DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Autor: Alberto J. Molina Cantero

Actualizado 26/5/2016

Programación del microcontrolador ATMega328P

1. Introducción y objetivos

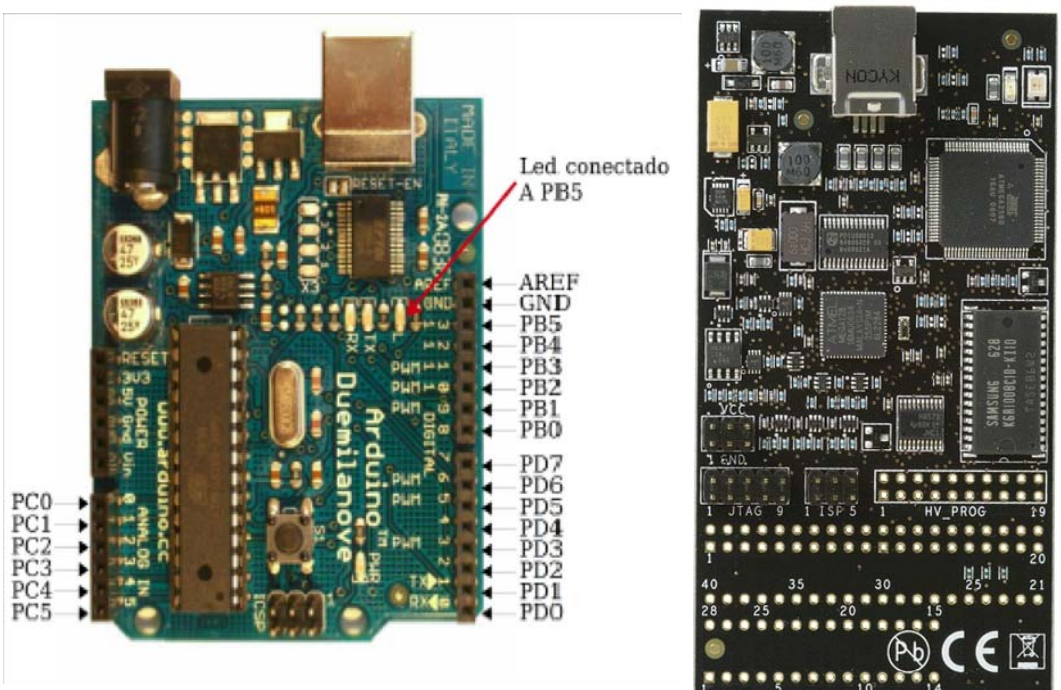
- Utilizar el entorno de programación y depuración de microcontroladores de ATMEL AVR-STUDIO que ya se presentó en la práctica anterior.

- Simular y depurar programas escritos en lenguaje ensamblador para el microcontrolador ATMEGA328P que usen puertos de E/S.

- Programar realmente el microcontrolador ATMEGA328P que se encuentra en una placa de desarrollo llamada Arduino1 Duemilanove (Figura 1a). Para ello se utiliza la plataforma de depuración/ programación AVR-DRAGON (Figura 1b) también del fabricante ATMEL.

Las placas Arduino están diseñadas para ser programadas en un lenguaje de programación propio, transfiriéndose los programas a través de su puerto USB. En esta sesión de laboratorio no se utilizarán estas características, es decir, se programarán directamente en ensamblador. Por ello, se han realizado modificaciones en dichas placas. Aunque no es relevante para esta sesión de laboratorio, se puede consultar toda la información adicional sobre esta placas en:
http://www.arduino.cc

AVR.STUDIO puede descargarse gratuitamente desde las páginas del fabricante de ATMEL en:
http://www.atmel.com



(a) **(b)**

Figura 1: (a) Placa de desarrollo Arduino. (b) Programador/depurador AVR-Dragon.

Durante la sesión de laboratorio se debe disponer de los ficheros indicados en la tabla 1.

Nombre del fichero	**Contenido**
contador_0_10.asm	Programa contador de 0 a 10. A completar por el alumno.
contador_0_1000.asm	Programa contador de 0 a 1000. A completar por el alumno.
conmutadores.asm	Programa para controlar dos pulsadores y dos leds. A completar por el alumno.

Tabla 1: Ficheros necesarios para realizar la práctica

2. Estudio teórico

Se van a utilizar tres programas (Tabla 1) en lenguaje ensamblador. Estos están disponibles en ficheros que se pueden encontrar en el directorio correspondiente a esta práctica en la web de la asignatura.

A continuación se presentan algunos detalles de los mismos:

1. Programa CONTADOR de 0 a 10:

Se trata de realizar un programa **contador_0_10.asm** en ensamblador que cuente de 0 a 10 utilizando un registro del microcontrolador. Cuando termine la cuenta el programa debe invertir el valor del pin PC0 y volver a empezar, es decir, volverá a contar de 0 a 10 e invertirá de nuevo el pin PC0. Así indefinidamente.

Para realizar el programa correctamente se debe configurar el puerto C como salida, para ello se propone comenzar el programa utilizando el siguiente fragmento de código (fichero contador_0_10.asm):

```
; Programa contador de 0 a 10
; Cada vez que se pase por 10 se debe invertir el PINC0

.include "m328pdef.inc"

.DEF  TMP=R19
      LDI R19,$FF
      OUT DDRC,TMP  ; Configura el puerto C completo como salida

                    ; A COMPLETAR
```

2. Programa CONTADOR de 0 a 1000:

Se trata de realizar un programa **contador_0_1000.asm** similar al anterior pero que cuente de 0 a 1000 antes de invertir el PC0. Ha de tenerse en cuenta que los registros del microcontrolador son de 8 bits y, por tanto, la cuenta máxima que se puede realizar con un único registro es de 0 a 255.

3. Programa para controlar los CONMUTADORES:

Se trata de realizar un programa **conmutadores.asm** que permita manejar los puertos de entrada/salida para poder activar ciertos leds cuando se pulsa un conmutador.

En el esquema de la Figura 2 aparecen dos conmutadores (PC4 y PC5 - línea discontinua) y dos leds (PC2 y PC3 - línea continua) que deben operar de la siguiente forma:

- si se pulsa el conmutador conectado a PC5 debe encenderse el led conectado a PC3 y mantenerse encendido hasta que se vuelva a pulsar el conmutador PC5.

- si se pulsa el conmutador conectado a PC4 debe encenderse el led conectado a PC2 y mantenerse encendido hasta que se vuelva a pulsar el conmutador PC4.

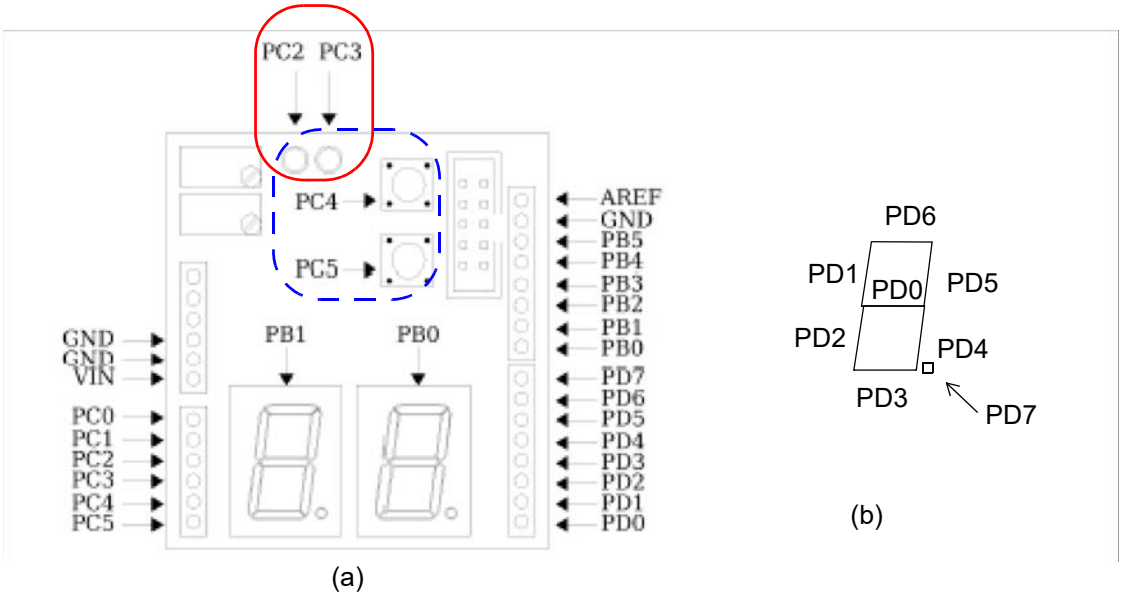


Figura 2: (a) Placa de expansión E/S para Arduino. (b) Detalle de conexión de los segmentos del display a los pines de E/S.

La Tabla 2 muestra los puertos y los bits asociados a los componentes así como la configuración necesaria para que operen correctamente. Téngalo en cuenta al realizar el programa.

Pin	Puerto	Componente	Configuración	Funcionamiento
PC2	C	Led	salida (DDRC2 = 1)	PINC2=0 apagado PINC2=1 encendido
PC3			salida (DDRC3 = 1)	PINC3=0 apagado PINC3=1 encendido
PC4		Conmutador	entrada (DDRC4 = 0)	**PINC4=1 no pulsado** PINC4=0 pulsado
PC5			entrada (DDRC5 = 0)	**PINC5=1 no pulsado** PINC5=0 pulsado

Tabla 2: Configuración del puerto de entrada/salida PORTC y funcionamiento de los leds y conmutadores.

3. Estudio experimental

El estudio experimental consistirá en **completar** los programas indicados en la Tabla 1 y **comprobar** su funcionamiento, así como contestar algunas cuestiones que se plantearán.

En primer lugar (apartado 3.1) usaremos el simulador para completar y probar los programas contador_0_10.asm y contador 0_1000.asm. A continuación (apartado 3.2), programaremos el microcontrolador con dichos programas y observaremos su funcionamiento sobre elementos de la placa, para lo que utilizaremos el osciloscopio y uno de los diodos. Finalmente, programaremos el microcontrolador con el programa conmutadores.asm para controlar los conmutadores y leds señalados en la Figura 2a.

En la guía de uso del programa AVR-Studio que encontrará en la web junto con este enunciado, puede recordar aspectos esenciales como son la creación de proyectos o la compilación y depuración de programas. No obstante, algunos pasos también se detallan a continuación.

3.1 Ejecución en el simulador

Para comprobar el funcionamiento de los programas de esta práctica **se debe realizar la ejecución paso a paso observando cómo cambian los valores de los registros y el PC0. Hay que desplegar los registros en la ventana del procesador y el puerto C en la ventana de E/S para visualizar los registros del puerto.**

1. Complete el programa **contador_0_10.asm**.

2. Compruebe que su programa opera correctamente. Para ello inicie la simulación con Start Debugging y ejecute paso a paso el programa. Puede utilizar la tecla F10 para no tener que utilizar los menús. Recuerde: para comprobar el funcionamiento del programa le convendrá desplegar el puerto C en la ventana de E/S.

3. Complete el programa **contador_0_1000.asm** y realice las siguientes tareas:

a. Simule el programa y obtenga el número de ciclos que tarda el su programa en conmutar de valor el pin PC0.

b. Dado que el PC0 está conmutando su valor continuamente podemos asegurar que se está generando una señal cuadrada. Conociendo el número de ciclos que tarda su programa en

conmutar el PC0 y sabiendo que el procesador funciona a una frecuencia de 1Mhz se puede calcular la frecuencia del señal cuadrada que se genera. Calcule dicha frecuencia.

c. Calcule el valor al que tendría que llegar la cuenta del programa para que el PC0 se encienda 5 veces por segundo trabajando a la misma frecuencia indicada anteriormente (1Mhz).

4. Utilice la opción AutoStep (icono ) y compruebe la ejecución animada que realiza el simulador del programa.

3.2 Programación del microcontrolador

El siguiente paso consiste en la programación con el programador AVR-DRAGON (Figura 1b) de un microcontrolador ATMEGA328P en una placa Arduino (Figura 1a). El entorno de pruebas utilizado en esta sesión de laboratorio está formada por tres componentes: programador AVR-DRAGON, placa de prototipo Arduino Duemilanove y placa de expansión para Arduino con componentes E/S.

La placa de expansión mostrada en la Figura 2a está conectada a la placa Arduino, quedando ocultos todos los componentes del Arduino. En la placa de expansión están disponibles todos los puertos del microcontrolador en los laterales de la placa, además, estos puertos también están conectados a diversos componentes como son, leds, displays, conmutadores, etc. Estos componentes se utilizarán posteriormente para realizar programas que controlen la entrada/salida.

3.2.1 Conexión de la placa y configuración para la programación

En primer lugar se deben conectar ambas placas a los conectores USB. No es necesaria ninguna alimentación adicional ya que utilizan los 5V suministrados por el bus USB. Tras la conexión USB puede aparecer en el ordenador algún cuadro de diálogo indicando que se ha encontrado nuevo hardware. Si esto ocurriera, debe instalar los controladores, no cancele la instalación o tendrá problemas de programación del microcontrolador.

La placa AVR-DRAGON dispone de dos leds, inicialmente se iluminan uno en verde y otro en rojo. El led de color rojo cambiará de color indicando el estado de la comunicación con AVR-STUDIO. La Tabla 3 muestra el significado de los diferentes colores de dicho led, debemos observarlo durante los siguientes pasos para detectar posibles problemas en la programación del microcontrolador.

Color	**Descripción**
Rojo	En reposo, no hay conexión con AVR Studio
Apagado	En reposo, conectado a AVR Studio
Verde	Transfiriendo datos
Amarillo	Inicialización o actualización del firmware

Tabla 3: Indicaciones del led de AVR-DRAGON.

Antes de realizar la programación se debe verificar la correcta configuración de AVR-STUDIO realizando una prueba de conexión con el microcontrolador. Accediendo al menú Tools hay que usar el submenú Program AVR y la opción Connect. Aparecerá el diálogo mostrado en la Figura 3

Alternativamente, dicho diálogo se puede obtener de manera directa utilizando el botón de la barra de herramientas.

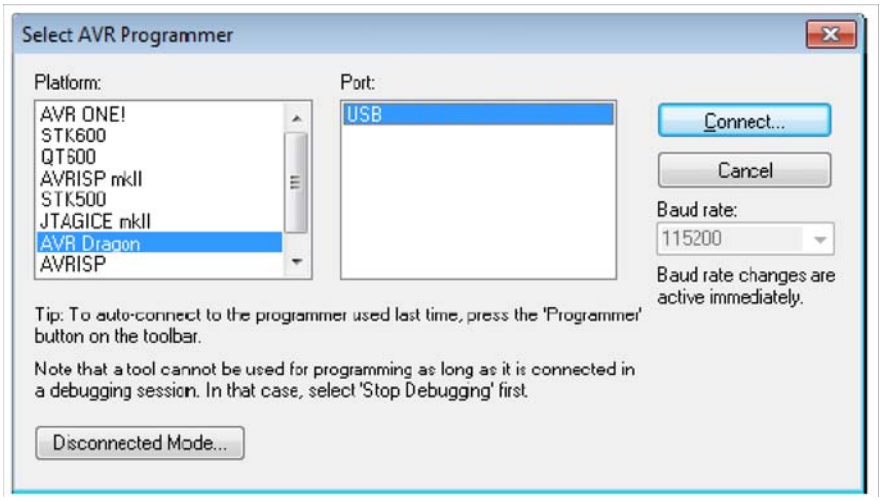


Figura 3: Selección del programador y el puerto

En este diálogo hay que establecer la configuración indicada en la Figura 3: plataforma AVR-DRAGON y puerto USB. Tras pulsar el botón Connect, si la conexión es correcta, debe aparecer automáticamente el diálogo mostrado en la Figura 4 y el led rojo de AVR-DRAGON se apagará.

En caso de no aparecer automáticamente el diálogo de la Figura 4 se puede utilizar el botón de la barra de herramientas o, la opción de menú Tools submenú Program AVR. Tras esto finalmente aparecerá la ventana mostrada en la Figura 4.

De las múltiples pestañas que contiene sólo utilizaremos la primera y la segunda: Main y Program. En primer lugar se realizará una prueba de comunicación siguiendo estos pasos:

- Seleccionar la pestaña Main.
- Seleccionar el microcontrolador correcto del cuadro desplegable indicado con Device and Signature Bytes. En estas placas disponemos del microcontrolador ATMega328P.
- Pulsar el botón Read Signature. El programa debe responder con el texto "Signature matches selected device". Si respondiera con un error, se debe volver a desplegar el cuadro selector de microcontrolador, seleccionar el correcto, y volver a realizar el test de comunicación.

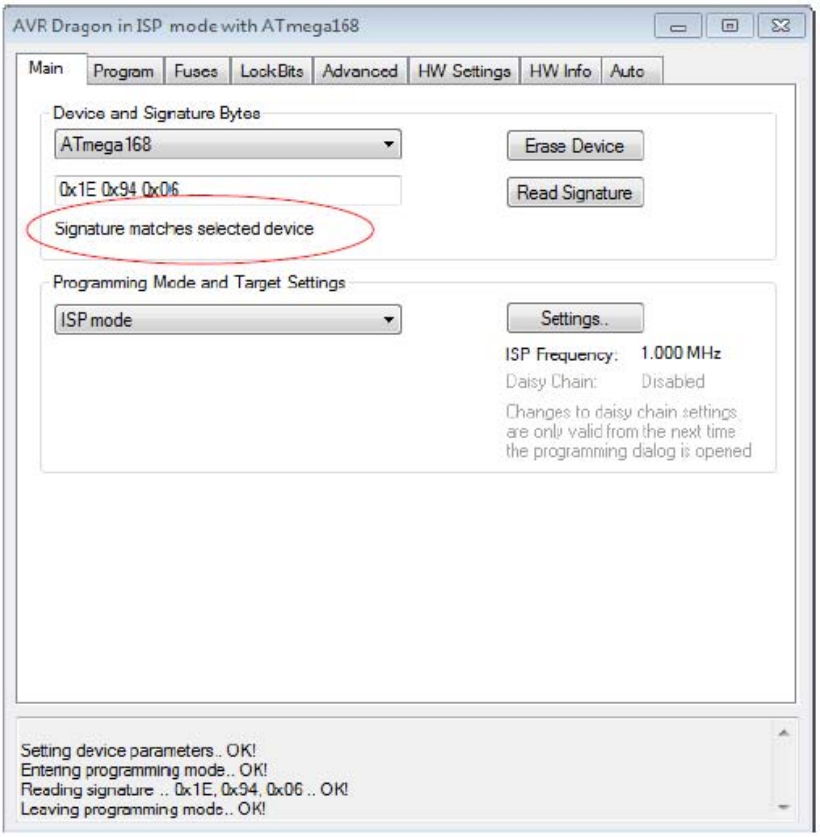


Figura 4: Pestaña principal de la ventana de programación del microcontrolador.

3.2.2 Programación del código ensamblado

El siguiente paso consiste en realizar la programación del microcontrolador con el código que se ha ensamblado, para ello, seleccione la pestaña Program. Si el ensamblado se realizó con éxito habrá generado un fichero con extensión .hex dentro del directorio del proyecto y con el nombre del proyecto. En la Figura 5 se muestra el diálogo de programación donde hay que seleccionar el fichero .hex. Este diálogo tiene tres cuadros donde se puede seleccionar un fichero: Flash, EEPROM y ELF. Hay que utilizar la sección Flash y el botón de selección de fichero (indicado con la flecha roja en la Figura 5). Tras esto basta con pulsar el botón Program para realizar la programación del microcontrolador.

Una vez realizada la programación hay que comprobar si el programa se está ejecutando correctamente. Para ello, debe conectar un canal del osciloscopio en el PC0 y comprobar si se visualiza una señal cuadrada entre 0 y 5 voltios (no olvide conectar la tierra del osciloscopio). Utilice el esquema de la Figura 2 para realizar las conexiones y realice las siguientes tareas:

1. Tras los pasos anteriores el microcontrolador se habrá programado con el programa **contador_0_1000.asm**, compruebe con el osciloscopio la frecuencia de la señal cuadrada generada en el PC0.

2. Modifique el programa para que la frecuencia de dicha señal sea aproximadamente 5Hz. Programe el microcontrolador de nuevo y compruebe la nueva frecuencia en el osciloscopio.

3. Modifique el programa de forma que el pin que conmute sea el PC2, de esta forma debe observar que el LED correspondiente se enciende 5 veces por segundo.

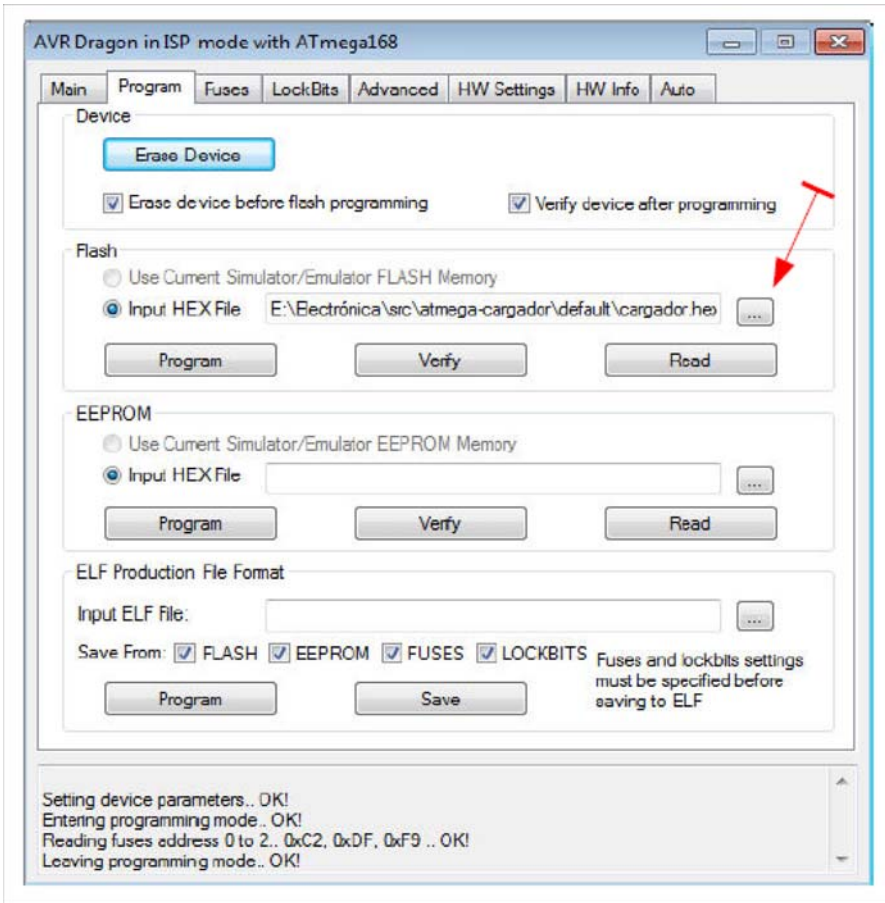


Figura 5: Pestaña de programación de la ventana de programación del microcontrolador.

3.3 Realización de un programa de control de conmutadores y leds

En este apartado se ha de trabajar con el programa **conmutadores.asm** ya presentado en el estudio teórico y que controla a través de los puertos de entrada/salida la activación de diodos leds cuando se pulsa un conmutador.

Las tareas a realizar son las siguientes:

1. Cree un nuevo proyecto utilizando el código suministrado en el fichero **conmutadores.asm** y complete el programa.

2. Utilice el simulador para comprobar que funciona correctamente. Debe conmutar manualmente los pines PC4 y PC5 desde el simulador. Esto se consigue desplegando el puerto C en árbol de dispositivos que muestra el AVR-STUDIO en la parte derecha durante la simulación y pulsando el botón del ratón sobre el cuadro que representa el bit correspondiente. Cuando el cuadro está relleno de color negro significara que el bit está a 1, si está en blanco es 0.

3. Una vez comprobado en el simulador el correcto funcionamiento, repita los pasos realizados en la sección 3.2.2 para programar el microcontrolador con este nuevo programa. Compruebe que funciona correctamente pulsando los conmutadores.

4. Apartado opcional

El nuevo programa a completar debe contar el número de pulsaciones de un conmutador y mostrarlo en el display 7 segmentos. En concreto, se trata de contar el número de pulsaciones (de 0 a 9) que se producen en el conmutador conectado a PC4.

La Tabla 4 muestra la información de los componentes de entrada/salida que se usarán. Se incluyen los puertos, los bits asociados a los componentes y la configuración necesaria para que operen correctamente.

Programa contador de PULSACIONES:

Se trata de realizar un programa **contador_BCD.asm** que permita contar el número de pulsaciones de un conmutador y mostrar dicho número en un display 7-segmentos. El programa constará de las siguientes partes:

- programa principal desde donde se llama a la subrutina inicializa_puertos e inicializa_tabla7seg, detecta si el conmutador se pulsa y cuenta el número de pulsaciones en código BCD (este código debe ser el de cuenta para controlar adecuadamente el display 7-segmentos),

- subrutina **inicializa_puertos**, utilizada para configurar correctamente los puertos (véase Tabla 4), debe completarla,

- subrutina **inicializa_tabla7seg**, para la creación en memoria de una tabla con el código 7 segmentos, debe completarla.

Pin	Puerto	Componente	Configuración	Funcionamiento
PC4	C	Conmutador	entrada (DDRC4 = 0)	**PINC4=1 no pulsado** PINC4=0 pulsado
PD0,PD1,...,PD7	D	Segmentos de los displays	salida (DDRD = 0xFF)	PORTDX=0 apagado PORTDX=1 encendido
PB0	B	Habilitador de display 0	salida (DDRB0 = 1)	PORTB0=1 apagado PORTB0=0 encendido
PB1		Habilitador del display 1	salida (DDRB1 = 1)	PORTB1=1 apagado PORTB1=0 encendido

Tabla 4: Configuración de los puertos de entrada/salida conectados con los displays 7 segmentos.

Mejoras en el programa contador de pulsaciones

También se propone hacer algunas mejoras en el programa contador: utilizar los dos conmutadores para incrementar/decrementar y realizar un control de rebote del conmutador

Se propone realizar las siguientes tareas (debe partir del programa realizado en la sección anterior):

1. Modifique el bucle principal del programa para que se comprueben las pulsaciones de los dos conmutadores. Si se pulsa el primero, el contador debe incrementarse, si se pulsa el segundo, el contador debe decrementarse. No olvide comprobar antes del decremento si la cuenta es cero para establecerla a 9, si no el decremento fallará.

2. Para realizar el control de rebotes se propone paralizar la ejecución del programa entre 5 y 10 ms. Para ello, realice las siguientes modificaciones en el programa:

2.1. Cree una subrutina llamada no_rebote que consista en un bucle que espere un tiempo determinado. Sabiendo que el procesador funciona a 1 MHz utilice un contador que mantenga un

bucle contando durante unos 10 ms aproximadamente. Esta subrutina es similar al programa realizado en el estudio teórico.

2.2. Modifique el programa principal incluyendo una llamada a esta nueva subrutina cuando detecte que el conmutador se ha pulsado. Además, debe llamar de nuevo a esta subrutina cuando detecte que el conmutador se ha soltado, ya que los rebotes pueden ocurrir tanto al pulsar como al soltar el conmutador.

2.3. Pruebe su nuevo programa en la placa de desarrollo.