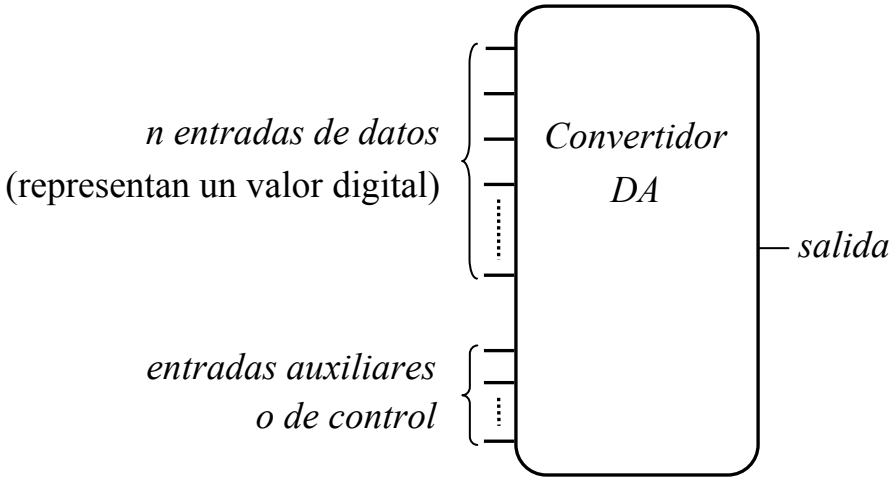
Digital to analog converter (*D/A converter* ≡ *DAC*)

- Un convertidor D/A de *n* bits es un circuito que en su concepción más básica consta de *n* *entradas de datos*, varias *entradas de control* y 1 *salida* analógica.

Las *n entradas de datos* representan en conjunto un valor digital a partir del cual, según el convertidor D/A que se considere, en su salida se genera una *tensión* o bien en una *corriente* constante (en estas notas se va a considerar que se genera una tensión).

Las *entradas auxiliares* o de *control* sirven, entre otras cosas, para indicarle al convertidor D/A que realice la conversión del valor digital de entrada en una tensión, así como para especificar las tensiones de referencia.



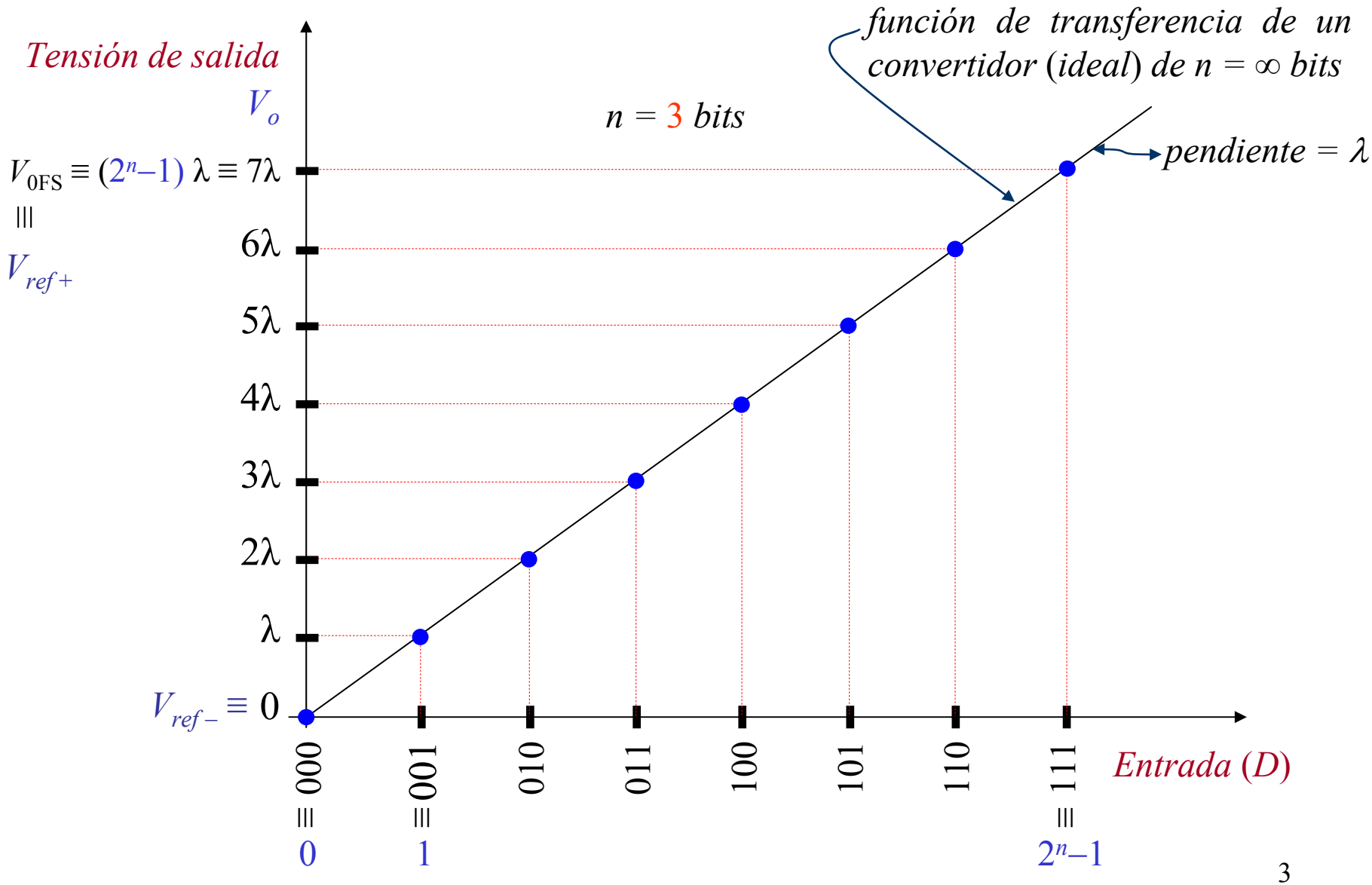
· Como se verá más adelante, el valor de la tensión de salida depende de unas *tensiones de referencia* y del valor digital representado por las n entradas binarias. Algunos convertidores D/A utilizan una tensión de referencia externa (≡ que hay que proporcionarle), mientras que otros utilizan una tensión de referencia interna que obtienen a partir de la tensión de alimentación.

Nota: los MDAC (≡ *multiplying digital to analog converters*) se caracterizan porque operan con una tensión de referencia externa, la cual puede superar ampliamente la tensión de alimentación del D/A.

Nota: dado que el dato a convertir se representa mediante n bits, este sólo puede tomar 2^n valores distintos y, por lo tanto, en la salida de un convertidor D/A sólo puede haber 2^n tensiones distintas (ver siguiente diapositiva).

En la siguiente diapositiva se representa la relación existente entre los 2^n valores que pueden representar los n terminales de datos y las correspondientes tensiones de salida, en el caso particular de un convertidor D/A ideal de $n = 3$ bits.

Función de transferencia de un convertidor D/A ideal de n = 3 bits



· Se define la *resolución* (λ) de un convertidor D/A como la variación que experimenta la tensión de salida ante el cambio de valor del bit menos significativo de las entradas de datos. Si consideramos que el mayor valor que puede tomar la tensión de salida es V_{ref+} y que el menor valor es V_{ref-}, entonces la resolución (λ) de un convertidor de *n* bits cumple lo siguiente (ver diapositiva anterior):

$$\frac{V_{ref+} - V_{ref-}}{2^n - 1 - 0} = \frac{\lambda - 0}{1 - 0} \quad \rightarrow \quad \lambda = \frac{V_{ref+} - V_{ref-}}{2^n - 1} \simeq \frac{V_{ref+} - V_{ref-}}{2^n}$$

aproximar $2^n - 1$ por 2^n tiene la ventaja de que la división por 2^n se puede realizar de forma muchísimo más rápida que la división por $2^n - 1$, siendo despreciable el error cometido si *n* es lo suficientemente grande.

Caso particular: si consideramos que el mayor valor que puede tomar la tensión de salida es $V_{ref+} \equiv V_{OFS}$ y que el menor valor es $V_{ref-} = 0$, entonces la resolución (λ) de un convertidor de *n* bits cumple lo siguiente:

$$\lambda = \frac{V_{ref+} - V_{ref-}}{2^n - 1} = \frac{V_{OFS} - 0}{2^n - 1} \simeq \frac{V_{OFS}}{2^n}$$

· La relación entre la tensión de salida (V_0) y el valor digital (D) que representan las n entradas binarias, cumple lo siguiente:

$$\frac{V_{ref+} - V_{ref-}}{2^n - 1} = \frac{V_0 - V_{ref-}}{D - 0} \quad \rightarrow \quad V_0 = \left[\frac{V_{ref+} - V_{ref-}}{2^n - 1}\right] D_{10} + V_{ref-}$$

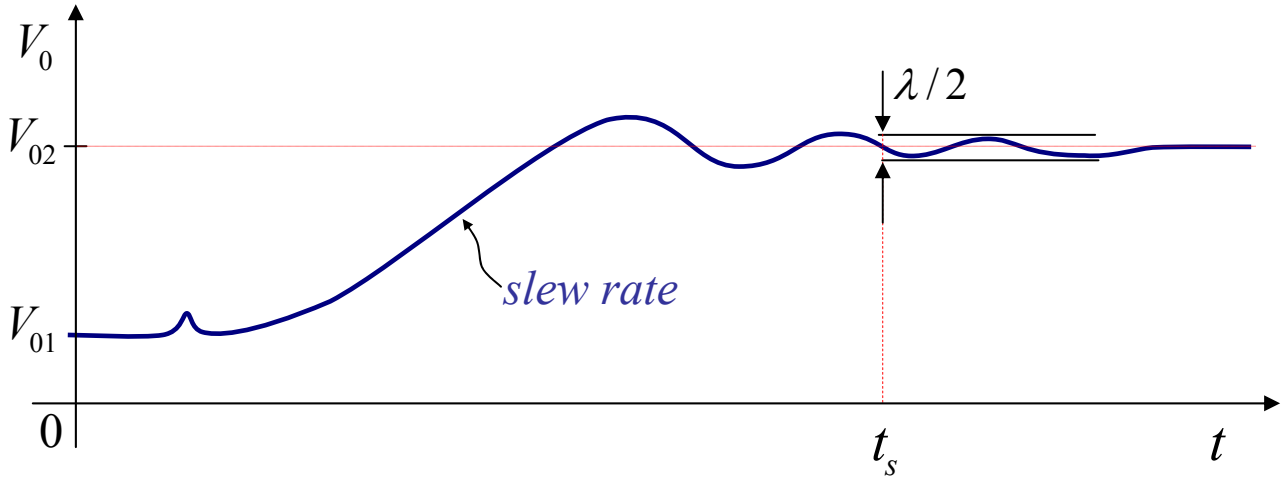
es decir, se cumple que:

$$V_0 = \lambda \cdot D_{10} + V_{ref-} \qquad \text{siendo:} \qquad \lambda = \frac{V_{ref+} - V_{ref-}}{2^n - 1} \simeq \frac{V_{ref+} - V_{ref-}}{2^n}$$

· Si se cumple que $V_{ref+} = V_{0FS}$ y $V_{ref-} = 0$, la relación entre la tensión de salida (V_0) y el valor digital (D) que representan las n entradas binarias, cumple lo siguiente:

$$V_0 = \left[\frac{V_{ref+} - V_{ref-}}{2^n - 1}\right] D_{10} + V_{ref-} = \left[\frac{V_{OFS} - 0}{2^n - 1}\right] D_{10} + 0 \simeq \frac{V_{OFS} \cdot D_{10}}{2^n}$$

· Se define el *tiempo de establecimiento* ($t_s \equiv$ *settling time*) como el tiempo que transcurre desde que el convertidor inicia la conversión de un nuevo valor (D) hasta que la tensión de salida (Vo) alcanza el valor correspondiente a dicho valor, con un margen de error dado (típicamente $\frac{1}{2}LSB = \lambda/2$)



· Se define la frecuencia de conversión (*conversion rate*) como la máxima frecuencia con la que puede realizar una conversión, de modo que la salida alcance siempre el valor correspondiente al valor de entrada a convertir (D). Los fabricantes suelen indicar una frecuencia de conversión inferior a $1/t_s$ (siendo t_s el *tiempo de establecimiento*) y su valor se suele expresar en *Hz* o en *samples/second*

Ejemplo: un convertidor *DA* de $n = 8$ bits tiene una tensión de referencia de 2,55V, ¿cuál es el valor de su *resolución*?

$$\lambda = \frac{V_{OFS}}{2^n - 1} = \frac{2,55}{2^8 - 1} = 10^{-2} = 10\ mV \simeq \frac{V_{OFS}}{2^n} = \frac{2,55}{2^8} = 0.996 \cdot 10^{-2} = 9.96 mV$$

Ejemplo: ¿Cuál es la tensión de salida de un DAC de $n = 12$ bits, correspondiente a un valor de entrada $D = 1101\ 1010\ 0011_2 = 3491$ si $V_{OFS} = 15v$?

$$V_0 = \lambda \cdot D_{10} = \left[\frac{V_{OFS}}{2^n - 1}\right] D_{10} = \left[\frac{15}{2^{12} - 1}\right] 3491 = 12,787V \simeq \left[\frac{15}{2^{12}}\right] 3491 = 12,784V$$

Ejemplo: Un DAC de 8 bits tiene una resolución de 10*m*V. Se pide:

i) V_{OFS}

ii) V_o para $D = 10000000_2 = 128$

Solución:

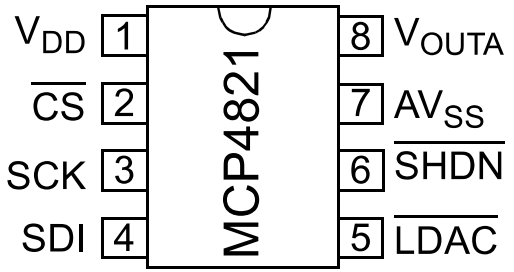
$$i)\ \lambda = \left[\frac{V_{OFS}}{2^n - 1}\right] \quad \Rightarrow \quad V_{OFS} = \lambda \cdot (2^n - 1) = 10^{-2} \cdot (2^8 - 1) = 2{,}55\text{v}$$

$$ii)\ V_o = \lambda \cdot D = 10^{-2} \cdot 128 = 1{,}28\text{v}$$

▪ Hay convertidores D/A en los que tanto las señales de control como los bits que representan el dato a convertir hay que enviárselos en serie, siguiendo un protocolo dado como el bus SPI, bus $I^2C \equiv$ IIC, bus Microwire, etc. A continuación se detallan las características más importantes de un convertidor D/A que recibe los bits de datos y de control en serie, siguiendo el protocolo SPI (*spy*).

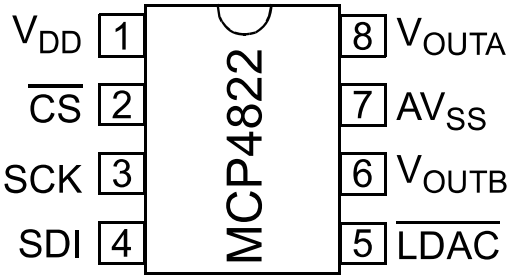
Notas DACs MCP4821-MCP4822

8-Pin PDIP, SOIC, MSOP



- El MCP4821 tiene 1 convertidor DAC de 12 bits
- El MCP4822 tiene 2 convertidores DAC de 12 bits
- La comunicación con el DAC se realiza en serie (bus SPI)
- $t_{settling\ time} = 4{,}5\mu seg$
- $V_{DD} \in [2.7V, 5.5V]$
- $AV_{SS} \equiv GND$
- $V_{REF+} = 2'048V$ (interna y constante) $[V_{REF-} = 0]$

8-Pin PDIP, SOIC, MSOP



- $v_{out} = \frac{2'048 \cdot Gain \cdot D}{2^{12}}$ (siendo D el dato a convertir ≡ representa el valor de una muestra)
- $\overline{CS}$: se tiene que poner a 0 para establecer una comunicación (por SPI) con el DAC

REGISTER 5-1: WRITE COMMAND REGISTER

Upper Half:							
W-x	W-x	W-x	W-0	W-x	W-x	W-x	W-x
$\overline{A}$/B	—	$\overline{GA}$	$\overline{SHDN}$	D11	D10	D9	D8
bit 15							bit 8

Lower Half:							
W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
D7	D6	D5	D4	D3	D2	D1	D0
bit 7							bit 0

- $\overline{A}/B = 0$ el dato enviado es para el convertidor DAC_A

  $\overline{A}/B = 1$ el dato enviado es para el convertidor DAC_B

- $\overline{GA} = 1$ (Gain = 1) $\rightarrow$ $v_{out} = \frac{2'048 \cdot D}{2^{12}}$

  $\overline{GA} = 0$ (Gain = 2) $\rightarrow$ $v_{out} = \frac{2'048 \cdot 2 \cdot D}{2^{12}}$

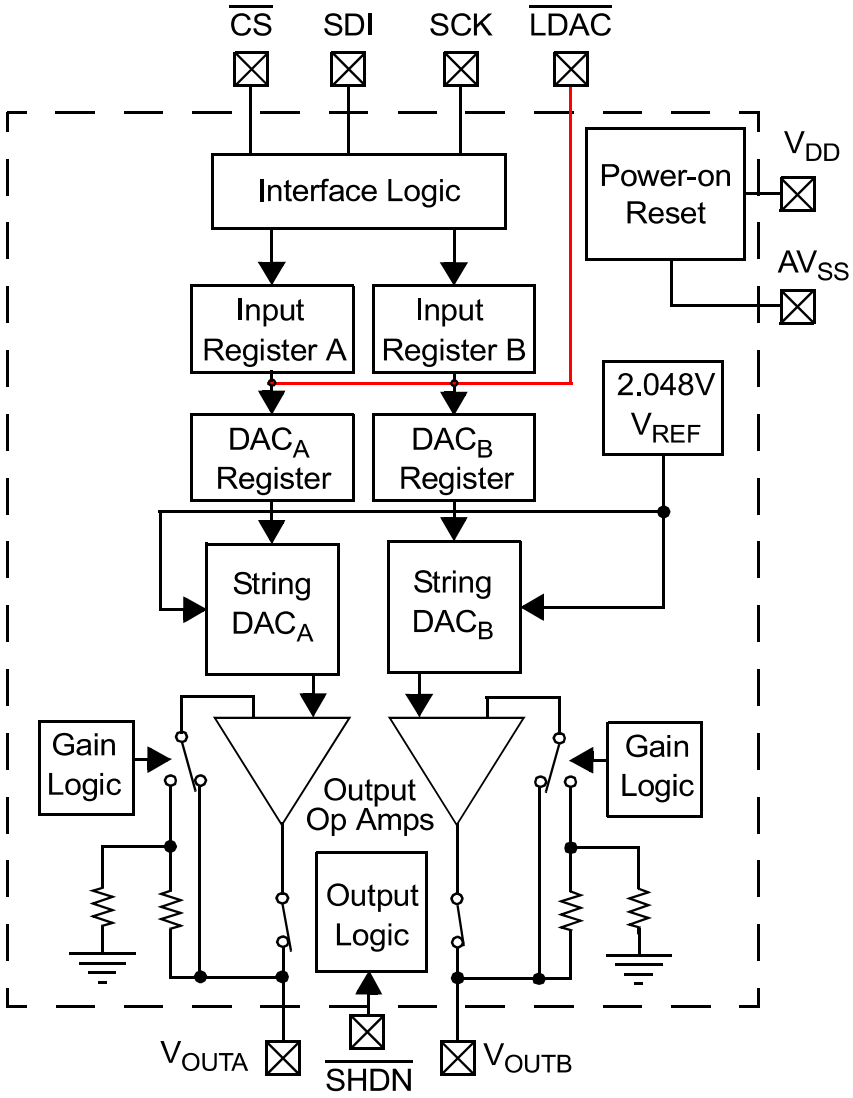
- $\overline{SHDN} = 1$ $\rightarrow$ el convertidor funciona

  $\overline{SHDN} = 0$ $\rightarrow$ la salida se pone en *tercer estado* (en un estado de alta impedancia) ≡ el convertidor no funciona !!!.

- $D_{11} - D_0 \in [0, 4095]$: valor a convertir en una tensión

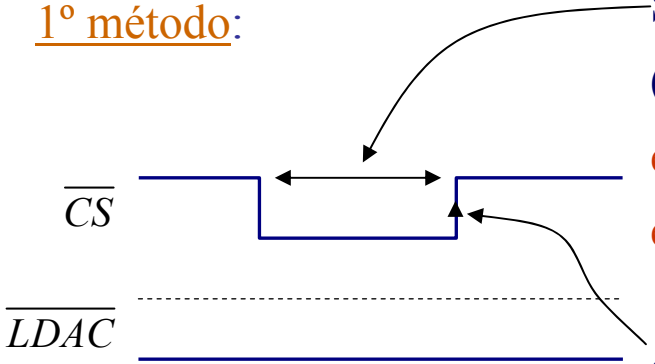
- $\overline{\text{LDAC}}$: cuando este pin se pone a 0, el contenido de los registros de entrada A y B se transfiere simultáneamente a los registros de salida DAC_A y DAC_B. Esto hace que ambas conversiones se inicien a la vez y que, como consecuencia de ello, las dos salidas actualicen sus respectivos valores al mismo tiempo.

Si se deja el pin $\overline{\text{LDAC}}$ siempre a 0, el contenido de los registros de entrada se transfiere a los registros de salida durante el flanco de subida del pin $\overline{\text{CS}}$.



- Escritura en 1 DAC

1º método:

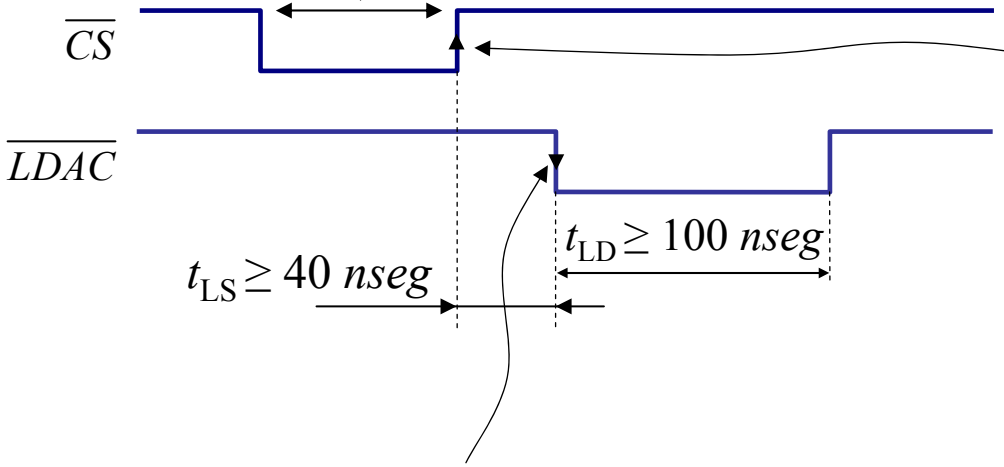


Se envía el comando de escritura, de 16 bits, al DAC (primero se envía el byte alto y a continuación se envía el byte bajo (*big endian*) ⇒ se realizan 2 operaciones de escritura consecutivas)

Se guarda el dato enviado (12 bits) en el correspondiente registro de entrada. Como el pin $\overline{\text{LDAC}}$ está a nivel bajo, a continuación se transfiere el dato guardado en el registro de entrada al registro de salida DACx y transcurrido un tiempo $t_{settling} \approx 4{,}5\mu$seg, la salida toma la tensión correspondiente al dato enviado

2º método:

Se envía el comando de escritura (de 16 bits) al DAC (primero se envía el byte alto y a continuación se envía el byte bajo ⇒ 2 operaciones de escritura consecutivas)



El dato enviado se guarda en el correspondiente registro de entrada.

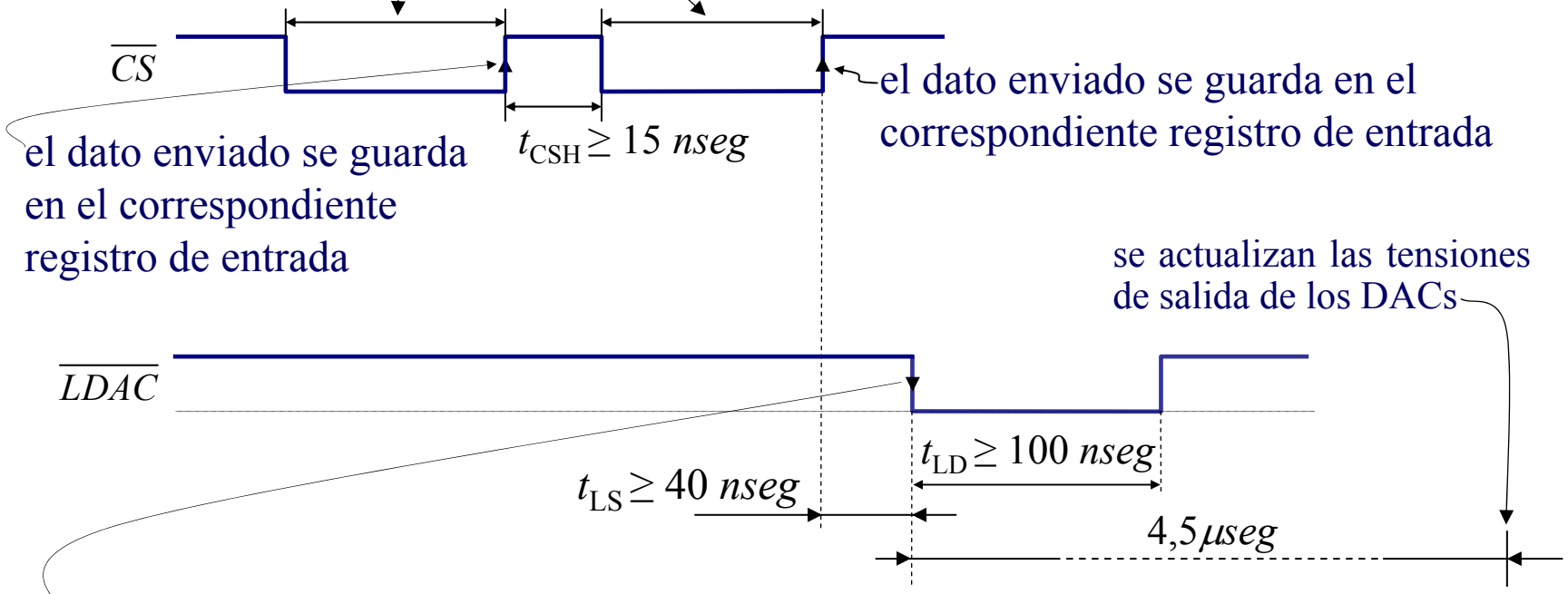
se transfiere el dato guardado en el registro de entrada al correspondiente registro de salida, iniciándose a continuación la conversión del dato transferido en una tensión.

no

- Escritura en los 2 DAC:

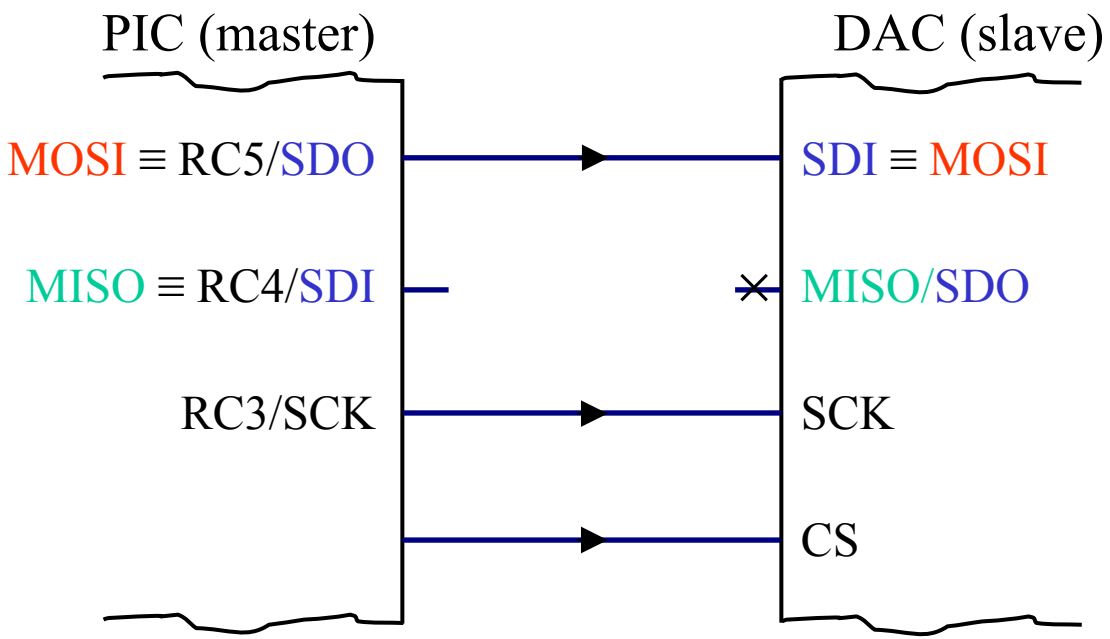
Se envía el comando de escritura, de 16 bits, a uno de los DAC (primero se envía el byte alto y a continuación se envía el byte bajo ⇒ 2 operaciones de escritura consecutivas)

Se envía el comando de escritura (16 bits) al otro DAC



se transfiere el contenido de los registros de entrada a los correspondientes registros de salida (DAC_A y DAC_B) y a continuación se inicia la conversión de los respectivos datos en tensiones.

- Conexiones al bus SPI



Nota: el DAC no tiene terminal SDO/MISO porque no envía datos, sólo los recibe.

Nota: *Big endian* indica que en primer lugar se envía el byte más significativo y que a continuación se envía el byte menos significativo.

Ejemplo de envío de un dato al DAC MCP4821-MCP4822

SPI1_Init();

⋮

Paquete_H

0		1	1				
$\overline{A}$/B	—	$\overline{GA}$	$\overline{SHDN}$	D11	D10	D9	D8
bit 15							bit 8

Paquete_L

D7	D6	D5	D4	D3	D2	D1	D0
bit 7							bit 0

- Para iniciar la comunicación con el DAC hay que poner su terminal *CS* a 0 (se supone que el terminal *LDAC* está a 0)

- SPI1_Write(Paquete_H); //se envía al DAC el *byte alto* del paquete de 16 bits.

- SPI1_Write(Paquete_L); //se envía al DAC el *byte bajo* del paquete.

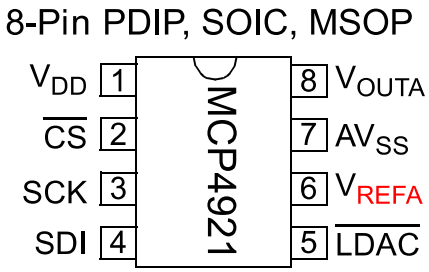
- Para finalizar la comunicación con el DAC hay que poner el terminal *CS* a 1 (se supone que el terminal LDAC está a 0). En el momento en el que se pone *CS* a 1 se inicia la conversión del dato de 12 bits que ha recibido el DAC (en un paquete de 16 bits).

Nota: Paquete_H = 0x30 | (paquete >> 8) & 0x0F; // NO FUNCIONA

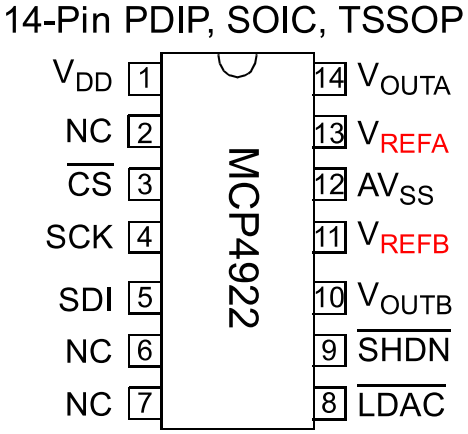
Paquete_H = 0x30 | ((paquete >> 8) & 0x0F); // FUNCIONA

no

Notas MDACs MCP4921-MCP4922 (*multiplying* DACs)



- El MCP4921 tiene 1 convertidor MDAC de 12 bits
- El MCP4922 tiene 2 convertidores MDAC de 12 bits
- La comunicación con el MDAC se realiza por medio de un bus SPI
- $t_{settling\ time} = 4{,}5\mu seg$



- $V_{DD} \in [2'7v, 5'5v]$
- $AV_{SS} \equiv GND$
- $V_{REF} \in [0.04 \div V_{DD} - 0.04]$ *buffered mode*

  $V_{REF} \in [0 \div V_{DD}]$ *unbuffered mode*

- $v_{out} = \frac{V_{REF} \cdot Gain \cdot D}{2^{12}}$ (D es el dato a enviar al DAC y representa el valor de una muestra)
- $\overline{CS}$: se tiene que poner a 0 para establecer una comunicación (por SPI) con el DAC

REGISTER 5-1: WRITE COMMAND REGISTER

no

Upper Half:							
W-x	W-x	W-x	W-0	W-x	W-x	W-x	W-x
$\overline{A}$/B	BUF	$\overline{GA}$	$\overline{SHDN}$	D11	D10	D9	D8
bit 15							bit 8

Lower Half:							
W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
D7	D6	D5	D4	D3	D2	D1	D0
bit 7							bit 0

- $\overline{A}/B = 0$ el dato enviado es para el convertidor DAC_A

  $\overline{A}/B = 1$ el dato enviado es para el convertidor DAC_B

- BUF = 0 *unbuffered* $R_{VREF} \approx 165k\Omega$ $C_{VREF} \approx 7pF$

  BUF = 1 *buffered* Z_{VREF} *muy alta*

- $\overline{GA} = 1$ (Gain = 1) $\rightarrow$ $v_{out} = \frac{V_{REF} \cdot D}{2^{12}}$

  $\overline{GA} = 0$ (Gain = 2) $\rightarrow$ $v_{out} = \frac{2 \cdot V_{REF} \cdot D}{2^{12}}$

no

- $\overline{SHDN} = 1 \rightarrow$ el convertidor funciona

  $\overline{SHDN} = 0 \rightarrow$ la salida se pone en tercer estado (en un estado de alta impedancia) ≡ el convertidor no funciona !!!.

- $D_{11} - D_0 \in [0, 4095]$: dato a convertir en una tensión

Nota: todas las demás características son iguales a las descritas para los DAC MCP4821 y MCP4822

Nota: el DAC no tiene terminal SDO/MISO porque no envía datos, sólo los recibe.

Nota: *Big endian* indica que en primer lugar se envía el byte más significativo y que a continuación se envía el byte menos significativo.

Ejemplo de envío de un dato al DAC MCP4921-MCP4922

no

Paquete_H

0	0	1	1				
$\overline{A}$/B	BUF	$\overline{GA}$	$\overline{SHDN}$	D11	D10	D9	D8
bit 15							bit 8

Paquete_L

D7	D6	D5	D4	D3	D2	D1	D0
bit 7							bit 0

SPI1_Init();

⋮

- Para iniciar la comunicación con el DAC hay que poner su terminal *CS* a 0 (se supone que su terminal *LDAC* está a 0)

- SPI1_Write(Paquete_H); //se envía al DAC el *byte alto* del paquete de 16 bits.

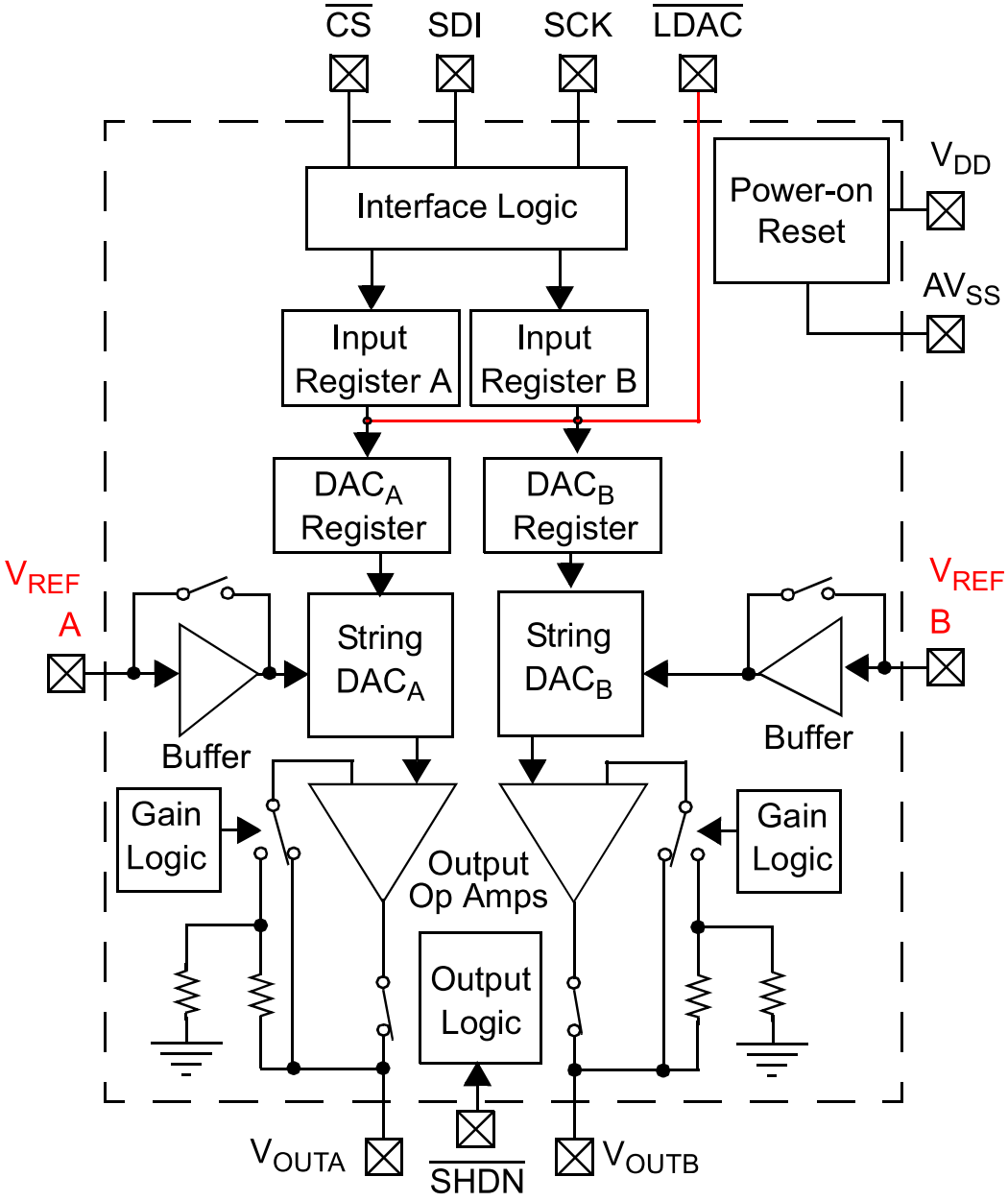
- SPI1_Write(Paquete_L); //se envía al DAC el *byte bajo* del paquete.

- Para finalizar la comunicación con el DAC hay que poner el terminal *CS* a 1 (se supone que el terminal LDAC está a 0). En el momento en el que se pone *CS* a 1 se inicia la conversión del dato de 12 bits que ha recibido el DAC (en un paquete de 16 bits).

Nota: Paquete_H = 0x30 | (paquete >> 8) & 0x0F; // NO FUNCIONA

Paquete_H = 0x30 | ((paquete >> 8) & 0x0F); // FUNCIONA

no


CS
SDI
SCK
LDAC
Interface Logic
Power-on Reset
VDD
AVSS
Input Register A
Input Register B
DACA Register
DACB Register
VREF A
VREF B
String DACA
String DACB
Buffer
Buffer
Gain Logic
Gain Logic
Output Op Amps
Output Logic
VOUTA
SHDN
VOUTB

NOTAS SOBRE EL BUS SERIAL PERIPHERAL INTERFACE (SPI)

- El SPI es un protocolo de comunicación serie, de alta velocidad, desarrollado por la empresa *Motorola*.

- Está pensado para intercomunicar todo tipo de dispositivos a pequeñas distancias (dentro de una misma placa de circuito impreso).

- La comunicación se realiza siempre entre un dispositivo denominado *master* y un dispositivo denominado *slave*, conectados al bus.

- La comunicación entre el *master* y el *slave* puede ser simultánea (*full duplex*), a diferencia de lo que ocurre con el protocolo I2C.

- El bus SPI utiliza cuatro líneas unidireccionales, además de masa, para transmitir datos entre 2 y 16 bits:

MOSI (*Master Output, Slave Input*): el dispositivo que actúa como *master* envía información por éste terminal al dispositivo que actúa como *slave*.

MISO (*Master Input, Slave Output*): el dispositivo que actúa como *slave* envía información al *master* por éste terminal.

SCLK ó SCK (*Serial clock*): este terminal porta la señal de reloj que sincroniza la comunicación entre master y el slave. En cada periodo de la señal *SCLK* se transmite un bit. El *master* es el encargado de generar la señal *SCLK*.

(*Slave Select*): el *master* utiliza este terminal para seleccionar el dispositivo *slave* con el que quiere comunicarse. Cada dispositivo *slave* tiene su propia entrada de habilitación (activa a nivel bajo) que hay que conectar al *master*.

- Aunque este protocolo no es un estándar, su popularidad es tal que se puede considerar que lo es de facto.

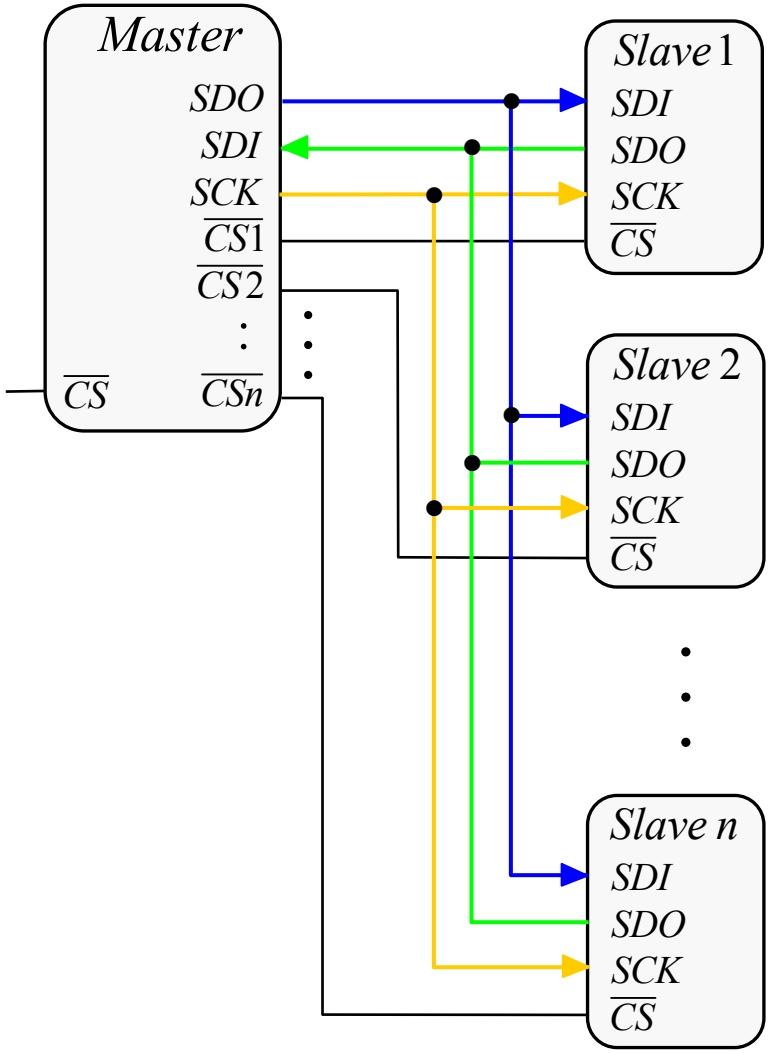
- En algunos documentos técnicos los terminales/líneas de un bus SPI se denominan de la siguiente forma:

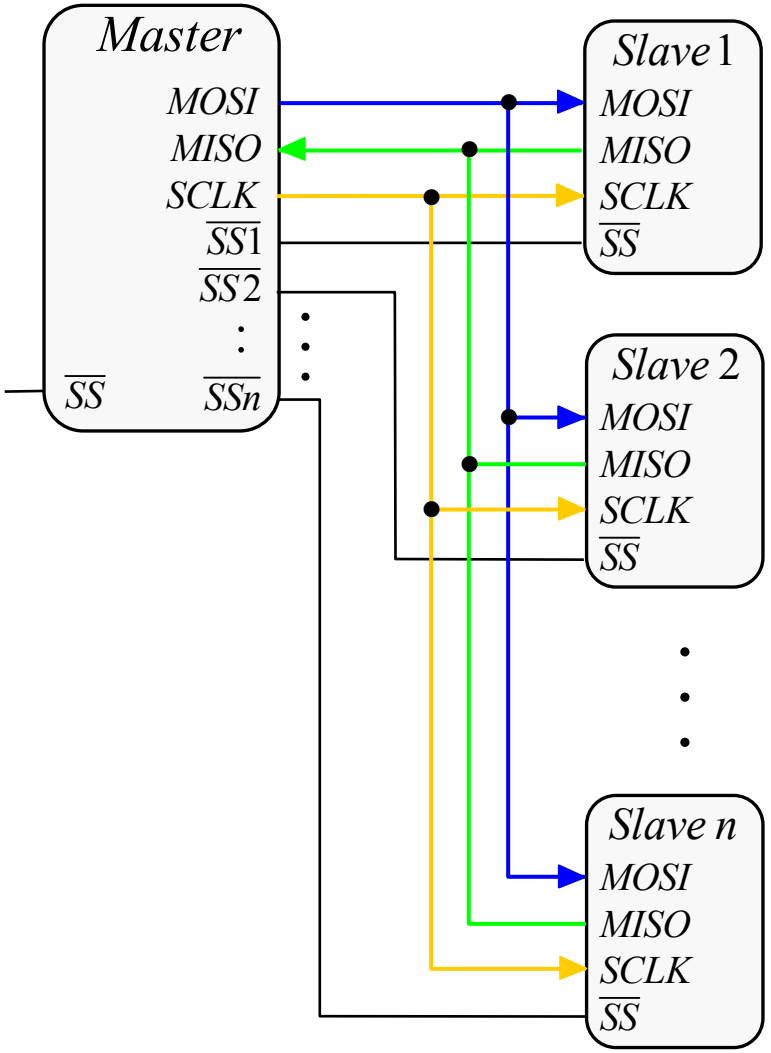
SDI (*Serial Data In*): terminal por el que se reciben los datos.

SDO (*Serial Data Out*): terminal por el que se envían los datos.

SCK (*Serial Clock*): terminal de sincronismo.

CS (*Chip Select*): terminal de habilitación de un dispositivo *slave*.


Master
SDO
SDI
SCK
CS1
CS2
CS
CSn
Slave 1
SDI
SDO
SCK
CS
Slave 2
SDI
SDO
SCK
CS
Slave n
SDI
SDO
SCK
CS


Master
MOSI
MISO
SCLK
SS1
SS2
SS
SSn
Slave 1
MOSI
MISO
SCLK
SS
Slave 2
MOSI
MISO
SCLK
SS
Slave n
MOSI
MISO
SCLK
SS

▪ En la mayoría de los sistemas de comunicación SPI se pueden configurar muchas características, entre las que cabe destacar las siguientes:

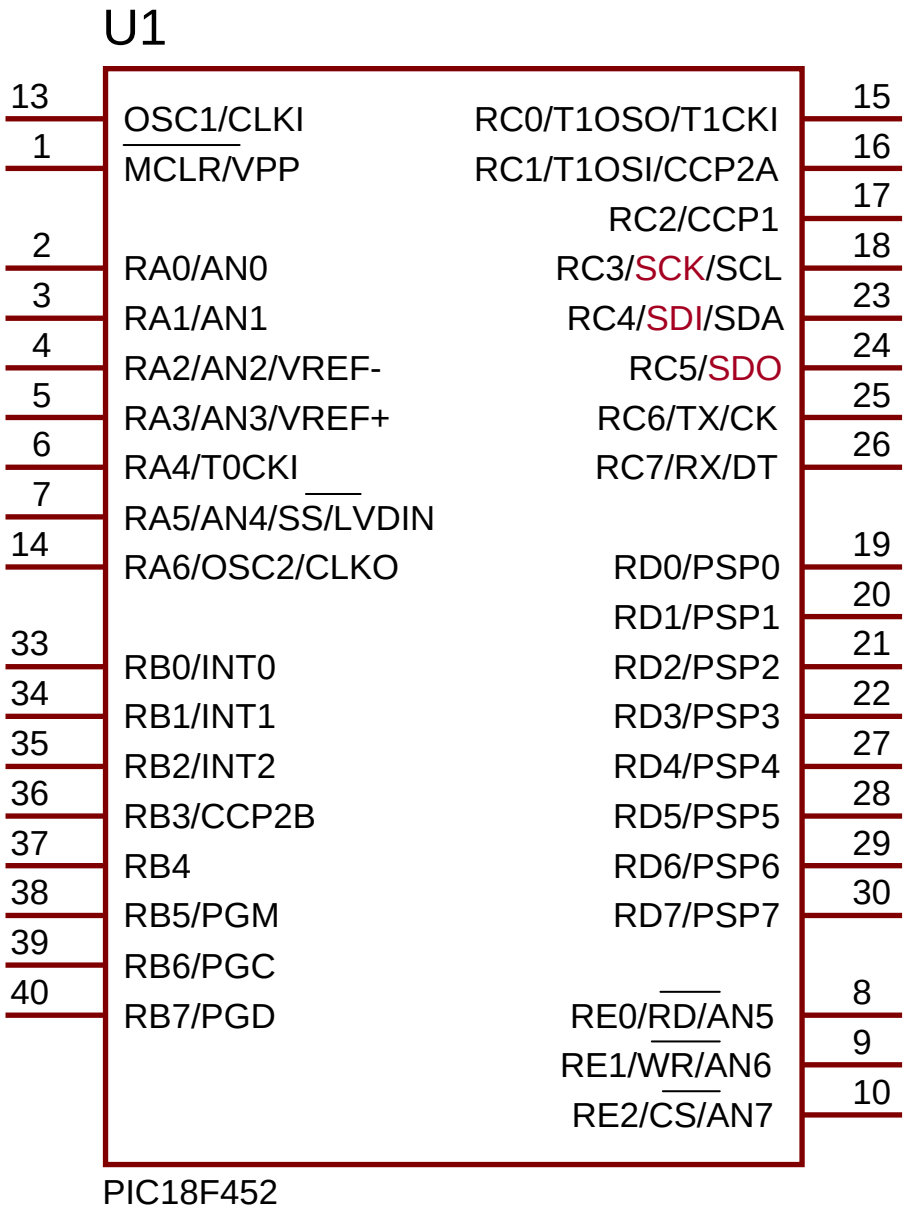
_ *El tamaño de los datos a transmitir*: su valor debe estar comprendido entre 2 y 16 bits

_ *El orden de envío de los bits*: se puede elegir si se envía el byte más significativo en primer lugar o de último (*big endian* o *little endian*).

▪ *El formato de transmisión*: existen cuatro formatos o modos de transmisión de los datos. La mayoría de los sistemas soportan al menos dos de ellos. El formato de transmisión se especifica mediante dos bits de control denominados *CPLD* (*clock polarity*) y *CPHA* (*clock phase*).

Nota: En el modo *big endian* el byte más significativo se guarda en la dirección más baja y el byte menos significativo en la dirección más alta. En el modo *little endian* ocurre lo contrario.

*PIC*18*F*452


U1
13
OSC1/CLKI
1
MCLR/VPP
2
RA0/AN0
3
RA1/AN1
4
RA2/AN2/VREF-
5
RA3/AN3/VREF+
6
RA4/T0CKI
7
RA5/AN4/SS/LVDIN
14
RA6/OSC2/CLKO
33
RB0/INT0
34
RB1/INT1
35
RB2/INT2
36
RB3/CCP2B
37
RB4
38
RB5/PGM
39
RB6/PGC
40
RB7/PGD
15
RC0/T1OSO/T1CKI
16
RC1/T1OSI/CCP2A
17
RC2/CCP1
18
RC3/SCK/SCL
23
RC4/SDI/SDA
24
RC5/SDO
25
RC6/TX/CK
26
RC7/RX/DT
19
RD0/PSP0
20
RD1/PSP1
21
RD2/PSP2
22
RD3/PSP3
27
RD4/PSP4
28
RD5/PSP5
29
RD6/PSP6
30
RD7/PSP7
8
RE0/RD/AN5
9
RE1/WR/AN6
10
RE2/CS/AN7
PIC18F452