# El CD4094

El CD4094 es un registro de desplazamiento de 8 bits, con salida tri-estado. Esto significa que dispone de un mecanismo para aislar sus 8 pines de salida del resto del circuito. Los datos son desplazados serialmente con cada flanco de subida del reloj (CLOCK) y cada bit es transmitido al latch correspondiente con cada flanco de bajada del pin STROBE.

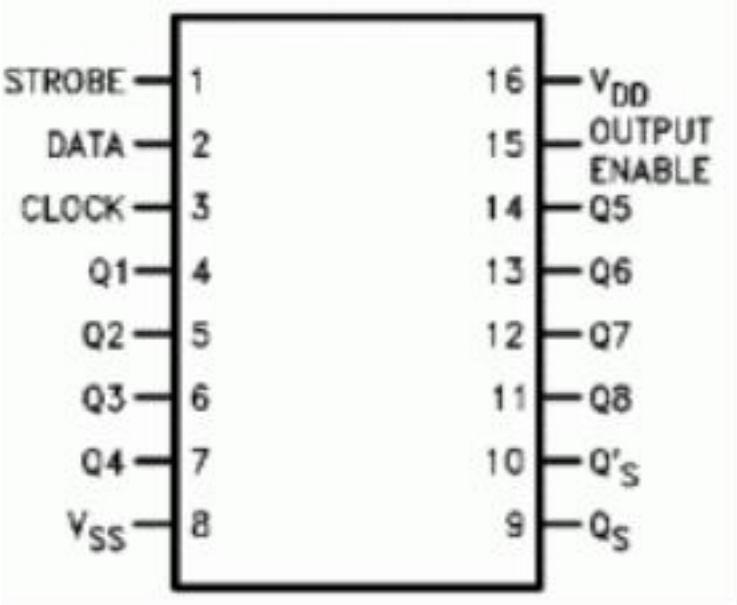



*Figura 1: Diagrama de Pines del CD4094*

Las características más destacadas de este circuito integrado son:

- Rango de voltaje: 3.0V a 18.0V.
- Compatibilidad con la familia TTL.
- Salida tri-estado.

Oscar Isaac Tinajero Maldonado

# ¿Por qué usar un registro de desplazamiento?

Esto es sencillo de responder, pues la respuesta más rápida es que se ahorran varias líneas de código. En el caso de usar un display de 7 segmentos se ahorran 5 pines, ya que se necesitan solo 3 para controlar los 7 segmentos y el punto decimal. Si se necesita controlar un número mayor de displays, el ahorro es mayor, ya que con las mismas tres líneas usadas para controlar un display se pueden controlar todos los que se necesiten, realizando una especie de ScrollText.

Para que el CD4094 pueda manejar un display de 7 segmentos, tienen que existir las señales de control establecidas, las cuáles pueden ser gestionadas con un microcontrolador, el cuál puede ser un PIC, un Arduino, un ESP32-CAM o cualquier placa de desarrollo.

# Diagrama de tiempos:

Para establecer las señales de control en el CD4094, es necesario estudiar su diagrama de tiempos:

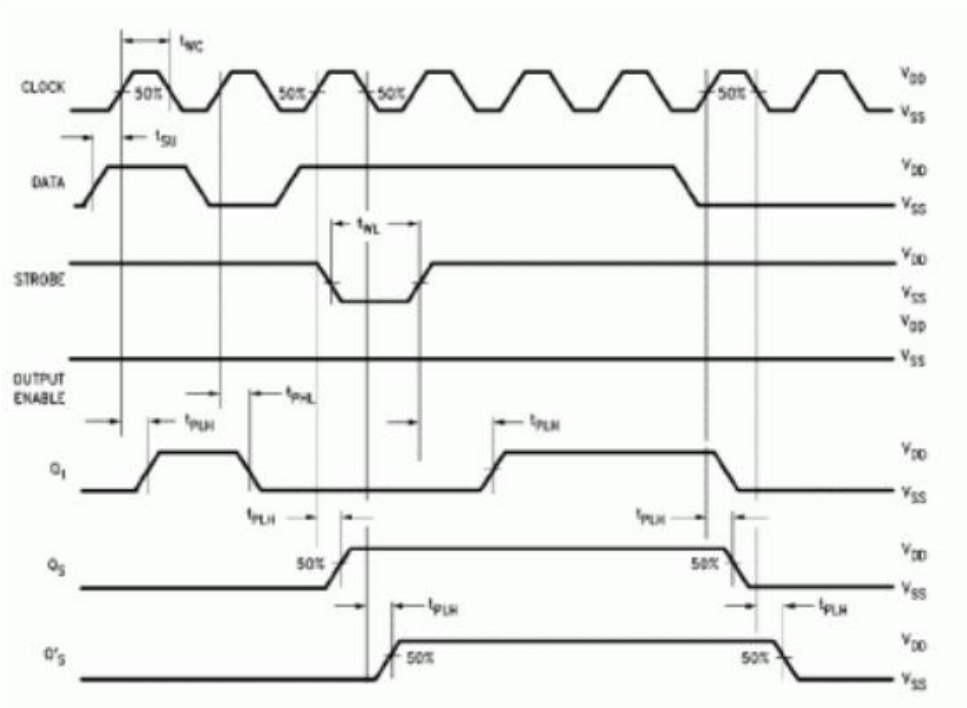



*Figura 2: Diagrama de tiempos*

La primera señal de control es DATA, que es el dato que se envía serialmente. Se envía un flanco ascendente y luego un flanco descendente. En el instante que el CLOCK esté subiendo, el estado de DATA será reconocido como un dato válido por el CD4094.

La señal de control STROBE se usa en caso de que se quieran tener en los latch de salida los bits enviados por el microcontrolador o placa de desarrollo que se van recibiendo.

Por último, el pin OUTPUT ENABLE. Este pin cumple la función del tercer estado del latch, el estado de alta impedancia. Generalmente se emplea en sistemas de buses, donde se conectan varios integrados a las mismas señales, permitiendo aislar un circuito de otro.

Este pequeño integrado de 16 pines, es un conversor de Serial a Paralelo, aceptando bits en forma serial desde un microcontrolador u otro CD4094 (necesita 3 pines para funcionar) y cuando se da la orden, arroja el byte por 8 de sus patas de una sola vez. Como agregado, tiene la bondad de funcionar en un rango de 3V a 18V. Usa 2 pines para alimentarse (Pin 16 y Pin 8), 3 pines para recibir información del integrado (DATA, CLOCK y STROBE), 8 pines para sacar el byte que se transmitió (Q1 a Q8), y 2 pines más para encadenarse a otro CD4094 (Qs y Q's).

# ¿Cómo funciona?

Explicaré con extractos de códigos como funciona este registro, los de la izquierda serán de Lenguaje *Ensamblador* para un **PIC16F84A** y los de la derecha serán en *C++* para una placa de desarrollo **ESP32-CAM**.

1. Presentar el bit en el pin **DATA**.

```
;*************** Letras de forma manual (DATA) 4***
LB      movlw   0x00        ;bit    B       Q1
        movwf   PORTB
        call    GUARDAR
        movlw   0x00        ;bit    A       Q2
        movwf   PORTB
        call    GUARDAR
        movlw   0x04        ;bit    F       Q3
        movwf   PORTB
        call    GUARDAR
        movlw   0x04        ;bit    G       Q4
        movwf   PORTB
        call    GUARDAR
        movlw   0x00        ;bit    DP      Q5
        movwf   PORTB
        call    GUARDAR
        movlw   0x04        ;bit    C       Q6
        movwf   PORTB
        call    GUARDAR
        movlw   0x04        ;bit    D       Q7
        movwf   PORTB
        call    GUARDAR
        movlw   0x04        ;bit    E       Q8
        movwf   PORTB
        call    GUARDAR
        call    ENVIAR
        return
;*****************************
```

```
shiftOut(pinData, pinClock, MSBFIRST, s);
```

2. Indicar al CD4094 que se le está transmitiendo un bit, mandando un 1 y luego un 0 al pin CLOCK.

```
;*************** Guardar bit (CLOCK) RB3 ***
GUARDAR movlw   0x08
        movwf   PORTB
        movlw   0x00
        movwf   PORTB
        return
```

```
shiftOut(pinData, pinClock, MSBFIRST, s);
```

3. Se repiten los pasos 1 y 2 para todos los bits que se quieran transmitir (si son más de 8 bits y se cuenta con más de un CD4094, el primero los empezará a pasar al que viene después de él en la cadena, y así hasta llegar al último).

4. Una vez que se transmitieron todos los bits, se indica que los mande por los pines Q1 al Q8, mandando un 1 y luego un 0 al pin STROBE.

```
;***************** Eviar bit al display (STROBE)
ENVIAR  movlw   0x02
        movwf   PORTB
        movlw   0x00
        movwf   PORTB
        return
```

```
s = secuencias[2];
digitalWrite(pinStrobe,LOW);
shiftOut(pinData, pinClock, MSBFIRST, s);
delay(seg);
digitalWrite(pinStrobe,HIGH);
```

Como se puede observar, el procedimiento es más sencillo con una placa de desarrollo que con un PIC programado Ensamblador, pero se ve más detalladamente como funciona el CD4094 al usar las instrucciones en el lenguaje ensamblador.

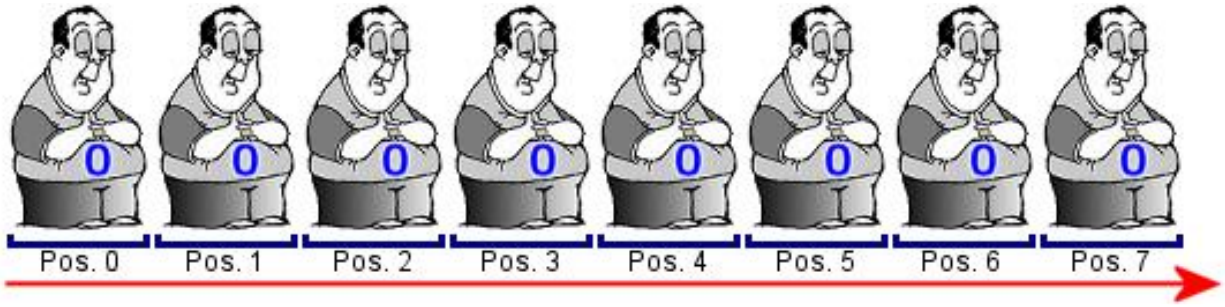
## Analogía explicativa del funcionamiento del registro de desplazamiento CD4094.

La mejor manera de entender este concepto es con el apoyo de analogías que nos son familiares. En este caso se usará el ejemplo del funcionamiento de una cola, como la de un banco, un supermercado, “la de las tortillas” o la de la beca.

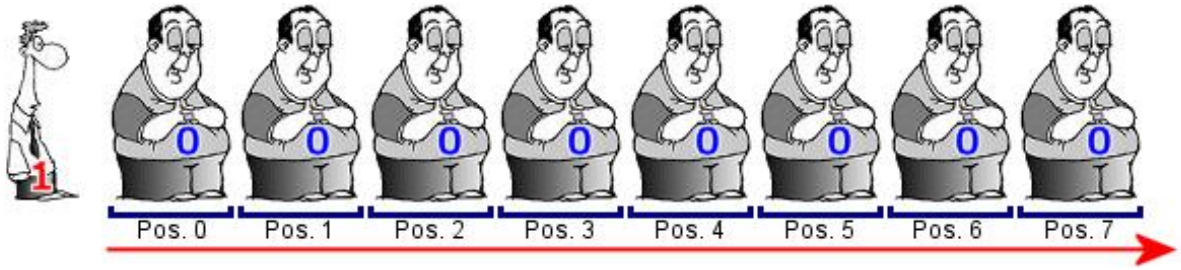
Supongamos que dos tipos de personas forman parte de una cola. Estos dos tipos son los que se observan en la figura siguiente, y es imposible confundirlos con alguien más, es decir, siempre estaremos seguros que en una posición determinada de la cola esta una u otra persona. Las llamaremos “0” (el “gordito”) y “1” (el “delgado”). Esto no discrimina a nadie, solo hace más fácil de entender la analogía.

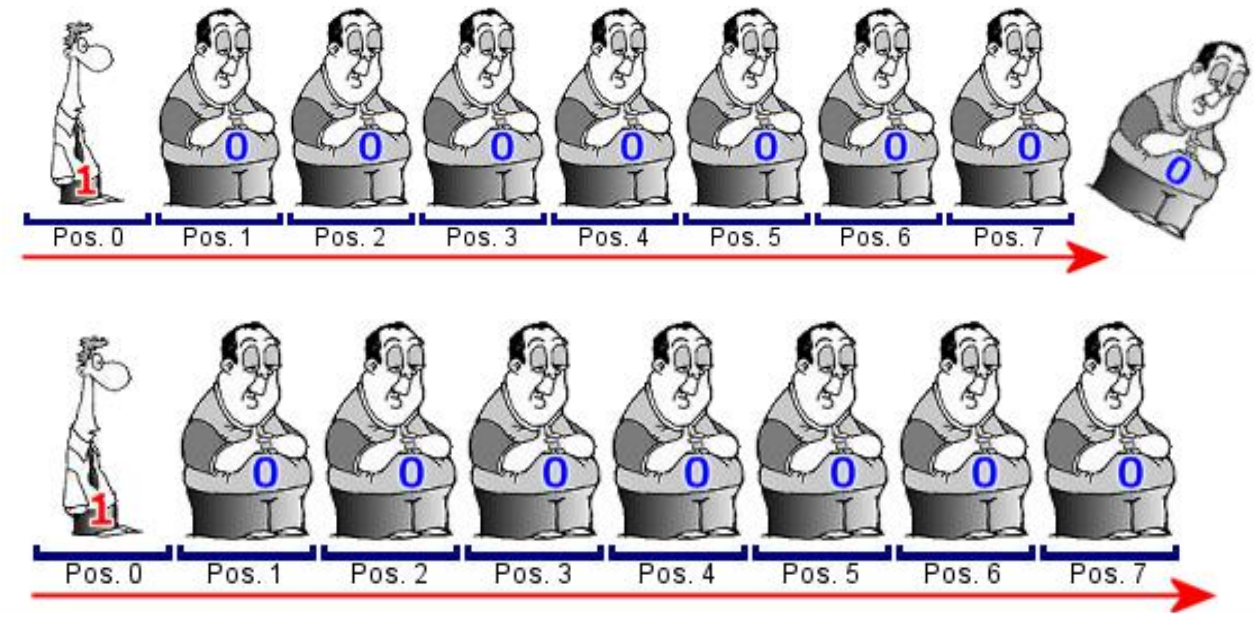
La cola que usaremos como ejemplo tiene 8 lugares, su numeración va del 0 al 7, esto no significa que no se puedan usar colas más largas, por lo cual, todo lo que se verá en este ejemplo puede ser generalizado para colas de la longitud que se desee.


Otra particularidad de nuestra cola hipotética, es que nunca puede estar vacía, todas sus posiciones tienen que estar ocupadas, ya sea por “gorditos” o “delgados” (0 y 1). En el estado inicial, la cola se encuentra completamente llena de “gorditos”, como se observa a continuación:



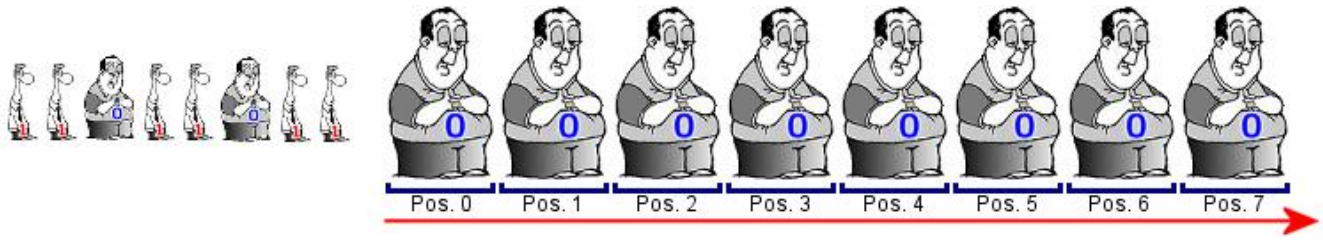

Nuestra cola funciona como cualquier cola de la vida real: cuando alguien nuevo llega a la cola, se coloca en el lugar de más atrás, que en este caso corresponde a la “posición 0”. Como la cola tiene una longitud máxima (en nuestro ejemplo) de 8 posiciones, para hacer lugar al recién llegado, es necesario que todos los que estaban en la fila “avancen” una posición. El que estaba en la posición ‘0’ pasa a la posición ‘1’, el que estaba en la ‘1’ a la ‘2’ y así hasta llegar al que estaba en la posición 7, que “sale” por el extremo opuesto.

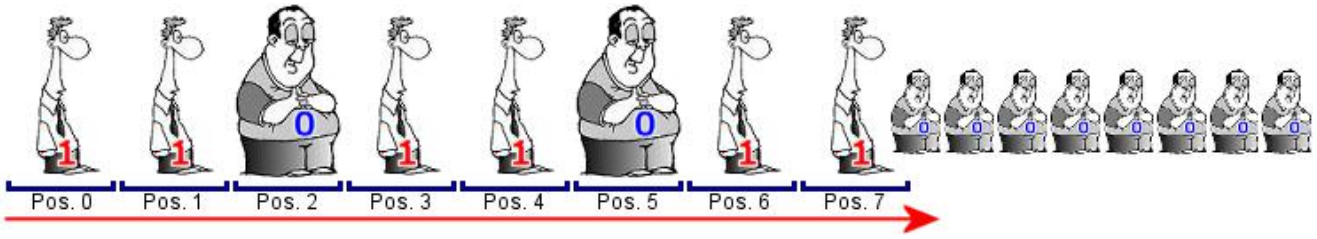





El proceso se repetirá conforme vayan llegando nuevas personas a la cola, a este tipo de colas se le conoce como **“FIFO”** (**F**irst **I**nput, **F**irst **O**utput, Primero en entrar es el primero en salir).

Con toda esta información, se puede avanzar en la comprensión del funcionamiento de los registros de desplazamiento. Supongamos que queremos que en la cola hayan dos “delgados” en los primeros lugares, luego un “gordito”, luego otra vez dos “delgados” y repetimos la secuencia hasta que se completen los 8 lugares. Sabemos que cada que alguien ingresa a la cola se “empuja” a todos los que ya estaban en ella una posición a la derecha, si queremos que el que termine ocupando el extremo derecho de la cola sea un “delgado”, ese será el que primero debe entrar. Siguiendo el razonamiento anterior, los personajes deberían entrar en la cola en el siguiente orden:






Terminando con la analogía, tendríamos que los integrantes de esta hipotética cola son los ‘0’s y los ‘1’s (estados altos y bajos, encendidos o apagados, etc.) de nuestros circuitos, es decir, nuestros datos. La cola en si es el registro de desplazamiento. Cuando decimos que el estado inicial de la cola eran 8 “gorditos”, decíamos que todos nuestros estados son ‘0’s al iniciar nuestro circuito, o que los datos de nuestro registro estaban abajo o apagados.

Hay una salvedad, y es la existencia del "reloj". Efectivamente, en un circuito real, los datos pasan al registro de desplazamiento con cada pulso de reloj. Podemos pensar en este reloj como si se tratase de un "maestro de ceremonias", que da una palmada cada vez que alguien debe ingresar en la cola.

Muchos circuitos de registros de desplazamiento "reales" también incluyen un sistema de RESET, que permite poner simultáneamente todas las salidas en "0" o estado bajo, sin necesidad de ingresar 8 ceros seguidos. Esto permite limpiar rápidamente el registro de desplazamiento.

Cuando decimos "*rápidamente*" nos referimos a que como la velocidad de los pulsos del reloj (CLOCK) no puede ser infinita (típicamente el máximo ronda los 10 o 20 MHz) y cada dato demora el tiempo de un pulso de reloj en desplazarse por el registro, introducir 8 "0"s llevaría 800 ns (100 ns * 8 bits), contra los 100 ns que demora en aplicarse el RESET. No obstante, para obtener los tiempos exactos implicados se debe consultar la hoja de datos del integrado que estemos utilizando, ya que los limites varían incluso con la tensión de alimentación y la temperatura.

Espero haberte ayudado con está información, si lo deseas, puedes seguirme en mis redes sociales:

https://github.com/OscarTinajero117

https://www.facebook.com/Aaisacks-Things-103263218507978

https://twitter.com/117Tinajero

https://www.linkedin.com/in/oscar-isaac-tinajero-maldonado-48b61b199/

https://www.youtube.com/c/oscartinajero117/featured


Oscar Isaac Tinajero Maldonado