hasta que no se presione ESC o ENTER no se continua:
    CJNE    A, #KEYPAD_ENTER, INTERFAZ_MENU_EDITAR_ENSAYO_WAIT_ENTER


    ; Si se presiona ENTER se guardan los datos en ROM:
    CALL    DISPLAY_CLEAR

    DISPLAY_MACRO_PUT_STRING_STATIC i18n_Saving_data, 3d, 0d

    CALL    ENSAYOS_GRABAR

    ; Delay ficticio:
    CALL    DELAY_ONE_SECOND
```

```
    ; Se podría agregar una progress bar para hacerlo más fantoche todavía.

    ; Al finalizar, se regresa al ciclo del anillo de ensayos:
    JMP     INTERFAZ_ANILLO_ENSAYOS_CYCLE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INTERFAZ_ANILLO_ENSAYOS:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se comienza mostrando el primer ensayo de la lista:
    MOV     ensayos_actual, #0d

    ; Se carga desde ROM toda la información de contexto para saber
    ; cuantos ensayos hay cargados:
    CALL    ENSAYOS_CARGAR_CONTEXTO

INTERFAZ_ANILLO_ENSAYOS_CYCLE:

    CALL    DISPLAY_CLEAR

    ; Se cargan de ROM los datos del experimento actual
    ; para poder mostrarlos:
    CALL    ENSAYOS_CARGAR

    CALL    DISPLAY_CLEAR

    ; Se informa en la cabecera el número del experimento actual:
    CALL    ENSAYOS_MOSTRAR_CABECERA

    ; Se muestran los parámetros del ensayo seleccionado:

    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Voltage,               1d, 0d
    MOV     A, ensayos_tension
    CALL    ENSAYOS_CONVERTIR_TENSION_ADC2HUMAN
    WIDGETS_MACRO_PUT_INT               A, B,       1d, 15d


    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Preassure,             2d, 0d
    MOV     A, ensayos_presion
    CALL    ENSAYOS_CONVERTIR_PRESION_ADC2HUMAN
    WIDGETS_MACRO_PUT_INT               A, B,       2d, 15d


    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Settling_time,         3d, 0d
    WIDGETS_MACRO_PUT_INT ensayos_tiempo_reposo_L,ensayos_tiempo_reposo_H,3d,15d

    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Discharge_time,        4d, 0d
    WIDGETS_MACRO_PUT_INT               ensayos_tiempo_descarga,#0d, 4d, 15d


    ; Se muestran las opciones con sus teclas asociadas:
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Enter_for_edit,        5d, 0d
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_F1_for_excute,         6d, 0d
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_DEL_for_delete,        7d, 0d


    ; Se detiene la ejecución hasta que se presione alguna tecla:
    CALL    KEYPAD_SCAN_WAIT

    ; La estructura siguiente es equivalente a el típico SWITCH
    ; de los lenguajes de nivel más alto.
```

```
    ; Se acompaña con su cuasi—equivalente en lenguaje C.

    ; Switch (keypad_pressed_key):
    MOV     A, keypad_pressed_key

    ; Case KEYPAD_RIGHT:
    CJNE    A, #KEYPAD_RIGHT, ($+3+2)
    SJMP    INTERFAZ_ANILLO_ENSAYOS_RIGHT
    ; Case KEYPAD_LEFT:
    CJNE    A, #KEYPAD_LEFT, ($+3+2)
    SJMP    INTERFAZ_ANILLO_ENSAYOS_LEFT
    ; Case KEYPAD_HELP:
    CJNE    A, #KEYPAD_HELP, ($+3+2)
    SJMP    INTERFAZ_ANILLO_ENSAYOS_HELP
    ; Case KEYPAD_ESC:
    CJNE    A, #KEYPAD_ESC, ($+3+3)
    LJMP    INTERFAZ_ANILLO_ENSAYOS_ESC
    ; Case KEYPAD_ENTER:
    CJNE    A, #KEYPAD_ENTER, ($+3+3)
    LJMP    INTERFAZ_ANILLO_ENSAYOS_ENTER
    ; Case KEYPAD_DEL:
    CJNE    A, #KEYPAD_DEL, ($+3+3)
    LJMP    INTERFAZ_ANILLO_ENSAYOS_DEL
    ; Default:
    JMP     INTERFAZ_ANILLO_ENSAYOS_CYCLE


INTERFAZ_ANILLO_ENSAYOS_RIGHT:
    ; Si se presiona esta tecla, y no se superó al último,
    ; hay que mostrar al próximo:
    MOV     A, ensayos_actual
    CJNE    A, ensayos_ultimo, INTERFAZ_ANILLO_ENSAYOS_RIGHT_NOT_LAST

    ; Si se superó al último, se vuelve al primero (topología circular):
    MOV     ensayos_actual, #0d
    JMP     INTERFAZ_ANILLO_ENSAYOS_CYCLE

INTERFAZ_ANILLO_ENSAYOS_RIGHT_NOT_LAST:
    INC     ensayos_actual
    JMP     INTERFAZ_ANILLO_ENSAYOS_CYCLE

INTERFAZ_ANILLO_ENSAYOS_LEFT:
    ; Si se presiona esta tecla, y no se superó al primero,
    ; hay que mostrar al anterio:
    MOV     A, ensayos_actual
    CJNE    A, #0d, INTERFAZ_ANILLO_ENSAYOS_LEFT_NOT_LAST

    ; Si se superó al primero, se vuelve al último (topología circular):
    MOV     ensayos_actual, ensayos_ultimo
    JMP     INTERFAZ_ANILLO_ENSAYOS_CYCLE

INTERFAZ_ANILLO_ENSAYOS_LEFT_NOT_LAST:
    DEC     ensayos_actual
    JMP     INTERFAZ_ANILLO_ENSAYOS_CYCLE

INTERFAZ_ANILLO_ENSAYOS_HELP:
    ; Si se presiona HELP, se llama a la rutina que
    ; ejecuta el ensayo seleccionado:
    CALL    ENSAYOS_REALIZAR_ACTUAL
    JMP     INTERFAZ_ANILLO_ENSAYOS_CYCLE

INTERFAZ_ANILLO_ENSAYOS_DEL:

    CALL    DISPLAY_CLEAR

    ; Se informa en qué consiste el procedimiento de "borrado":
```

```
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Delete_message_A,      2d, 0d
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Delete_message_B,      3d, 0d
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Delete_message_C,      4d, 0d

    ; Se muestran las opciones con sus teclas asociadas:
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Enter_to_accept,       6d, 0d
    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Esc_to_cancel,         7d, 0d

INTERFAZ_ANILLO_ENSAYOS_DEL_WAIT_ENTER_OR_ESC:

    ; Se detiene la ejecución hasta que se presione alguna tecla:
    CALL    KEYPAD_SCAN_WAIT

    ; Si se presiona ESC es porque se arrepiente de eliminar:
    MOV     A, keypad_pressed_key
    CJNE    A, #KEYPAD_ESC, ($+3+3)
    LJMP    INTERFAZ_ANILLO_ENSAYOS_CYCLE

    ; Hasta que no se presione ENTER o ESC no continua:
    CJNE    A, #KEYPAD_ENTER, INTERFAZ_ANILLO_ENSAYOS_DEL_WAIT_ENTER_OR_ESC

    ; Si se presiona ENTER se procede a eliminar el ensayo:
    CALL    DISPLAY_CLEAR

    DISPLAY_MACRO_PUT_STRING_STATIC     i18n_Updating_data,         3d, 0d

    CALL    ENSAYOS_ELIMINAR

    ; Delay ficticio:
    CALL    DELAY_ONE_SECOND

    ; Se recarga la información de contexto para actualizar la posición del
    ; último:
    CALL    ENSAYOS_CARGAR_CONTEXTO

    ;; Si el que se acaba de eliminar era el último, entonces se debe
    ;; ajustar la posición antes de salir, para evitar comportamientos
    ;; anómalos:
    MOV     A, ensayos_ultimo
    CJNE    A, ensayos_actual, ($+3)
    ; Si ultimo < actual, hay que decrementar actual:
    JNC     ($+2+2)
    DEC     ensayos_actual



    JMP     INTERFAZ_ANILLO_ENSAYOS_CYCLE

INTERFAZ_ANILLO_ENSAYOS_ENTER:
    ; Si se presiona enter, se pasa el control al menú
    ; para editar parámetros de un ensayo:
    JMP     INTERFAZ_MENU_EDITAR_ENSAYO

INTERFAZ_ANILLO_ENSAYOS_ESC:
    JMP     INTERFAZ_MENU_PRINCIPAL
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

6.13. screens.asm

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutinas para manipular pantallas gráficas.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Definiciones:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Dirección en memoria ROM del logo de FIUBA:
SCREENS_LOGOFIUBA_ADDRESS   EQU     1000H
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Macros:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Simplifica la grabación de una pantalla desde memoria de código a ROM.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SCREENS_MACRO_WRITE_FROM_CODE   MACRO   rom_address, code_addres
    MOV     rom_address_L,  #LOW(rom_address)
    MOV     rom_address_H,  #HIGH(rom_address)
    MOV     DPTR, #code_addres
    CALL    SCREENS_WRITE_TO_ROM_FROM_CODE
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


CSEG


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SCREENS_WRITE_TO_ROM_FROM_CODE:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para enviar una pantalla de 1024 bytes desde memoria de código
;; hacia ROM.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Requisitos:
;; ------------
;;              La dirección del primer byte en DPTR.
;;
;; Registros alterados:
;; ---------------------
;;                      B, A, C
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ROM_MACRO_SELECT
    MOV     B, #16d
SCREENS_WRITE_TO_ROM_FROM_CODE_LOOP:
    CALL    ROM_WRITE_PAGE_FROM_CODE
    CALL    ROM_INC_ADDRESS_64
    DJNZ    B, SCREENS_WRITE_TO_ROM_FROM_CODE_LOOP
```

```
    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SCREENS_DUMP_TO_DISPLAY_FROM_ROM:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para mostrar una pantalla grabada en ROM en el display.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Requisitos:
;; ___________
;;
;;
;; Registros alterados:
;; ____________________
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    CALL    DISPLAY_OFF
    MOV     B, #64d
    CALL    SCREENS_DUMP_TO_DISPLAY_FROM_ROM_SECTOR

    MOV     B, #128d
    CALL    SCREENS_DUMP_TO_DISPLAY_FROM_ROM_SECTOR
    CALL    DISPLAY_ON
    RET

SCREENS_DUMP_TO_DISPLAY_FROM_ROM_SECTOR:
    MOV     display_page, #0d
SCREENS_DUMP_TO_DISPLAY_FROM_ROM_LOOP_PAGES:
    CALL    ROM_READ_PAGE_TO_BUFFER
    CALL    ROM_INC_ADDRESS_64

    MOV     A, B
    CLR     C
    SUBB    A, #64d
    MOV     display_column, A
    MOV     R0, #rom_page_buffer
SCREENS_DUMP_TO_DISPLAY_FROM_ROM_LOOP_COLUMNS:
    MOV     A, @R0
    MOV     display_data, A
    CALL    DISPLAY_PUT_BYTE
    INC     R0
    MOV     A, display_column
    CJNE    A, B, SCREENS_DUMP_TO_DISPLAY_FROM_ROM_LOOP_COLUMNS

    INC     display_page
    MOV     A, display_page
    CJNE    A, #8d, SCREENS_DUMP_TO_DISPLAY_FROM_ROM_LOOP_PAGES

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SCREEN:
; Esta línea debe ser habilitada cuando se desee ensamblar un programa
; para cargar una imagen desde memoria de código a memoria ROM:
;$INCLUDE(imagenes/logofiuba.asm)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

6.14. misc.asm

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Conjunto de rutinas y definiciones misceláneas.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;  Rutinas disponibles:
;;  ____________________
;;                      BCD2HEX: convierte 5 dígitos BCD a HEX.
;;                      HEX2ASCII: convierte 2 bytes HEX a 5 dígitos ASCII.
;;                      DIV16: realiza divisiones enteras de 16 bits.
;;                      MUL16: realiza multiplicaciones enteras de 16 bits.
;;                      DELAY_ONE_SECOND: delay por software de 1 segundo.
;;                      DELAY_ONE_MSEC: delay por software de 1 milisegundo.
;;                      DELAY_TEN_MSEC: delay por software de 10 milisegundos.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

CSEG

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
BCD2HEX:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para convertir un entero de 5 dígitos en BCD,
;; a su representación binaria de 2 bytes.
;; (Sólo arroja resultados coherentes para números menores a 2^16 - 1)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Requisitos:
;; ___________
;;                  Unidades en R1.
;;                  Decenas en R2.
;;                  Centenas en R3.
;;                  Millares en R4.
;;                  Decenas de millares en R5.
;;                  Espacio de 5 bytes disponible en el stack.
;;
;; Registros afectados:
;; ____________________
;;                  Todos los registros del banco seleccionado.
;;                  R2 <- parte alta del entero.
;;                  R3 <- parte baja del entero.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se salvan las unidades:
    MOV     A, R1
    PUSH    Acc

    ; Se salvan las decenas:
    MOV     A, R2
    PUSH    Acc

    ; Se salvan las centenas:
    MOV     A, R3
    PUSH    Acc

    ; Se salvan los millares:
```

```
MOV     A, R4
PUSH    Acc

; Se pesan las decenas de millares:
MOV     A, R5
MOV     R6, #0d ; Parte alta.
MOV     R7, A
MOV     R4, #HIGH(10000d)
MOV     R5,  #LOW(10000d)
CALL    MUL16
; R2 <- Byte alto del producto.
; R3 <- Byte bajo del producto.

; Se recuperan los millares:
POP     Acc
MOV     R4, A

; Se salva el resultado intermedio en el stack:
MOV     A, R2
PUSH    Acc
MOV     A, R3
PUSH    Acc

; Se pesan los millares:
MOV     A, R4
MOV     R6, #0d ; Parte alta.
MOV     R7, A
MOV     R4, #HIGH(1000d)
MOV     R5,  #LOW(1000d)
CALL    MUL16
; R2 <- Byte alto del producto.
; R3 <- Byte bajo del producto.

; Se recupera la parte baja del resultado intermedio:
POP     Acc

; Se suma la parte baja del aporte de los millares:
ADD     A, R3
MOV     R3, A

; Se recupera la parte alta del resultado intermedio:
POP     Acc

; Se suma la parte alta del aporte de los millares:
ADDC    A, R2
MOV     R2, A

; Se recuperan las centenas y se las pesa:
POP     Acc
MOV     B, #100d
MUL     AB

; Se suma la parte baja del aporte de las centenas:
ADD     A, R3
MOV     R3, A

; Se suma la parte alta del aporte de las centenas:
MOV     A, B
ADDC    A, R2
MOV     R2, A

; Se recuperan las decenas:
POP     Acc
MOV     B, #10d
MUL     AB
```

```
    ; Se suma el aporte de las decenas
    ; a la parte baja del resultado intermedio:
    ADD     A, R3
    MOV     R3, A

    ; Se suma el carry a la parte alta del resultado intermedio:
    CLR     A
    ADDC    A, R2
    MOV     R2, A

    ; Se recuperan las unidades:
    POP     Acc
    ADD     A, R3
    MOV     R3, A

    ; Se suma el carry a la parte alta del resultado intermedio:
    CLR     A
    ADDC    A, R2
    MOV     R2, A

    ; R2 <- Byte alto del resultado.
    ; R3 <- Byte bajo del resultado.

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
HEX2ASCII:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para convertir un entero de 2 bytes representado en binario,
;; a sus 5 dígitos decimales en código ASCII.
;;
;; Construida sobre la base del algoritmo disponible en
;; http://www.8052.com/codelib/bcd.asm
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Requisitos:
;; ___________
;;              El byte alto en R1 y el byte bajo en R2.
;;
;; Registros afectados:
;; ____________________
;;                  A, B, C
;;                  R3 <- Decenas de millares.
;;                  R4 <- Millares.
;;                  R5 <- Centenas.
;;                  R6 <- Decenas.
;;                  R7 <- Unidades.
;;                  R1 <- 0
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se limpian los registros que se utilizarán para devolver el resultado:
    MOV     R3, #0d
    MOV     R4, #0d
    MOV     R5, #0d
    MOV     R6, #0d
    MOV     R7, #0d

    ; Se guardan las unidades en R7:
```

```
    MOV     B, #10d
    MOV     A, R2
    DIV     AB
    MOV     R7, B

    ; Se guardan las decenas en R6:
    MOV     B, #10d
    DIV     AB
    MOV     R6, B

    ; Se guardan las centenas en R5:
    MOV     R5, A

    ; Si no hay parte alta, no hay más nada que hacer:
    CJNE    R1, #0d, HEX2ASCII_HIGH
    SJMP    HEX2ASCII_END

    ; Hasta el momento se tienen las centenas (R5), las decenas (R6) y las
    ; unidades (R7) de la parte *baja del número completo*.
    ;
    ; Dado que R1 contiene la parte alta, el numero completo puede escribirse
    ; de la siguiente manera: R1*(256) + B, donde B=R5R6R7. Expresado de otra
    ; forma:

    ;        __R1
    ;        \
    ;        /_   256 + B = B +256 + 256 +256 +.... donde la cantidad de veces
    ;        n=1                                    que se suma 256 es
    ;                                               justamente R1.
    ; Entonces se suma R1 veces 256 más la parte baja ya convertida (B).
HEX2ASCII_HIGH:
    MOV     A, #6d
    ADD     A, R7
    MOV     B, #10d
    DIV     AB
    MOV     R7, B

    ADD     A, #5d
    ADD     A, R6
    MOV     B, #10d
    DIV     AB
    MOV     R6, B

    ADD     A, #2d
    ADD     A, R5
    MOV     B, #10d
    DIV     AB
    MOV     R5, B

    ; Se realiza una consideracion importante:
    ; Es posible que al sumar 256, el resultado no pueda ser representado por
    ; tres dígitos. Éste se almacena en R4:
    CJNE    R4, #0d, ($+3+2)
    SJMP    $+2+1

    ADD     A, R4

    MOV     R4, A
    DJNZ    R1, HEX2ASCII_HIGH


    ; Hasta el momento se tiene un numero que tiene la siguiente forma:
    ; N=R4R5R6R7 , donde R7 representa las unidades del número *completo*, R6
    ; las decenas y R5 las centenas. R4 representa los millares y las decenas
    ; de millares, codigicado en hexadecimal. Se realiza una nueva división
    ; por diez.
```

```
    MOV     B, #10d
    MOV     A, R4
    DIV     AB
    MOV     R4, B
    MOV     R3, A

HEX2ASCII_END:
    ; Finalmente, se suma el código ASCII del '0' a todos
    ; los dígitos BCD para convertirlos a ASCII:
    MOV     A, R3
    ADD     A, #'0'
    MOV     R3, A

    MOV     A, R4
    ADD     A, #'0'
    MOV     R4, A

    MOV     A, R5
    ADD     A, #'0'
    MOV     R5, A

    MOV     A, R6
    ADD     A, #'0'
    MOV     R6, A

    MOV     A, R7
    ADD     A, #'0'
    MOV     R7, A

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DIV16:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para dividir dos enteros de 2 bytes cada uno.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Requisitos:
;; ___________
;;              Dividendo: byte alto en R1; byte bajo en R0.
;;              Divisor:   byte alto en R3; byte bajo en R2.
;;
;; Registros afectados:
;; ____________________
;;                  A, B, C
;;                  Cociente: byte alto en R3; byte bajo en R2.
;;                  Resto:    byte alto en R1; byte bajo en R0.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Limpieza preliminar de registros necesarios:
    CLR     C
    MOV     R4, #0d
    MOV     R5, #0d
    MOV     B, #0d

    ; En el bucle siguiente, se rota hacia la izquierda al divisor
    ; hasta que alcance los 16 dígitos:
```

```
DIV16_ROTATE_DIVISOR_UNTIL_16_DIGIT:
    ; En B se cuenta la cantidad de bits rotados:
    INC     B

    ; Se rota primero la parte baja, incluye al Carry,
    ; para utilizarlo luego con la parte alta:
    MOV     A, R2
    RLC     A
    MOV     R2, A

    ; Se rota a la parte alta, junto al Carry que quedó
    ; de la rotación de la parte baja:
    MOV     A, R3
    RLC     A
    MOV     R3, A

    ; Cuando luego de rotar hacia la izquiera a la parte alta el carry
    ; se enciende, es porque el divisor ya alcanzó los 16 dígitos.
    ; En ese caso, se finaliza la rotación:
    JNC     DIV16_ROTATE_DIVISOR_UNTIL_16_DIGIT


    ; En el bucle siguiente, se rota hacia la derecha al nuevo divisor
    ; de 16 dígitos y en cada iteración se realiza una resta:
DIV16_ROTATE_DIVISOR_AND_SUBSTRACT:
    ; Se rota hacia la derecha a la parte alta, incluyendo al Carry,
    ; para utilizarlo luego con la parte baja:
    MOV     A, R3
    RRC     A
    MOV     R3, A

    ; Se rota hacia la derecha a la parte baja, junto al Carry que quedó
    ; de la rotación de la parte alta:
    MOV     A, R2
    RRC     A
    MOV     R2, A

    ; Se guarda una copia del dividendo en R7/R6:
    MOV     A, R1
    MOV     R7, A
    MOV     A, R0
    MOV     R6, A

    ; Eliminamos el Carry (Borrow) para la resta.
    CLR     C
    ; Se realiza la resta entre el dividendo (o lo que va quedando de él)
    ; y el divisor (rotado un paso en cada iteración):
    ; Parte baja:
    MOV     A, R0
    SUBB    A, R2
    MOV     R0, A
    ; Parte alta:
    MOV     A, R1
    SUBB    A, R3
    MOV     R1, A

    JNC     DIV16_C     ;No hay Borrow. Sigue más adelante.

    ; Hubo Borrow. Se recupera la copia del dividendo para deshacer
    ; la substracción:
    MOV     A, R7
    MOV     R1, A
    MOV     A, R6
    MOV     R0, A

DIV16_C:
```

```
    CPL     C
    MOV     A, R4
    RLC     A
    MOV     R4, A
    MOV     A, R5
    RLC     A
    MOV     R5, A

    ; Itera hasta que se rotaron hacia la derecha todos los bits
    ; del divisor que previamente se habían rotado hacia la izquierda:
    DJNZ    B, DIV16_ROTATE_DIVISOR_AND_SUBSTRACT

    ; Finalmente, se devuelve el resultado:
    ; Parte alta en R3:
    MOV     A, R5
    MOV     R3, A
    ; Parte baja en R2:
    MOV     A, R4
    MOV     R2, A

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MUL16:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rutina para multiplicar dos enteros de 2 bytes cada uno y obtener un
;; resultado de 24 bits.
;;
;; Tomada de http://www.8052.com/mul16.phtml
;;
;; Idea:
;; ————
;;
;;                Byte 4      Byte 3      Byte 2      Byte 1
;; Factor 1:                                  R6          R7
;; Factor 2:                                  R4          R5
;;               ——————————————————————————————————————————
;; Producto:          R0          R1          R2          R3
;;
;; 1. Se multiplica a R5 por R7, dejando el resultado de 16—bit en R2 y R3.
;; 2. Se multiplica a R5 por R6, sumando el resultado de 16—bit a R1 y a R2.
;; 3. Se multiplica a R4 por R7, sumando el resultado de 16—bit a R1 y a R2.
;; 4. Se multiplica a R4 por R6, sumando el resultado de 16—bit a R0 y a R1.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Requisitos:
;; ————————
;;              Factor 1: byte alto en R6; byte bajo en R7.
;;              Factor 2: byte alto en R4; byte bajo en R5.
;;
;; Registros afectados:
;; ———————————————
;;                  A, B, C
;;                  Producto: 32 bits ordenados en R0/R1/R2/R3.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; Se multiplica a R5 por R7:
```

```
    MOV     A, R5
    MOV     B, R7
    MUL     AB
    ; Se dejan el resultado de 16-bit en R2 y R3:
    ;Parte alta a R2:
    MOV     R2, B
    ;Parte baja a R3:
    MOV     R3, A


    ; Se multiplica a R5 por R6:
    MOV     A, R5
    MOV     B, R6
    MUL     AB
    ; Se suma el resultado de 16-bit a R1 y a R2:
    ; Parte baja a R2:
    ADD     A, R2
    MOV     R2, A
    ; Parte alta a R1:
    MOV     A, B
    ADDC    A, #0d
    MOV     R1, A
    ; Se suma el eventual carry a R0:
    CLR     A
    ADDC    A, #0d
    MOV     R0, A


    ; Se multiplica a R4 por R7:
    MOV     A, R4
    MOV     B, R7
    MUL     AB
    ; Se suma el resultado de 16-bit a R1 y a R2:
    ; Parte baja a R2:
    ADD     A, R2
    MOV     R2, A
    ; Parte alta a R1:
    MOV     A, B
    ADDC    A, R1
    MOV     R1, A
    ; Se suma el eventual carry a R0:
    CLR     A
    ADDC    A, R0
    MOV     R0, A


    ; Se multiplica a R4 por R6:
    MOV     A, R4
    MOV     B, R6
    MUL     AB
    ; Se suma el resultado de 16-bit a R0 y a R1:
    ; Parte baja a R1:
    ADD     A, R1
    MOV     R1, A
    ; Parte alta a R0:
    MOV     A, B
    ADDC    A, R0
    MOV     R0, A

    RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

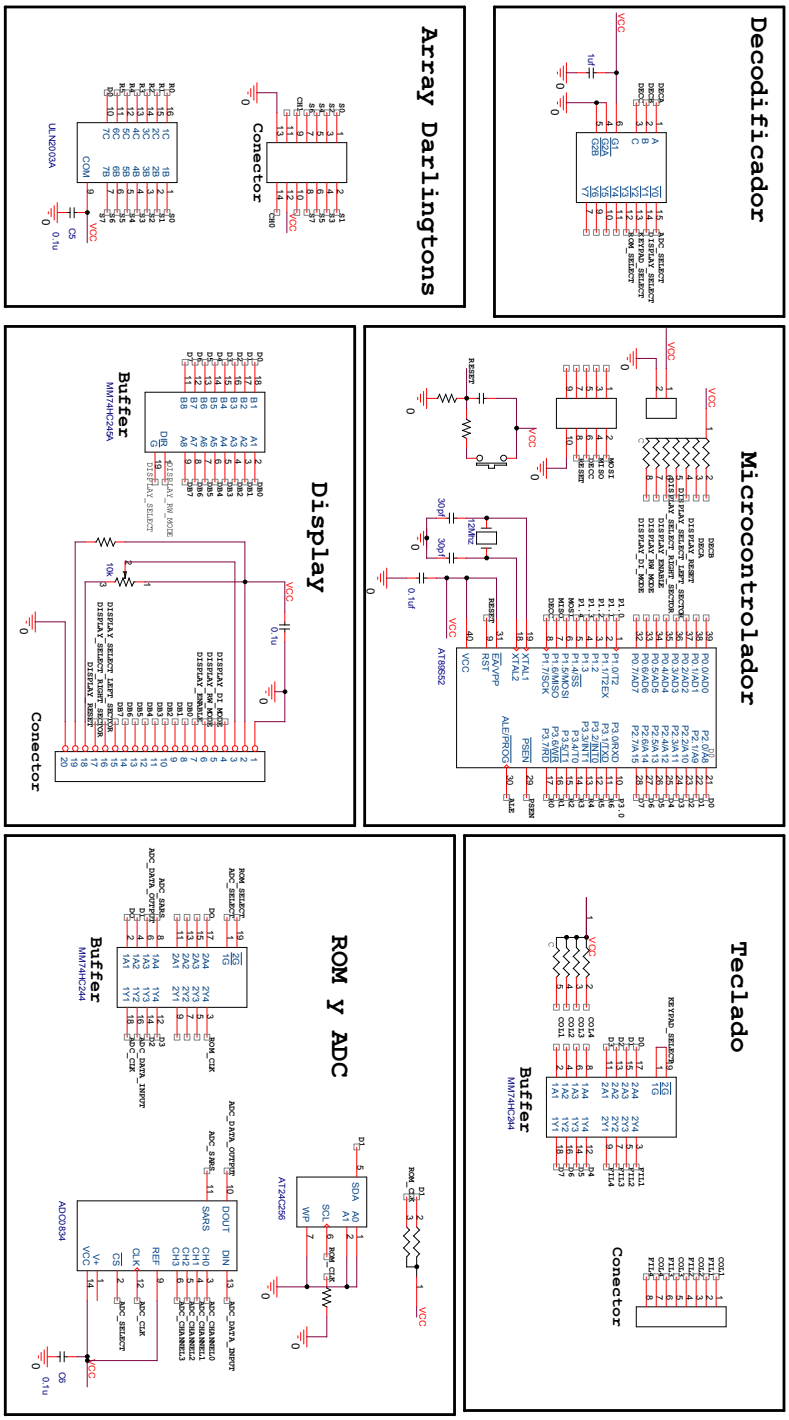
```
;; Rutinas de delay por software.
;;
;; En un AT89S52, cada CM equivale a 12 Ciclos de clock, o sea 1 CM equivale
;; a 1 microsegundo si el clock es de 12 MHz.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

DELAY_ONE_SECOND:
    ; Se necesitan ejecutar 1000000 CM para lograr 1 segundo.

                                    ; Ciclos de máquina de cada instrucción:
    MOV     R7, #130d               ; 1 CM * 1
DELAY_ONE_SECOND_A:
    MOV     R6, #15d                ; 1 CM * R7
DELAY_ONE_SECOND_B:
    MOV     R5, #255d               ; 1 CM * R6 * R7
    DJNZ    R5, $                   ; 2 CM * R5 * R6 * R7
    DJNZ    R6, DELAY_ONE_SECOND_B  ; 2 CM * R6 * R7
    NOP                             ; 1 CM * R7
    DJNZ    R7, DELAY_ONE_SECOND_A  ; 2 CM * R7
    RET                             ; 2 CM * 1
                                    ; Total: 3 + R7·(4 + R6·(3 + 2·R5))
                                    ; Con R5 = 255d, R6 = 15d y R7 = 130d,
                                    ; se logran 1000873 CM.
                                    ; Error menor al 0.1 %.

DELAY_ONE_MSEC:
    ; Se necesitan ejecutar 1000 CM para lograr 1 milisegundo.

                                    ; Ciclos de máquina de cada instrucción:
    MOV     R7, #165d               ; 1 CM * 1
    MOV     R6, #2d                 ; 1 CM * 1
DELAY_ONE_MSEC_LOOP:
    NOP                             ; 1 CM * R7 * R6
    DJNZ    R7, DELAY_ONE_MSEC_LOOP ; 2 CM * R7 * R6
    MOV     R7, #165d               ; 1 CM * R6
    DJNZ    R6, DELAY_ONE_MSEC_LOOP ; 2 CM * R6
    RET                             ; 2 CM * 1
                                    ; Total: 4 + R6·(3 + 3·R7)
                                    ; Con R7 = 165 y R6 = 2,
                                    ; se logran 1000 CM.


DELAY_TEN_MSEC:
    ; Se necesitan ejecutar 10000 CM para lograr 10 milisegundo.

                                    ; Ciclos de máquina de cada instrucción:
    MOV     R7, #166d               ; 1 CM * 1
    MOV     R6, #20d                 ; 1 CM * 1
DELAY_TEN_MSEC_LOOP:
    NOP                             ; 1 CM * R7 * R6
    DJNZ    R7, DELAY_TEN_MSEC_LOOP ; 2 CM * R7 * R6
    MOV     R7, #166d               ; 1 CM * R6
    DJNZ    R6, DELAY_TEN_MSEC_LOOP ; 2 CM * R6
    RET                             ; 2 CM * 1
                                    ; Total: 4 + R6·(3 + 3·R7)
                                    ; Con R7 = 166 y R6 = 20,
                                    ; se logran 10024 CM.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

6.15. macros.asm

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPLAY_MACRO_SELECT  MACRO
    MOV     DISPLAY_DATABUS, #00h
    MOV     decoder_selected_device, #DECODER_DISPLAY
    CALL    DECODER_SELECT_DEVICE
ENDM

KEYPAD_MACRO_SELECT  MACRO
    MOV     KEYPAD_DATABUS, #00h
    MOV     decoder_selected_device, #DECODER_KEYPAD
    CALL    DECODER_SELECT_DEVICE
ENDM

ROM_MACRO_SELECT  MACRO
    MOV     decoder_selected_device, #DECODER_ROM
    CALL    DECODER_SELECT_DEVICE
ENDM

ADC_MACRO_SELECT  MACRO
    MOV     decoder_selected_device, #DECODER_ADC
    CALL    DECODER_SELECT_DEVICE
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

START_SENSIBLE_OP MACRO
    CLR     EA
    NOP
ENDM

END_SENSIBLE_OP MACRO
    SETB    EA
    NOP
ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

MACRO_SELECT_BANK_0   MACRO
    CLR     RS0
    CLR     RS1
ENDM
MACRO_SELECT_BANK_1   MACRO
    SETB    RS0
    CLR     RS1
ENDM
MACRO_SELECT_BANK_2   MACRO
    CLR     RS0
    SETB    RS1
ENDM
MACRO_SELECT_BANK_3   MACRO
    SETB    RS0
    SETB    RS1
ENDM
```

7. Esquemáticos

7.1. Dispositivo



7.2. Circuito de prueba