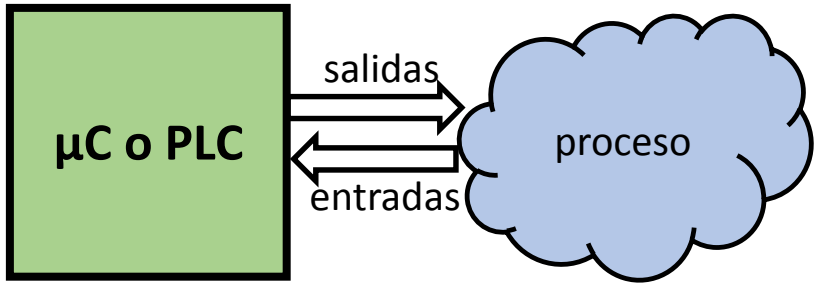
Programación de un automatismo

Un **automatismo**, en el ámbito que nos ocupa, es un dispositivo que realiza alguna tarea en interacción con el mundo físico de forma automática. Se puede resolver en forma eléctrica, neumática, hidráulica o electrónica. En forma electrónica puede ser a través de un circuito lógico (combinacional/secuencial) o mediante un sistema programable.



µC: microcontrolador
PLC: controlador lógico programable

Nos ocupamos particularmente de los **automatismos programables** porque son los más flexibles y versátiles.
En este breve apunte trataremos de mostrar los rudimentos de la programación de un automatismo. El objetivo no es aprender a programar sino verificar que es posible pasar de una descripción de funcionamiento de un automatismo, realizada en lenguaje natural, a un **diagrama** que luego permitirá realizar el **programa** que se ejecute en algún tipo de sistema programable (PLC, microcontrolador etc).

En los proyectos integradores de Industrial pedimos que supervisen un automatismo mediante un software SCADA, pero también que representen su funcionamiento mediante un diagrama.

Ejemplo: Portón automático

Esquema físico (muestra los elementos físicos y su interconexión)

El bloque celeste es el **controlador**, que tiene 4 entradas y 2 salidas:

Entradas. Allí ingresan las señales de:

PA: Pulsador para abrir

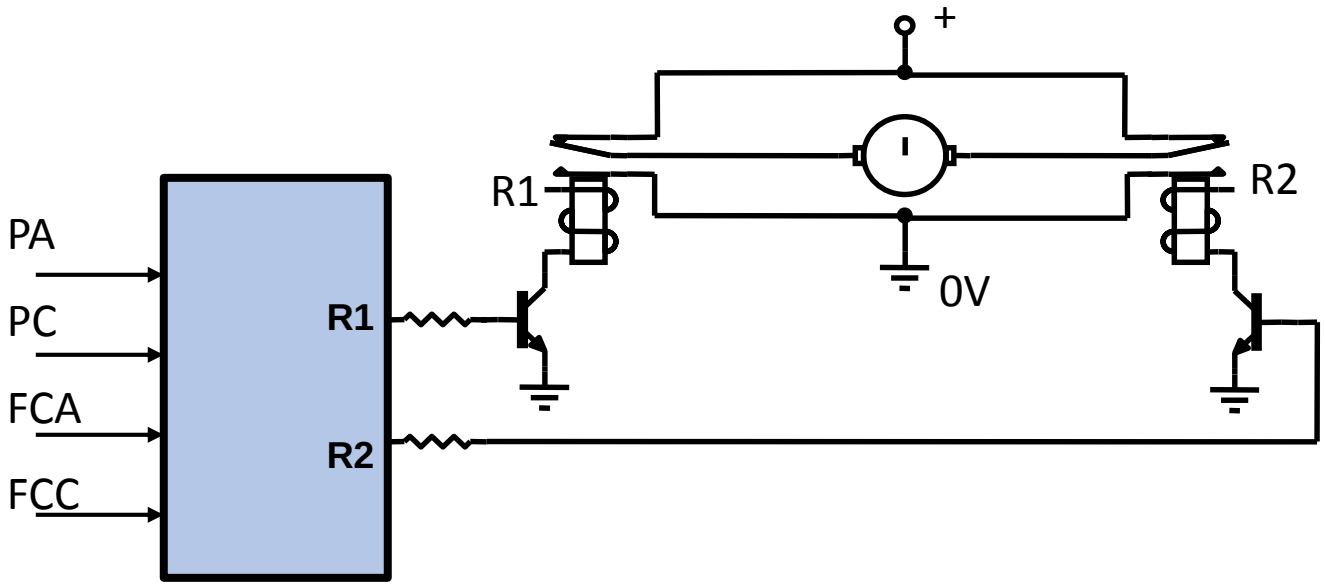
PC: Pulsador para cerrar

FCA: Final de carrera al abrir

FCC: Final de carrera al cerrar

Salidas. Allí salen las señales a:

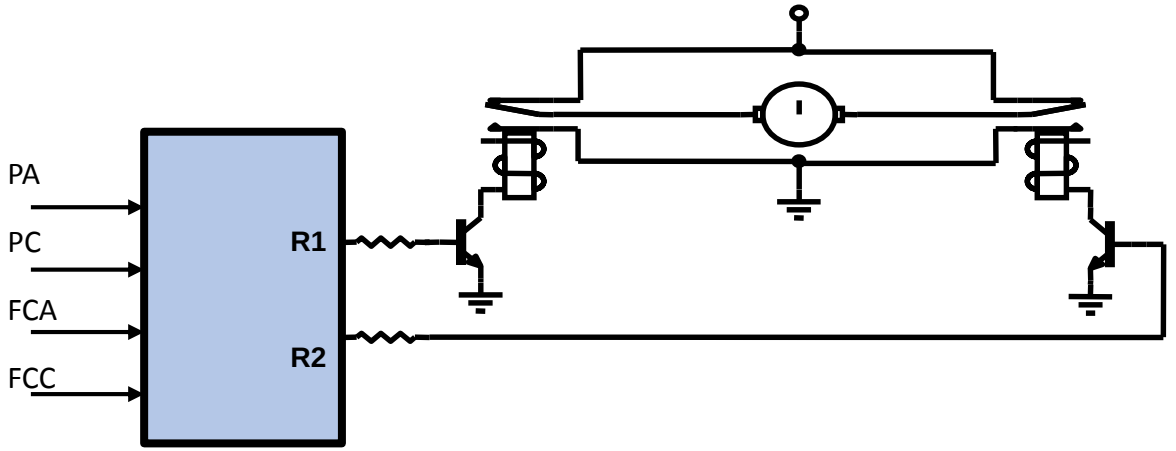
R1, R2: Relés para giro directo-inverso de motor (se usan transistores para activar los relés). Como se ve, si R1 conecta a (+) y R2 conecta a 0V, el motor gira en un sentido. Si R1 conecta a 0V y R2 a (+), gira en el otro sentido. En otro caso el motor no gira.



Ejemplo: Portón automático

Comportamiento del automatismo (lógica de funcionamiento):

Funcionamiento básico:
El portón se encuentra inicialmente en **Reposo (motor detenido)**.
Si se pulsa PA el portón se debe **abrir**, y parar cuando se detecta FCA.
Si se pulsa PC el portón se debe **cerrar**, y parar cuando se detecta FCC.



Opciones:
¿Se supone el portón inicialmente cerrado? ¿Se cierra inicialmente?
¿Mientras el portón se está moviendo ignora los pulsadores? (SI/NO)

Representación del comportamiento

El funcionamiento del automatismo se puede representar – por ejemplo – mediante un **Diagrama de Estados** o mediante un **Diagrama de Flujo**. El Diagrama de Estados suele ser una representación de más alto nivel de abstracción, más cercana a la descripción del comportamiento que se ha hecho en lenguaje natural. El Diagrama de Flujo es una representación más detallista, y suele estar más cercana a la codificación del programa.

Diagrama de Estados

Leyendo la descripción del funcionamiento se reconocen 3 estados.

Reposo: El motor está detenido (portón abierto o cerrado). R1=R2=0 (o R1=R2=1)

Abre: El motor gira en sentido de abrir portón R1=1 R2=0

Cierra: El motor gira en sentido de cerrar portón R1=0 R2=1

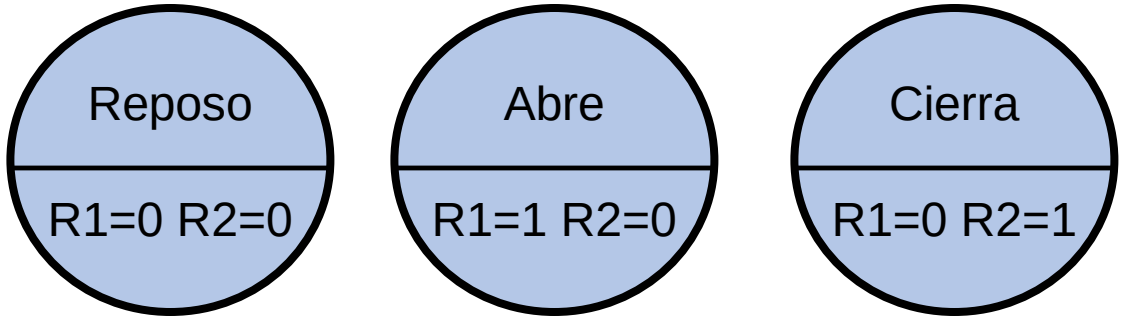


Diagrama de Estados (2)

Opción: Supone portón inicialmente cerrado

En el Diagrama de Estados se representan los estados mediante círculos, elipses u otras figuras, y las transiciones entre estos estados mediante flechas. Sobre las flechas se escribe la condición o evento que causa la transición. En este ejemplo, del estado Inicial se pasa al de reposo (siempre, sin condición). Del **Reposo** se pasa al estado **Abre** en caso de que se pulse PA, y se vuelve a Reposo si se activa FCA.
De forma similar es con el estado Cierra y las señales PC y FCC.

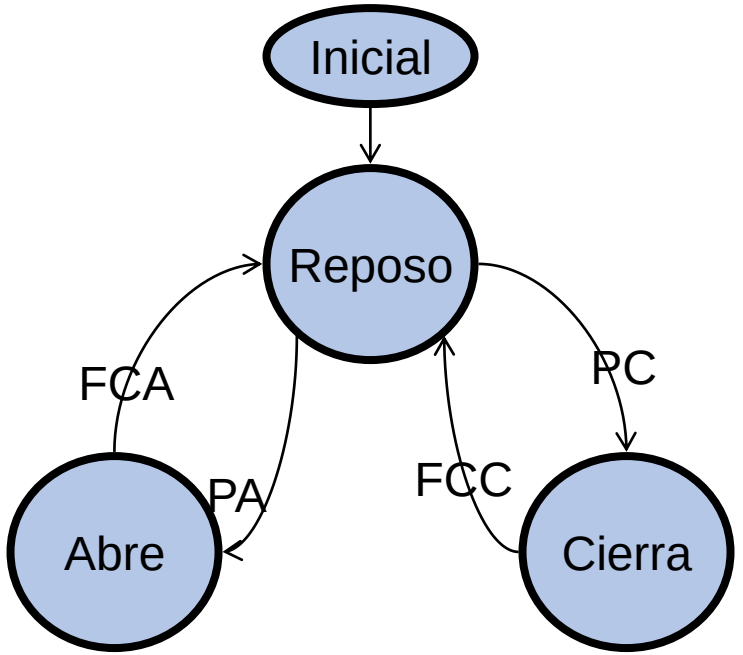


Diagrama de Estados (3)

Opción: Inicialmente se cierra

Este diagrama es similar al anterior, pero supone una “mejora” en el comportamiento. En este caso, al energizar el automatismo pasa al estado Cierra, de forma de que no quede inicialmente en reposo si el portón estaba abierto.
Observar que si al iniciar el automatismo el portón ya estaba cerrado, se detectará FCC y se pasará al reposo casi instantáneamente.

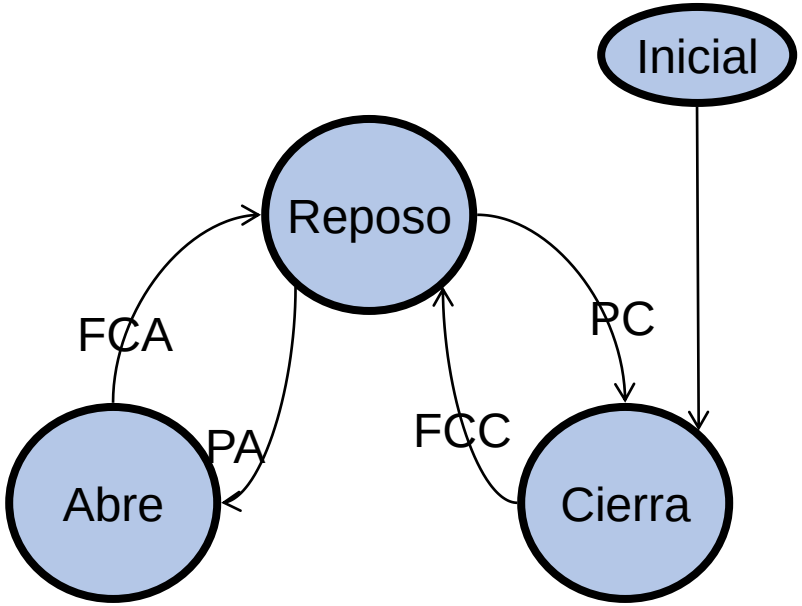


Diagrama de Estados (4)

Opción: Atiende pulsadores para invertir marcha

Otra mejora al anterior puede ser que no haya que esperar a que el portón se abra totalmente para poder dar la orden de cerrarlo con PC, o cerrar totalmente para poder dar la orden de abrirlo con PA. En este caso mientras está en estado Abre el automatismo es sensible al evento PC y permite pasar al estado Cierra (y cambiar el sentido del movimiento instantáneamente)

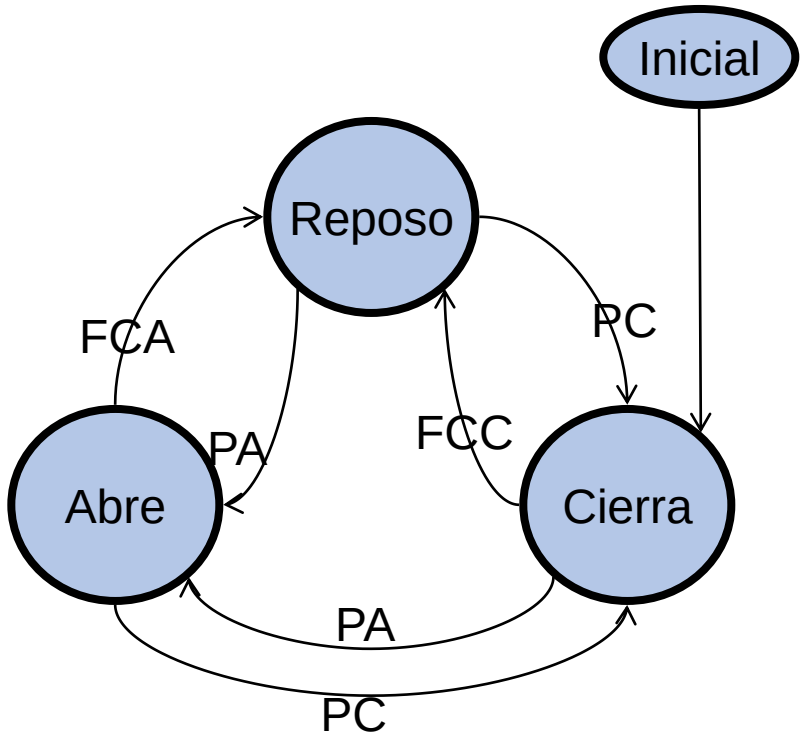


Diagrama de Estados (5)

Opción: Atiende pulsadores para invertir marcha pero previamente pasa por Reposo

Similar al anterior, pero en este caso hace un breve paso por el estado Reposo antes de cambiar de Abre a Cierra o de Cierra a Abre. Esto podría servir para evitar un cambio brusco del sentido (si se agrega un breve retardo a la salida del estado Reposo)

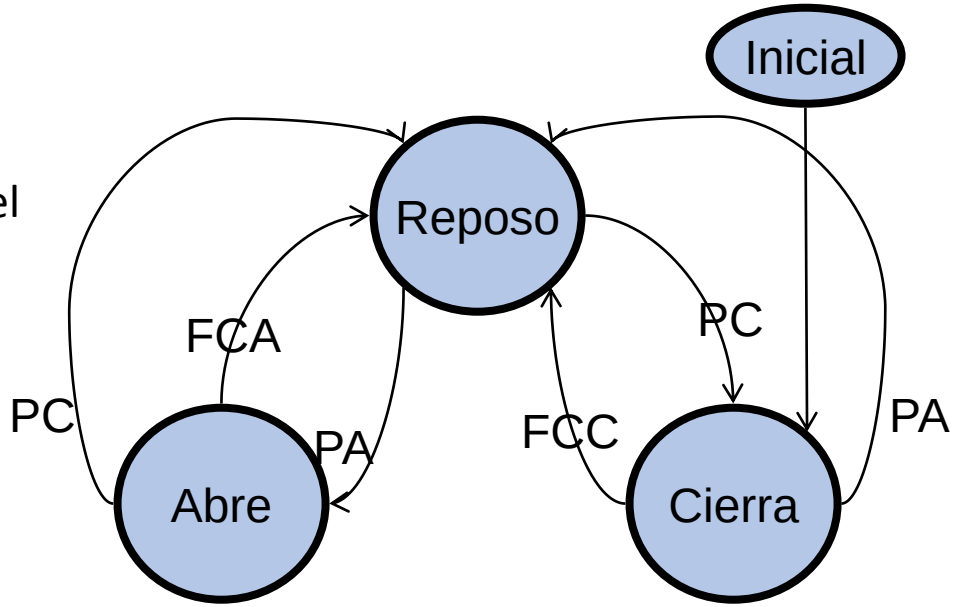
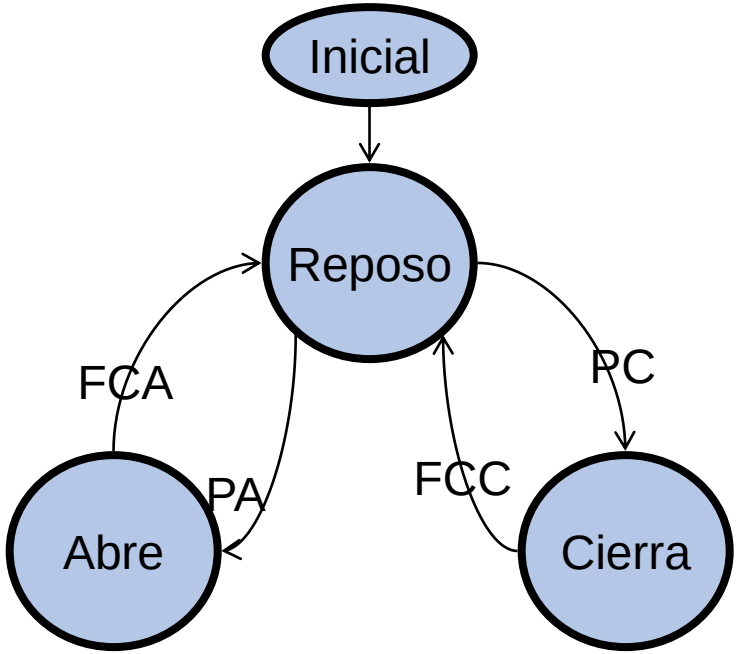
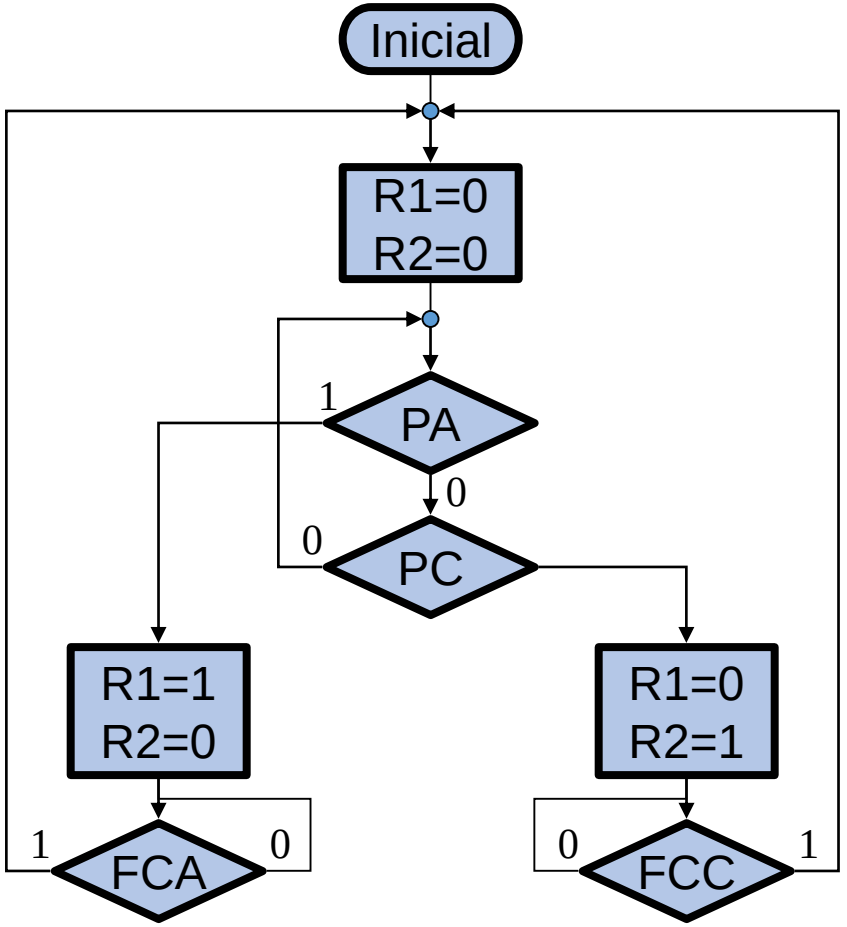


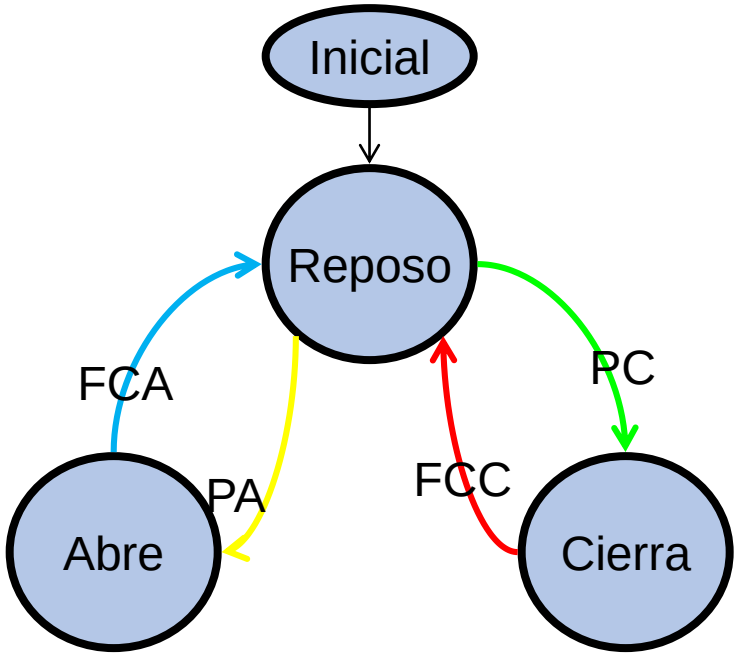
Diagrama de Estado y Diagrama de Flujo

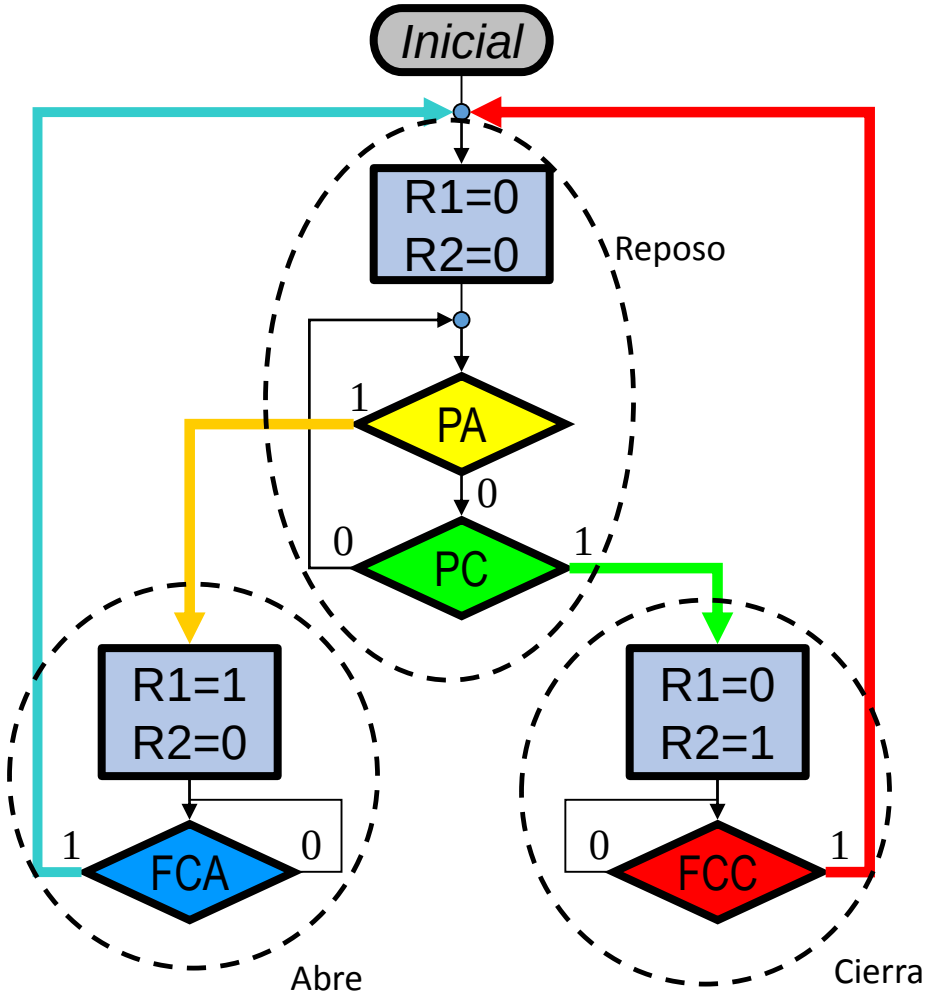




Retomando el ejemplo más simple, veremos que es posible implementar un Diagrama de Estado a través de un Diagrama de Flujo

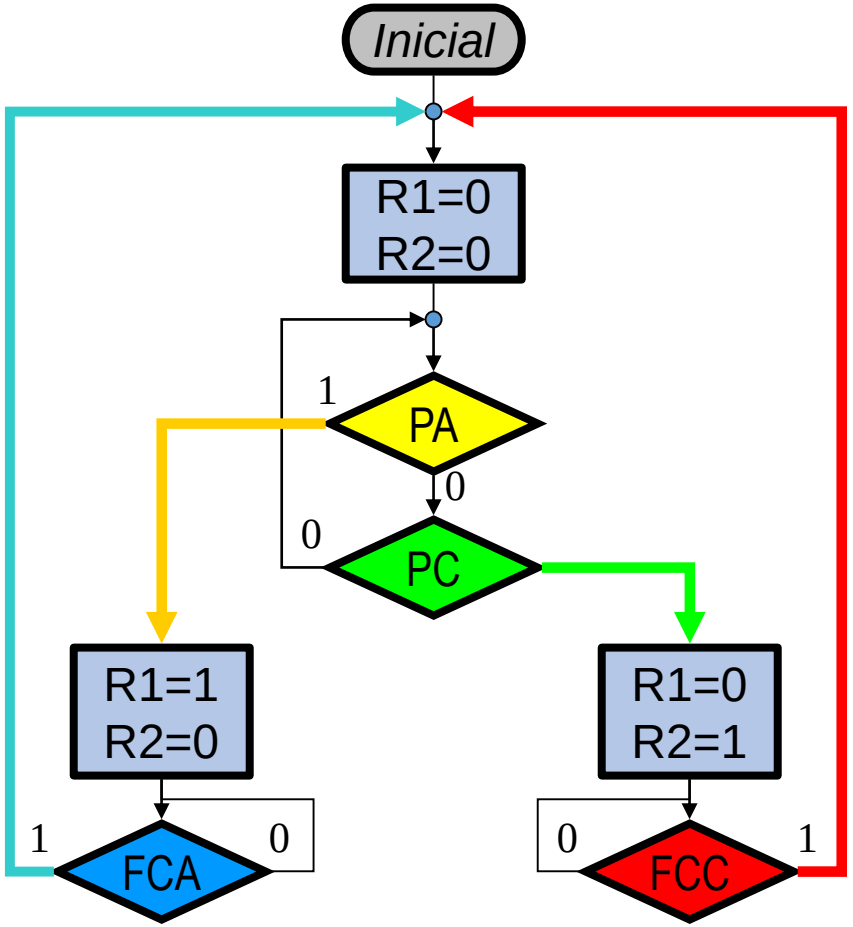
Relación entre Diagramas





En el Diagrama de Estado (izquierda) se ha coloreado las flechas de las transiciones, que son justamente condiciones o eventos que producen el "salto" de un estado a otro, y se puede observar en el diagrama de flujo (derecha) que estas condiciones corresponden a las bifurcaciones o saltos condicionales (rombos) del diagrama de flujo.

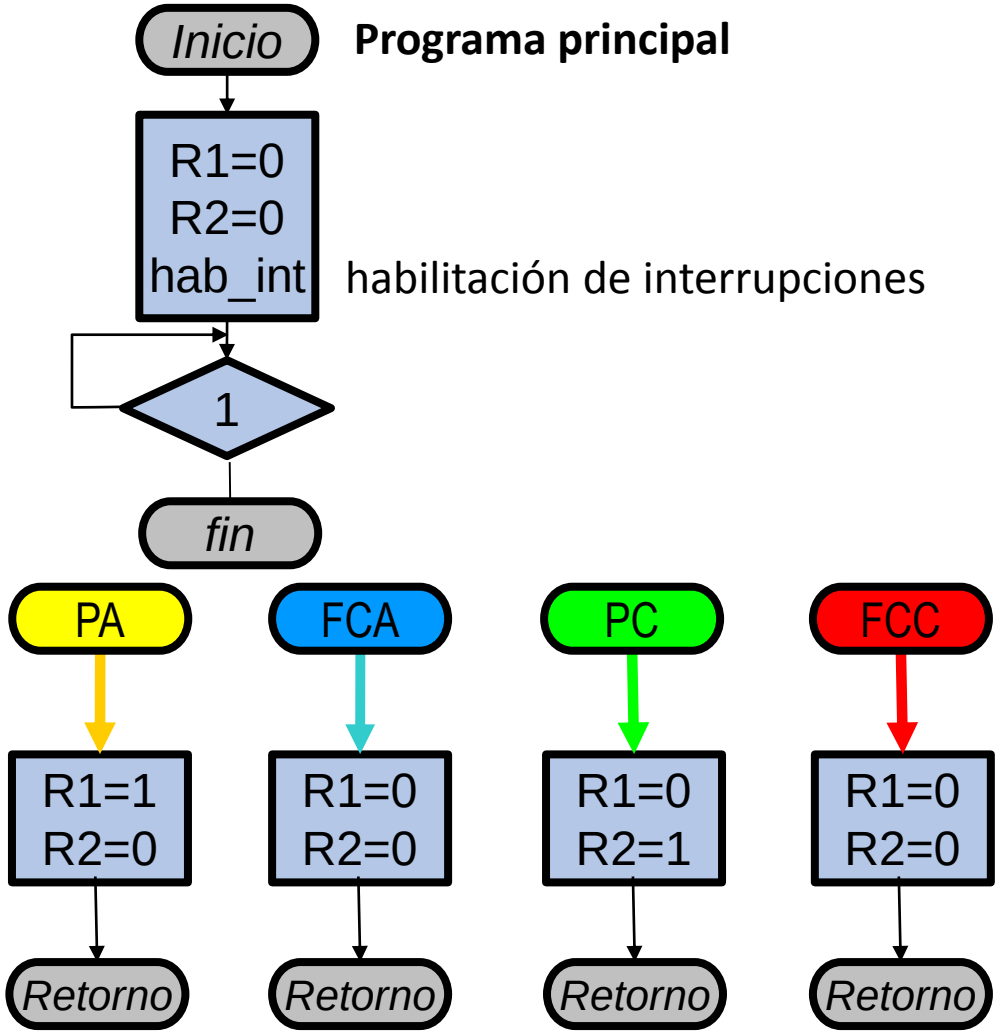
Del diagrama de flujo a la Codificación




```
REPOSO:     R1=0
            R2=0
REPOSO2:    IF PA=1 THEN
            GOTO      ABRE
            END IF
            IF PC=1 THEN
            GOTO      CIERRA
            END IF
            GOTO REPOSO2
ABRE:       R1=1
            R2=0
ABRE2:      IF FCA=1 THEN
            GOTO REPOSO
            END IF
            GOTO ABRE2
CIERRA:     R1=0
            R2=1
CIERRA2:    IF FCC=1 THEN
            GOTO REPOSO
            END IF
            GOTO CIERRA2
```

Finalmente, vemos que un diagrama de flujo puede pasarse de manera simple a un programa en lenguaje de alto nivel.

Uso de interrupciones de hardware




```
INICIO:     R1=0
            R2=0
            ENA
            WHILE 1

INT_PA:     R1=1
            R2=0
            RETFIE

INT_PC      R1=0
            R2=1
            RETFIE

INT_FCA:    R1=0
            R2=0
            RETFIE

INT_FCC:    R1=0
            R2=0
            RETFIE
```

Programa principal

Inicialización o Setup habilita interrupciones (pseudocódigo) bucle infinito)

Rutina de servicio de interrupción por PA (RETFIE: pseudocódigo para retorno de interrupción)

Rutina de servicio de interrupción por PC

Rutina de servicio de interrupción por FCA

Rutina de servicio de interrupción por FCC

El código anterior está lleno de verificaciones periódicas de las entradas (***polling***) y saltos condicionales. Las interrupciones de hardware permiten liberarse de esto. Cuando se produce el evento, automáticamente el programa salta a una subrutina específica. Esto simplifica el diagrama de flujo. Hay que habilitarlas con instrucciones específicas.

NOTAS

Esta codificación es muy rudimentaria, solamente con el propósito de ilustrar un modo de pasar del diagrama de flujo a un programa. Si bien puede funcionar, hace abuso de saltos (GOTO),lo que es considerado una mala programación.

Para los objetivos de la asignatura, y del trabajo integrador en particular, es suficiente con poder realizar un Diagrama de Estados (o Diagrama de Flujo) a partir de la descripción del funcionamiento.

Por otra parte, en muchos automatismos reales, haciendo uso de interrupciones de hardware, se facilita la codificación porque no es necesario realizar la verificación (***polling***) del estado de las entradas.

```
REPOSO:     R1=0
            R2=0
REPOSO2:    IF PA=1 THEN
            GOTO ABRE
            END IF
            IF PC=1 THEN
            GOTO CIERRA
            END IF
            GOTO REPOSO2
ABRE:       R1=1
            R2=0
ABRE2:      IF FCA=1 THEN
            GOTO REPOSO
            END IF
            GOTO ABRE2
CIERRA:     R1=0
            R2=1
CIERRA2:    IF FCC=1 THEN
            GOTO REPOSO
            END IF
            GOTO CIERRA2
```