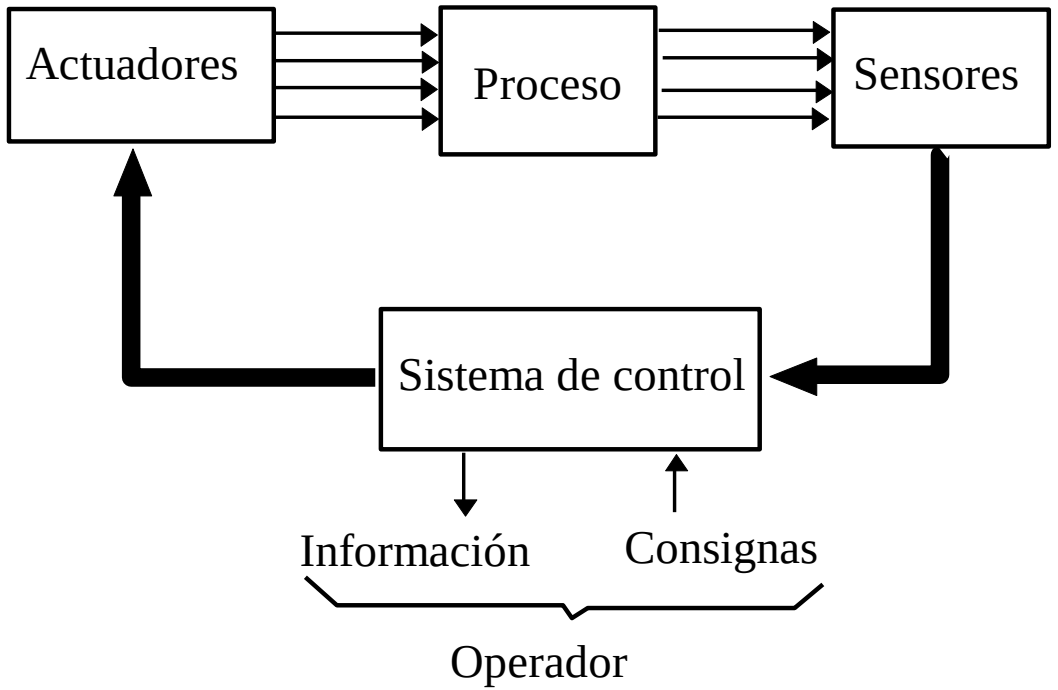
Introducción a Arduino

Ignacio Peñarrocha Alós

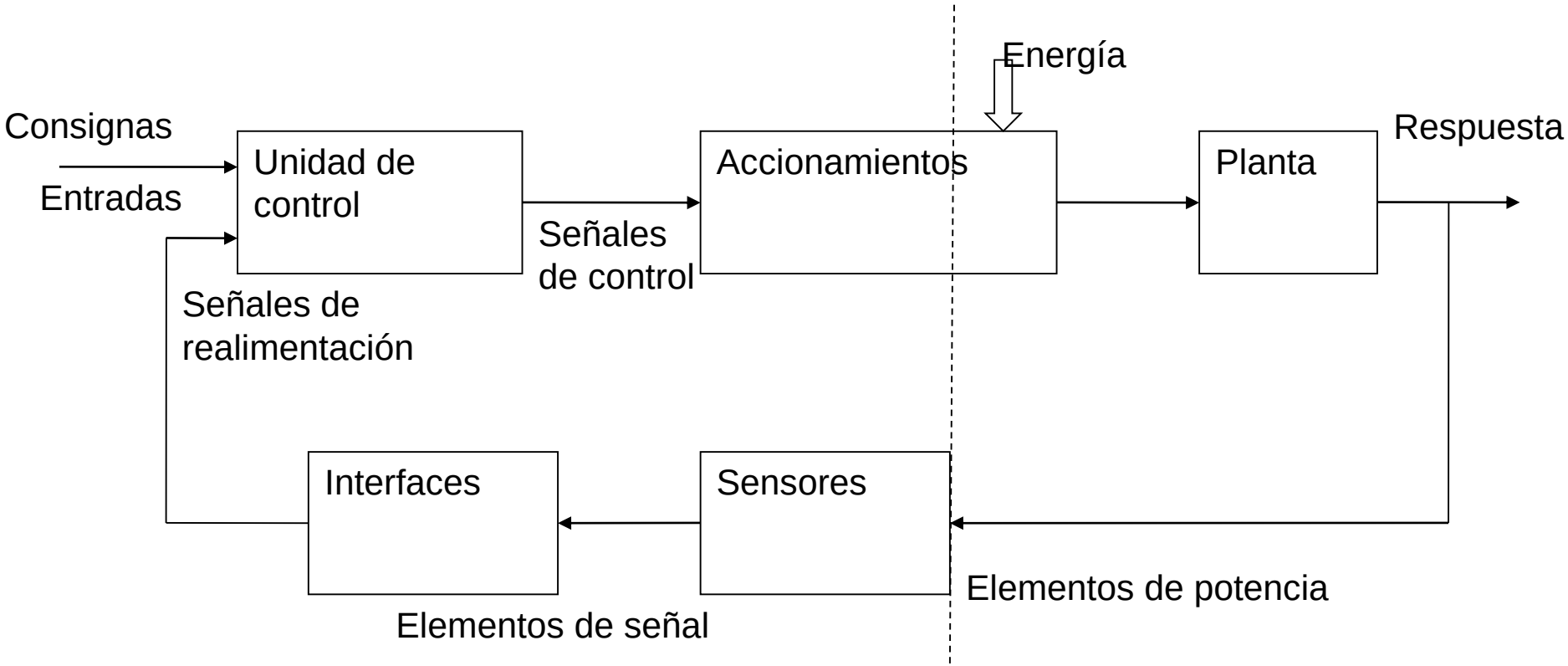
Definiciones

Sistema automatizado

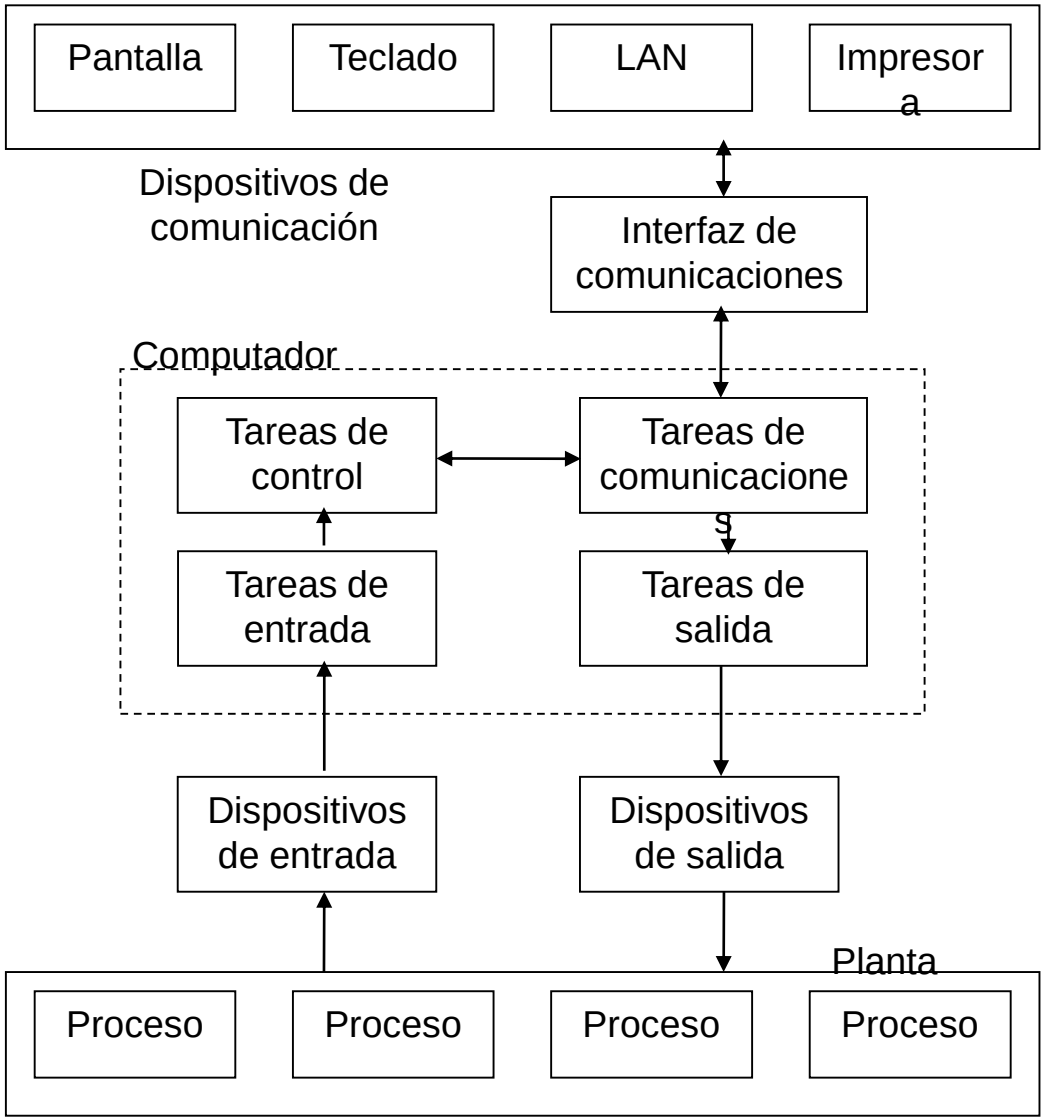


Definiciones

Sistema de control digital



Definiciones



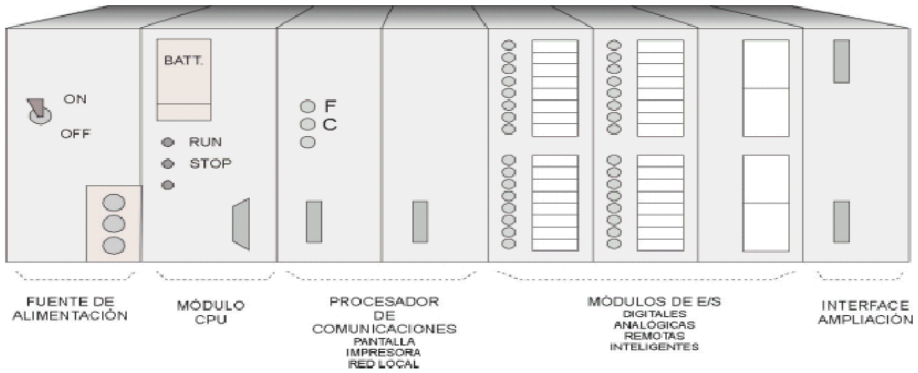
Plataformas para la implementación de algoritmos de control

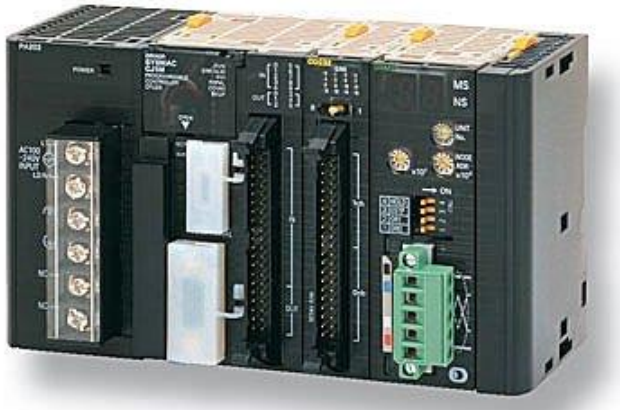
Plataformas:

- Autómatas programables (PLCs)
- Microcontroladores de propósito general
- Microcontroladores basados en DSP
- PC con tarjeta de adquisición de datos

Plataformas

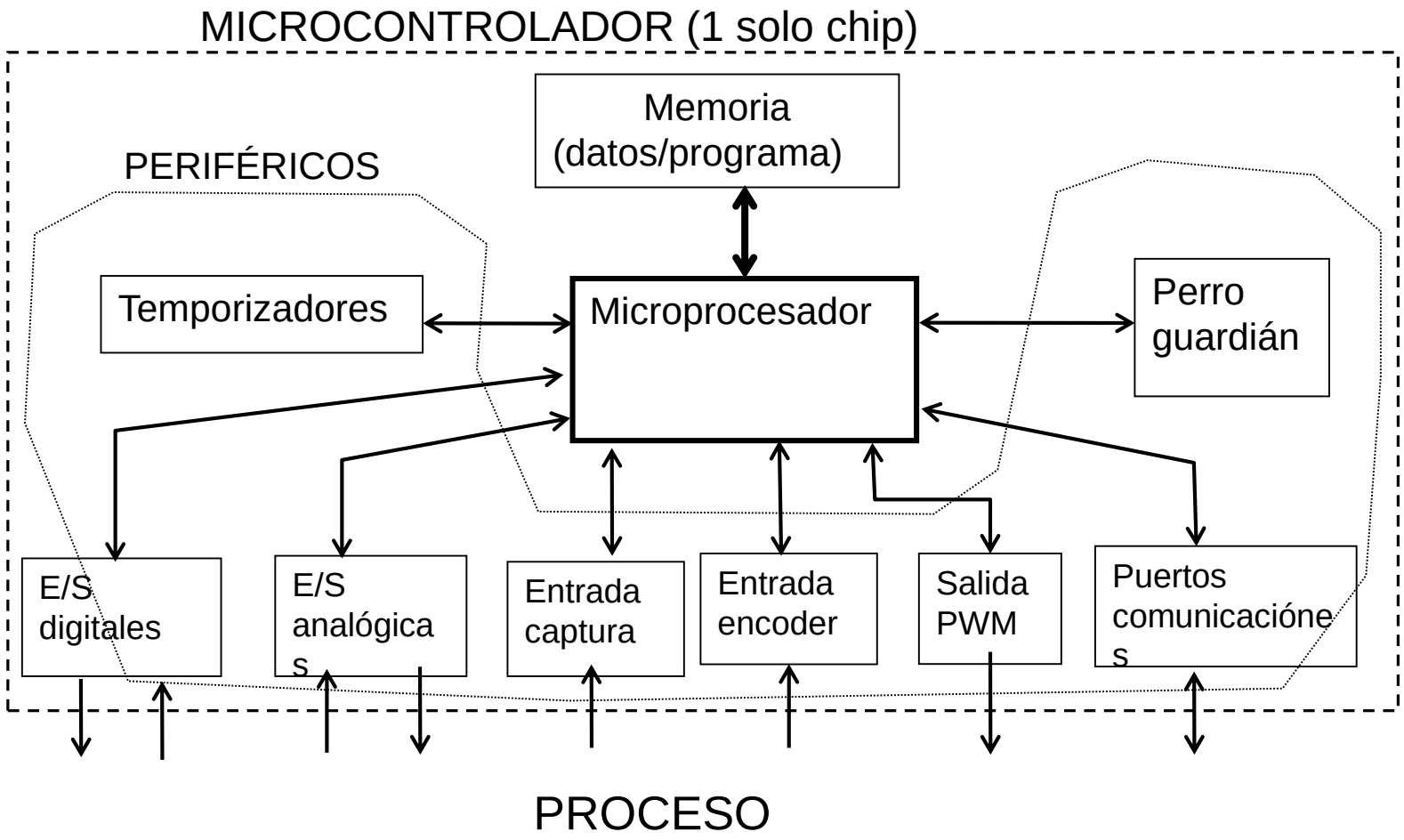
PLCs





Plataformas

Microcontroladores



Plataformas

Microcontroladores. Selección

Factores a tener en cuenta

- Coste
- Características de la aplicación
  - Procesamiento de datos
  - Entradas/Salidas
  - Consumo
  - Memoria
  - Ancho de palabra
  - Diseño de placa

Plataformas

PCs industriales

Lenguajes para la programación de algoritmos de control

- Lenguajes de bajo nivel: Ensamblador
- Lenguajes de alto nivel: C, Java
- Lenguajes de la norma IEC 61131
- Lenguajes gráficos: Labview, Simulink
- Lenguajes interpretados: Matlab, Python

- Otros lenguajes para monitorización, acceso a base de datos y tareas generales: Javascript, Java, SQL, etc

Lenguajes

Microcontroladores. Lenguajes

Ensamblador

- Nemotécnicos del código máquina
- Etiquetas para datos y direcciones de programa
- Se obtiene el código máquina

Lenguajes de alto nivel (C, Compiladores)

- Desarrollo rápido de programas
- No se conocen las instrucciones finales
- Compilador: C > ensamblador > c. máquina

Lenguajes

Lenguaje C

```
boolean variable_binaria_1, variable_binaria_2;
int variable_entera_1, variable_entera_2;
float variable_real_1, variable_real_3;
void setup(){
    instruccions;
}
void loop(){
    isntrucciones;
    if(condición){
        instrucciones;
    }
}
```

Derivados: Processing/Arduino

Lenguaje Java

```
public class Parametros {
    public static void main(String[] args) {
        int var1;
        Objeto obj;
        obj = new Objeto(1);
        var1 = 2;
        System.out.println("Valor del objeto= "+obj.variable);
        System.out.println("Valor de la variable ="+var1);
        modifica(var1,obj);
        System.out.println("-Despues de llamar a modifica()-");
        System.out.println("Valor del Objeto= "+obj.variable);
        System.out.println("Valor de la variable= "+var1);

    }
    static void modifica(int vv, Objeto oo){
        vv++;
        oo.variable++;
    }
```

Lenguajes

Lenguajes norma IEC 61131-3

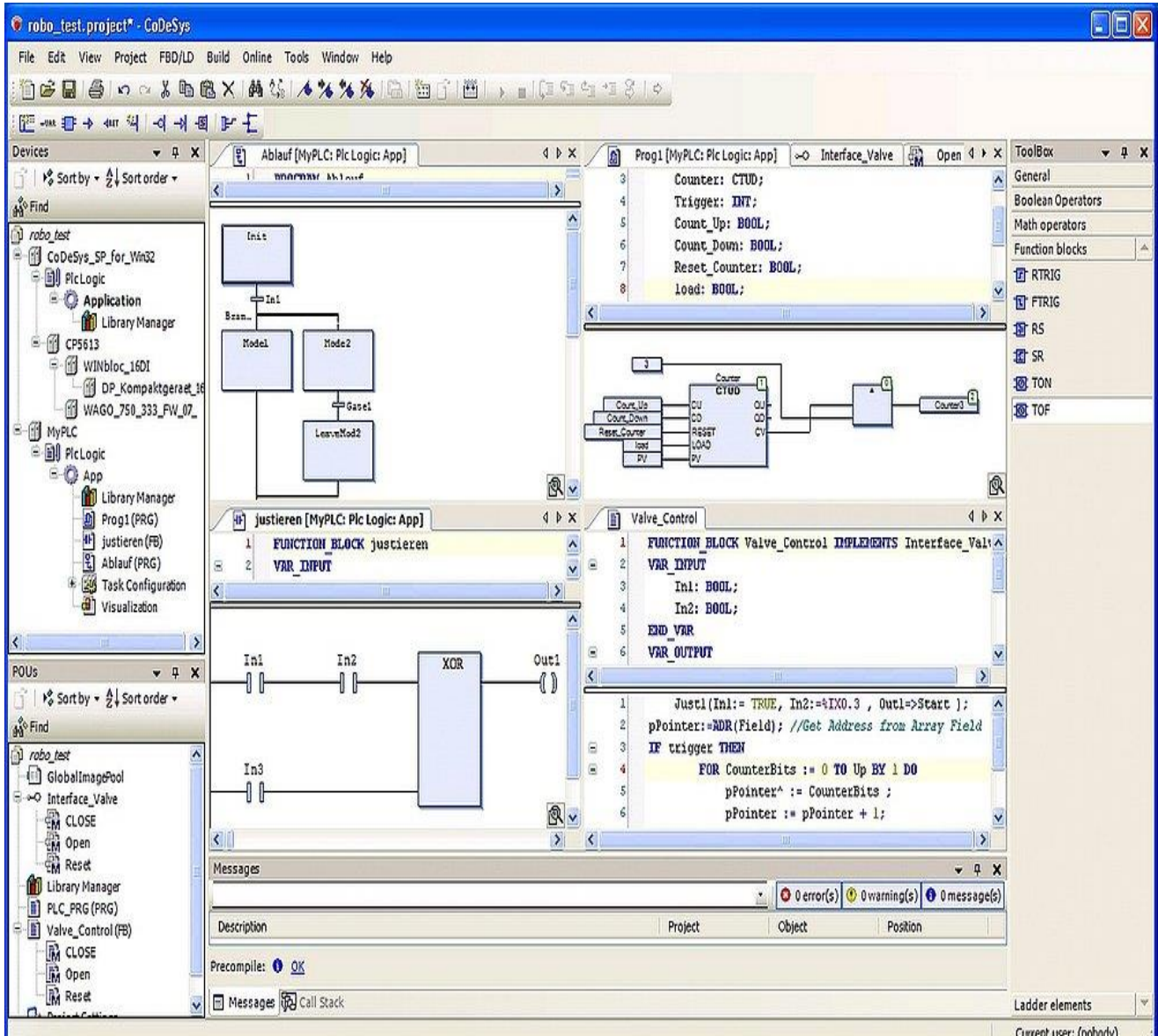
LD - Ladder Diagram

FBD - Function Block Diagram

ST - Structured Text

IL - Instruction List

SFC - Sequential Function Chart

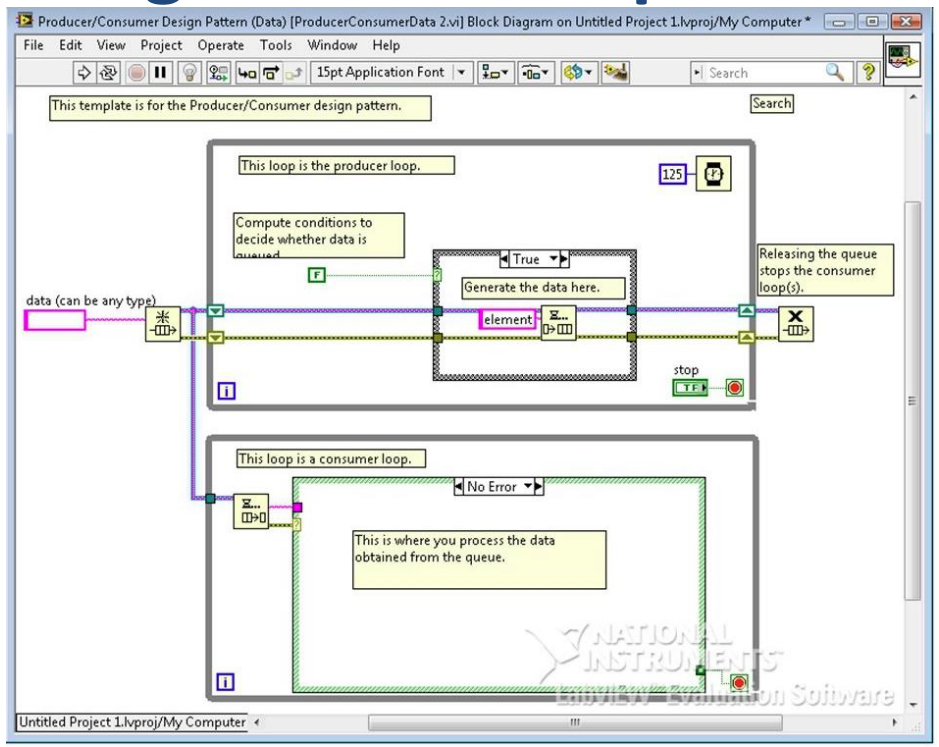


Lenguajes

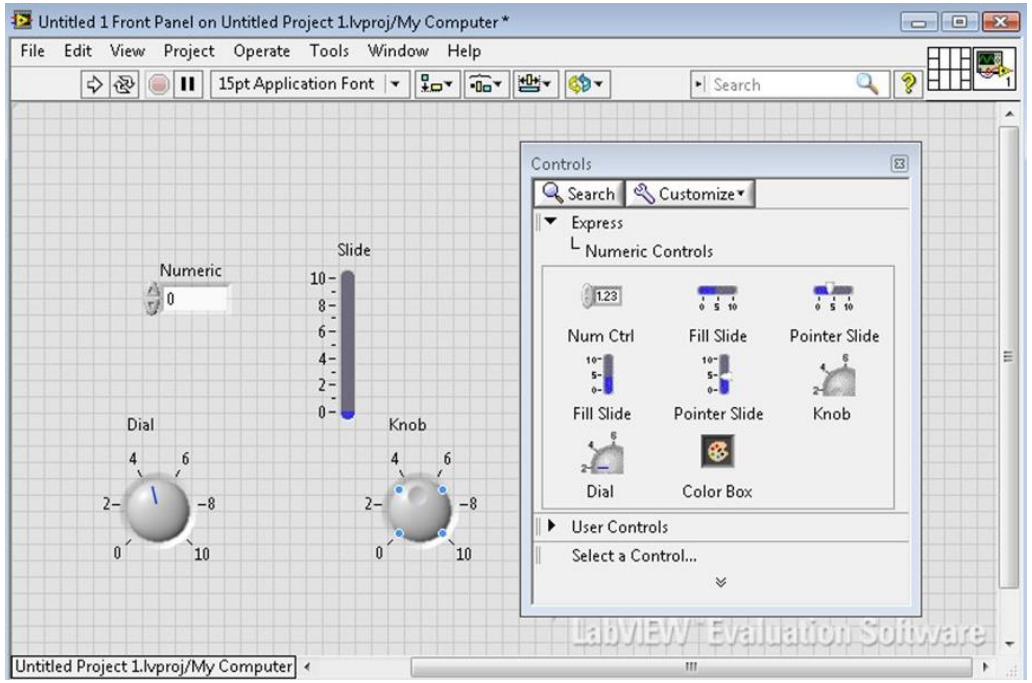
Lenguajes gráficos. Labview.

Programación simultánea de 2 aplicaciones que interactúan:

Diagrama de bloques

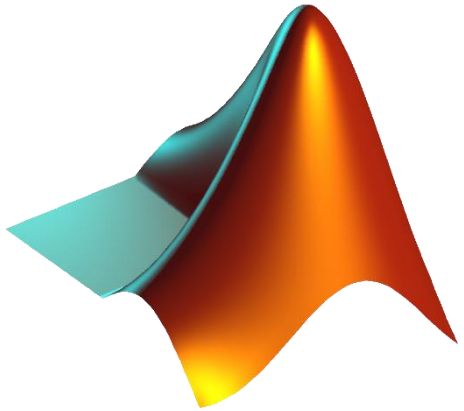


Panel frontal



Lenguajes

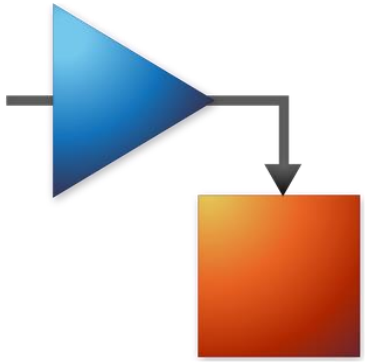
Matlab



Python

PYTHON

Lenguajes interpretados muy similares. Para desarrollo rápido de aplicaciones de muy alto nivel. Multitud de bibliotecas (packages, toolboxes) disponibles.



Simulink. Paquete de programación gráfica dentro de Matlab para simulación de control de procesos

Lenguajes

Otros lenguajes de interés:

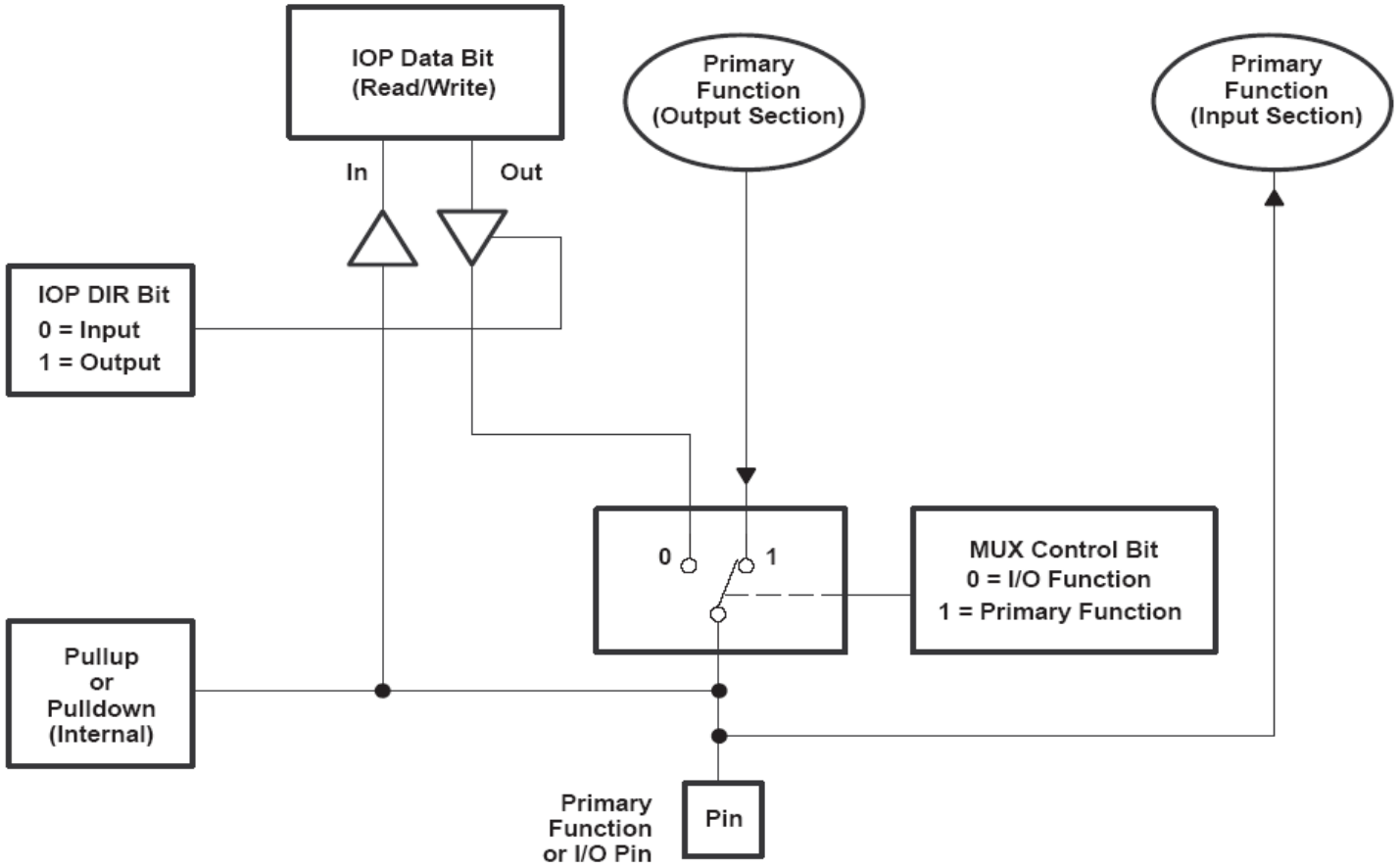
HTML – Creación de páginas web, orientado a interfaz usuario

Javascript – Contenido con cálculo en páginas web

```
<script>

  document.write("<b>Uso de variables</b><br><br>");

  // To start we will create 2 variables:
  var variable1 = 'Hello ';
  var variable2 = 'World!<br><br>'; // <br> means line break in html

  var n = 9;
  var x = n*n;

  // Create a loop of 10 elements.
  // Variable "i" starts with value 1 and while i<=10 it will increment 1 (i=i+1)
  for (var i=1; i<=10; i=i+1) {
    document.write(i); // Print the current "i" number
    // Print a comma followed by a space if i < 10
    if (i<10) {
      document.write(", ");
    }
  }
  console.log(x);
</script>
```

Periféricos

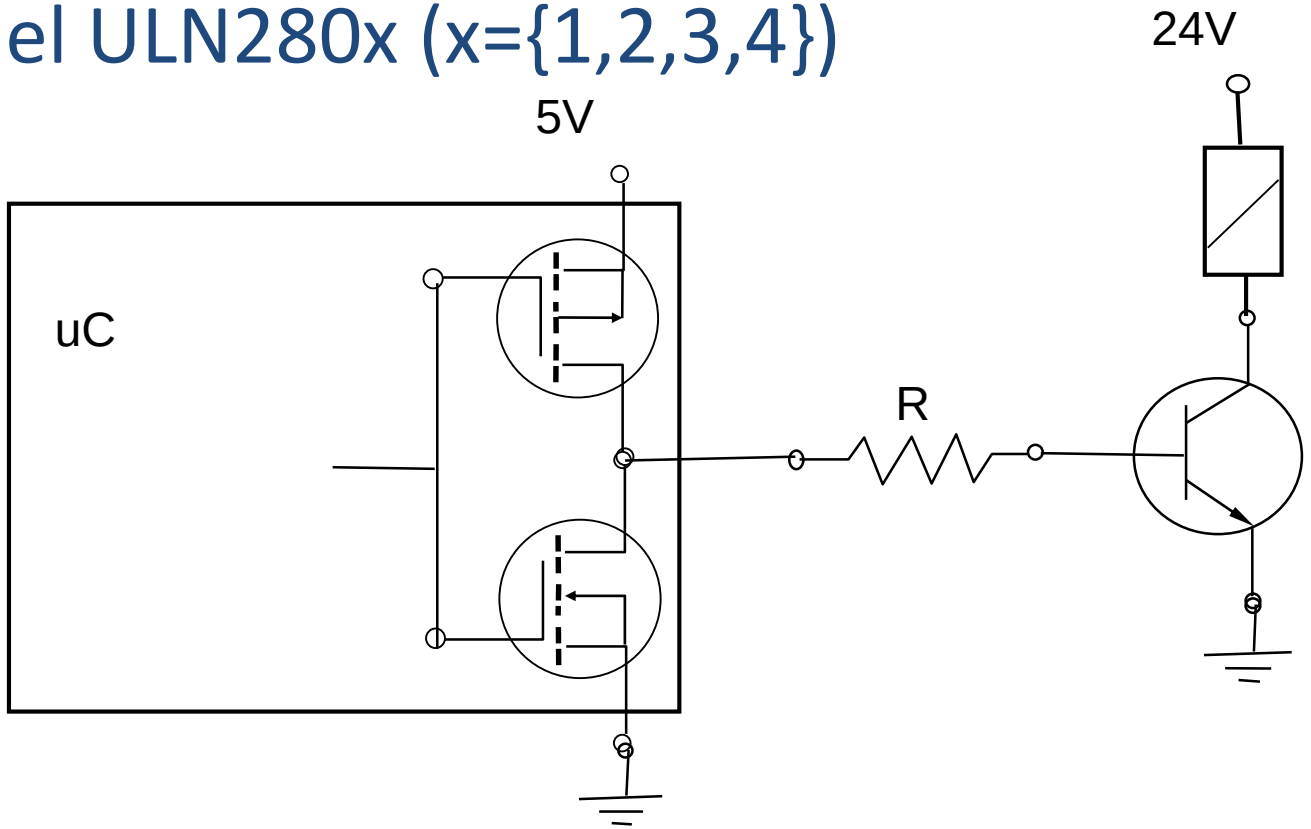
Entradas y salidas digitales



Periféricos

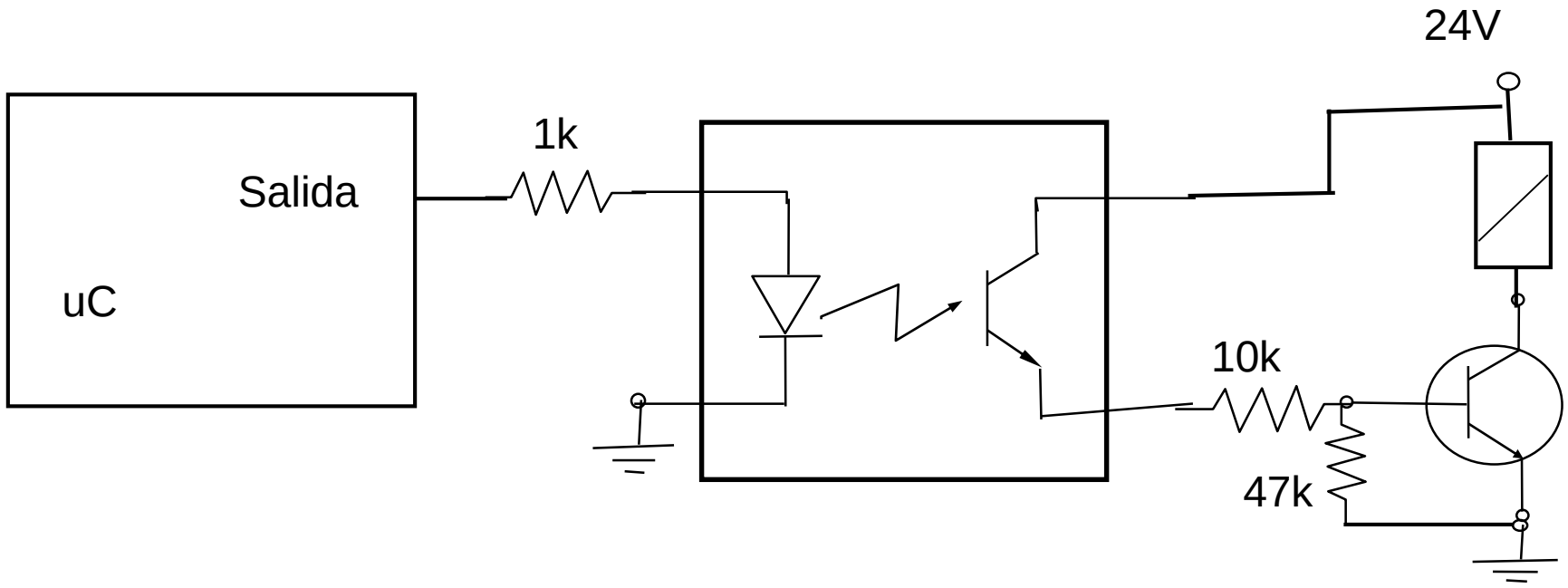
Salidas digitales $(R > \frac{5}{I_{max}})$.

ULN200x y el ULN280x (x={1,2,3,4})



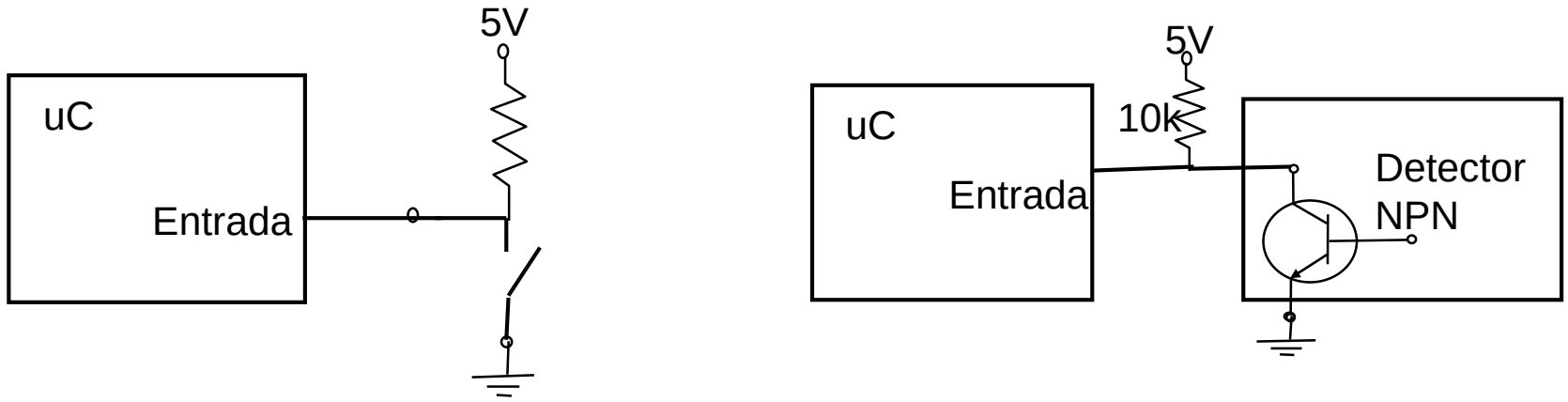
Periféricos

Salidas digitales. Salidas optoacopladas



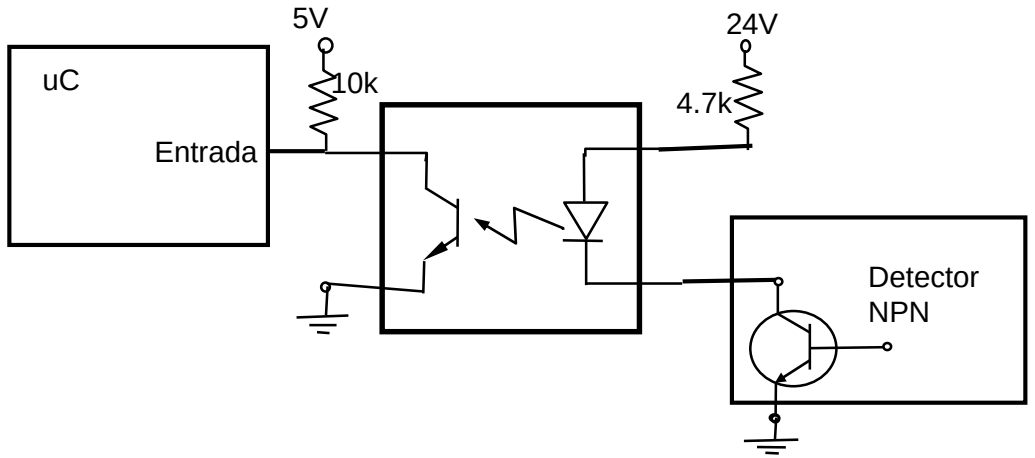
Periféricos

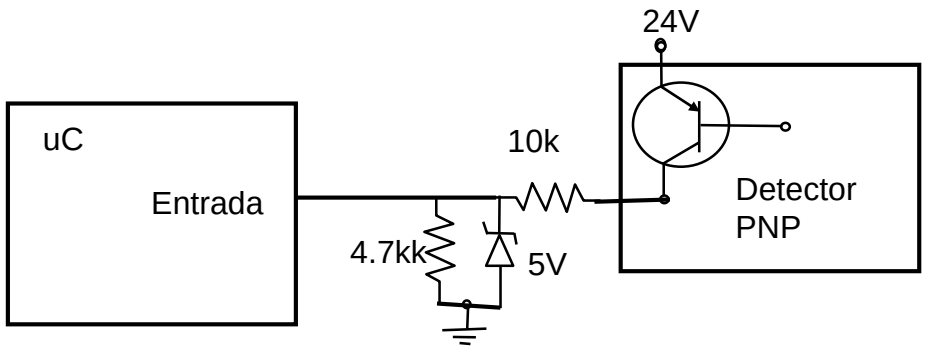
Entradas digitales.



Periféricos

Entradas digitales. Entradas optoacopladas





Periféricos

Entradas analógicas (ADC).

- Usan un convertidor ADC (analógico-digital)
- Se definen por:
  - Rango de entrada
  - Resolución
  - Precisión
  - Función de transferencia
  - Velocidad de conversión
    - máxima frecuencia de muestreo

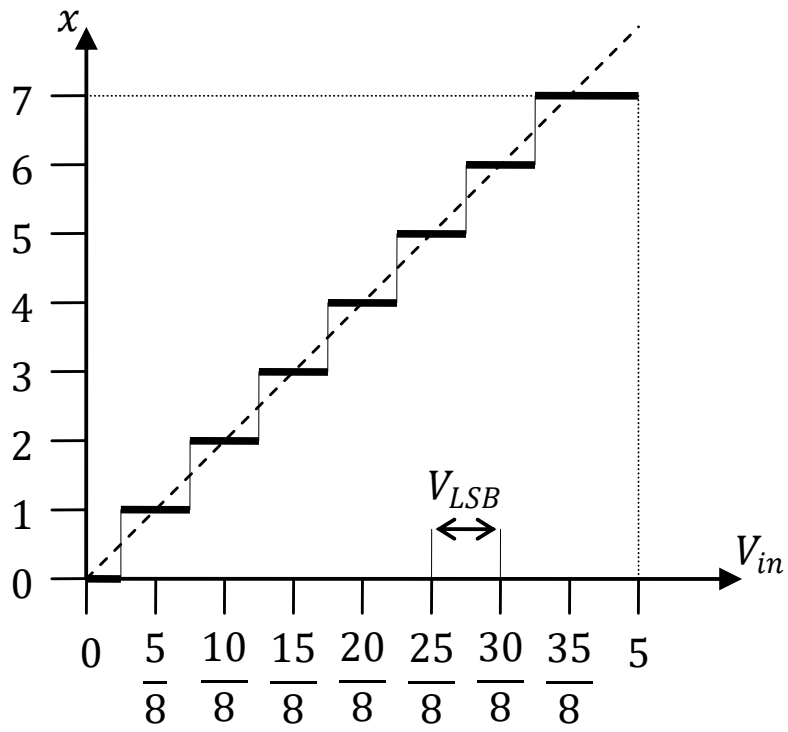
Periféricos

Entradas analógicas (ADC).

- Función de transferencia
- Rango: : $V_{in} \in [R^-, R^+]$
- Resolución: $n\ bits$
- Precisión

$$V_{LSB} = \frac{R^+ - R^-}{2^n}$$

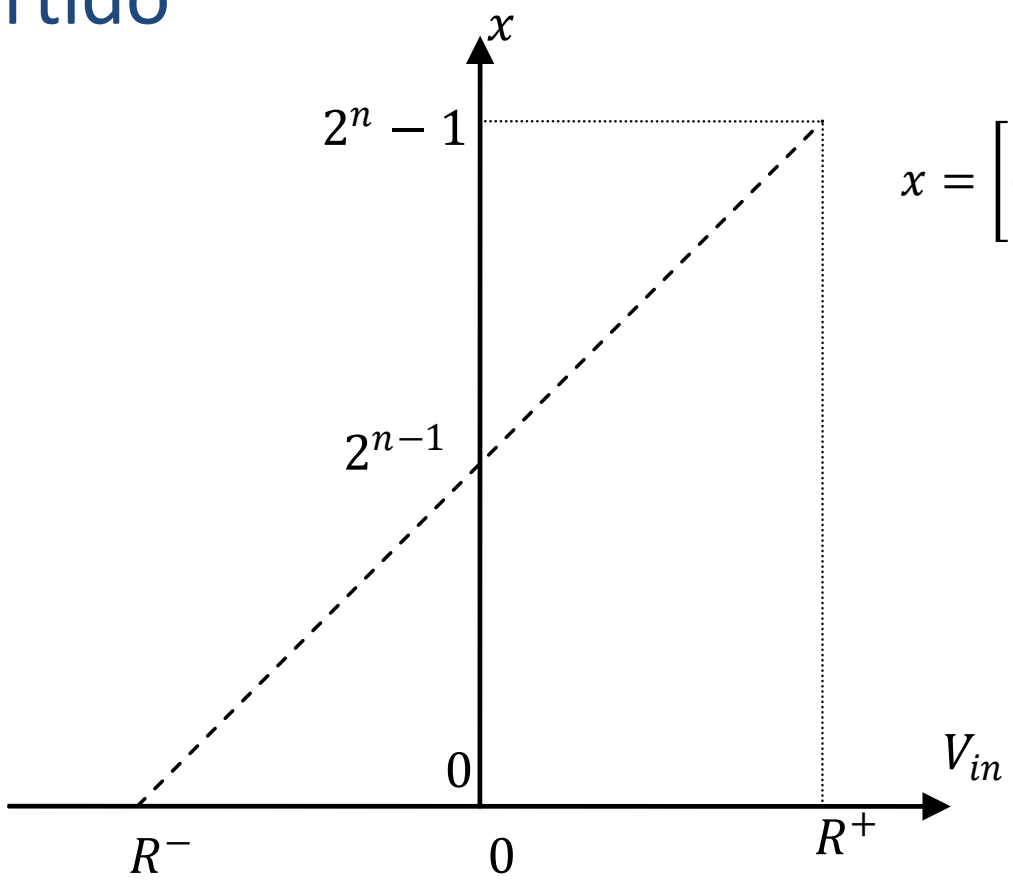
- Máxima entrada: $V_{in,máx} = R^+ - V_{LSB}$



Periféricos

Entradas analógicas (ADC).

Valor convertido



$$x = \left[(V_{in} - R^-) \frac{2^n - 1}{(R^+ - R^-) - V_{LSB}} \right]$$

Periféricos

Entradas analógicas (ADC).

- Resoluciones habituales

Resolución (bits)	Rango del valor convertido
8	[0, 255]
10	[10, 1023]
12	[0, 4095]

- Rango sensor: [-10, 10] V, ADC 12 bits. (V_{LSB} ≈0.005V)

Salida del sensor (V)	Valor digital convertido
-10.0	0
0.0	2047
0.0049	2048
+10.0	4095

Periféricos

Salidas analógicas (DAC).

- Usan un convertidor DAC (digital-analógico)
- Se definen por:
  - Rango de salida
  - Resolución
  - Precisión
  - Función de transferencia
  - Velocidad de conversión
    - máxima frecuencia de actualización

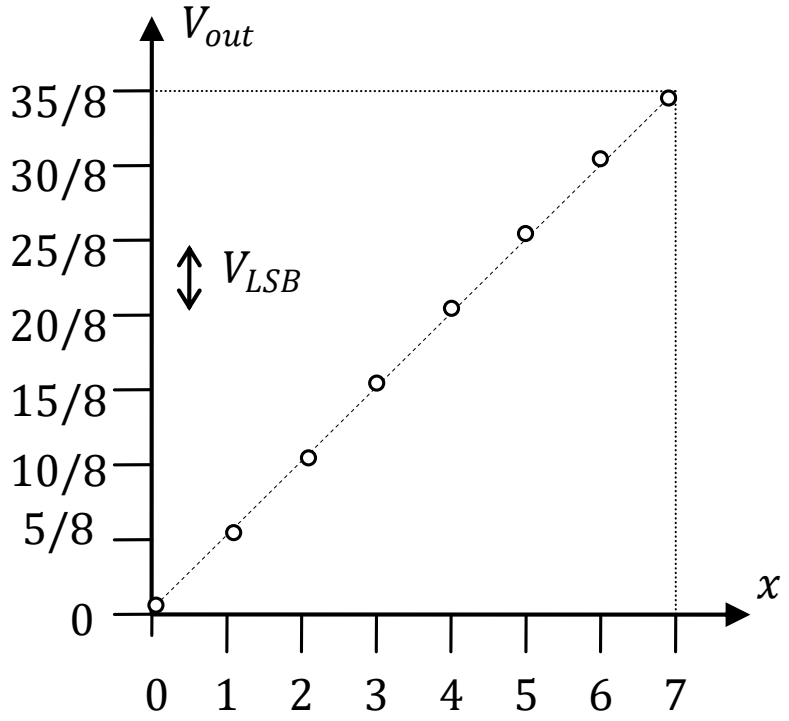
Periféricos

Salidas analógicas (DAC).

- Función de transferencia
- Rango: : $V_{out} \in [R^-, R^+]$
- Resolución: $n\ bits$
- Precisión

$$V_{LSB} = \frac{R^+ - R^-}{2^n}$$

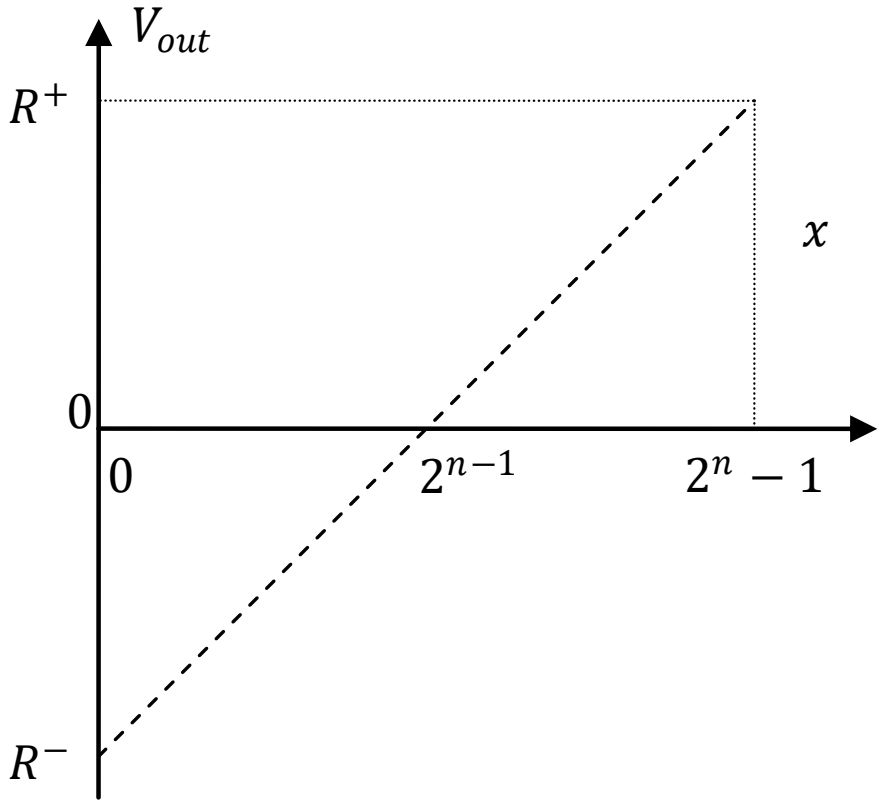
- Máxima salida: $V_{in,máx} = R^+ - V_{LSB}$



Periféricos

Salidas analógicas (DAC).

Valor convertido



$$V_{out} = V_{FE}(D_{n-1}2^{-1} + D_{n-2}2^{-2} + \cdots + D_1 2^{-n-1} + D_0 2^{-n})$$

Periféricos

Salidas analógicas (DAC).

- DAC: Rango [0, 10] V, res. 10 bits. ($V_{LSB} \approx 0.01$V)

Valor digital	Tensión de salida (V)
0	0,0
1	0,0099
511	5,0
1023	10,0

Periféricos

Problemas con la cuantización del ADC y del DAC.

- Cuando el rango del sensor o actuador no coincide con los rangos analógicos de los convertidores.
- Ejemplo:
  - Sensor: [1, 5] V
  - ADC de **12 bits**, rango: [-10, 10]V
  - Sólo se distinguen 4096*4/20=820 valores (≈ **10 bits**)
- Conclusión: Para aumentar la precisión:
  - Rango sensor ≃ Rango ADC
  - Rango actuador ≃ Rango DAC

Periféricos

Temporizadores.

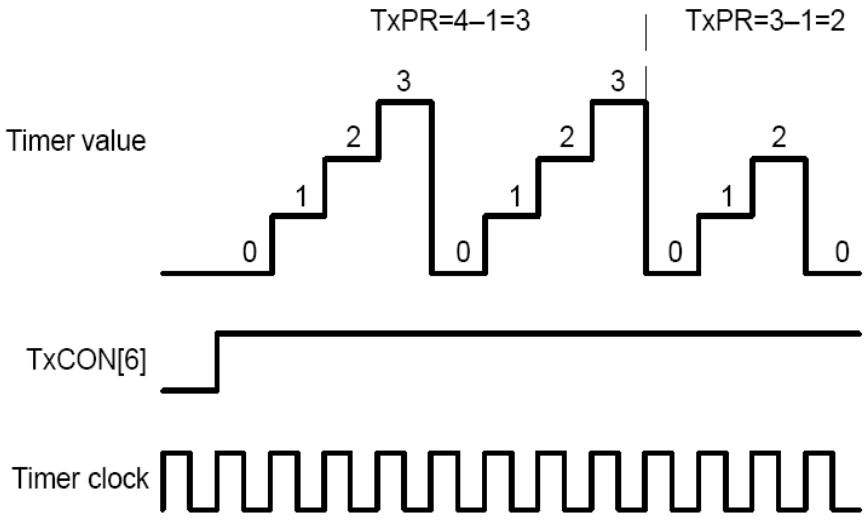
- Contador digital de pulsos de reloj
- Preescalado (se incrementa tras varios pulsos)
- Resolución: nº de bits n
- Máximo tiempo: $T_{máx} = \frac{2^n P}{F_{CLK}}$
- Para contar menos tiempo: $T = \frac{(T_{PR}+1)P}{F_{CLK}}$

Periféricos

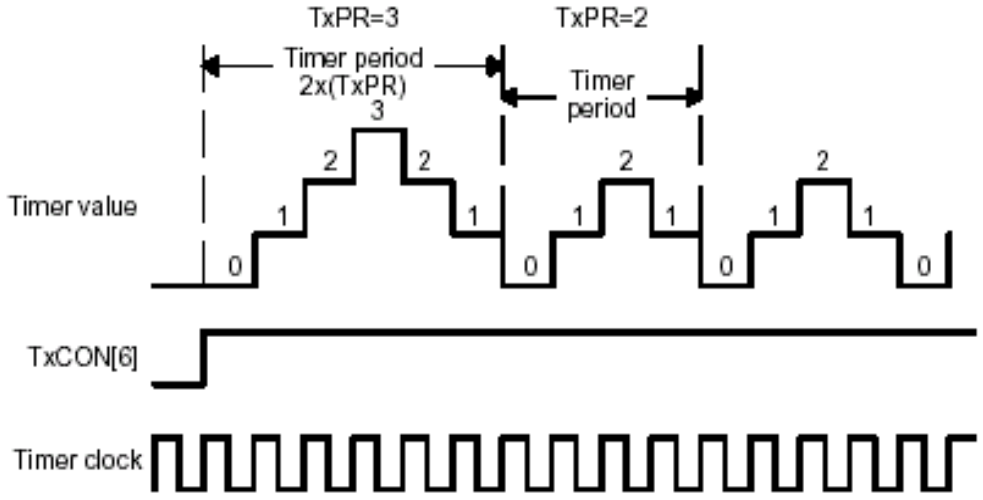
Temporizadores.

Modos de operación

Ascendente.



Asc.-descendente

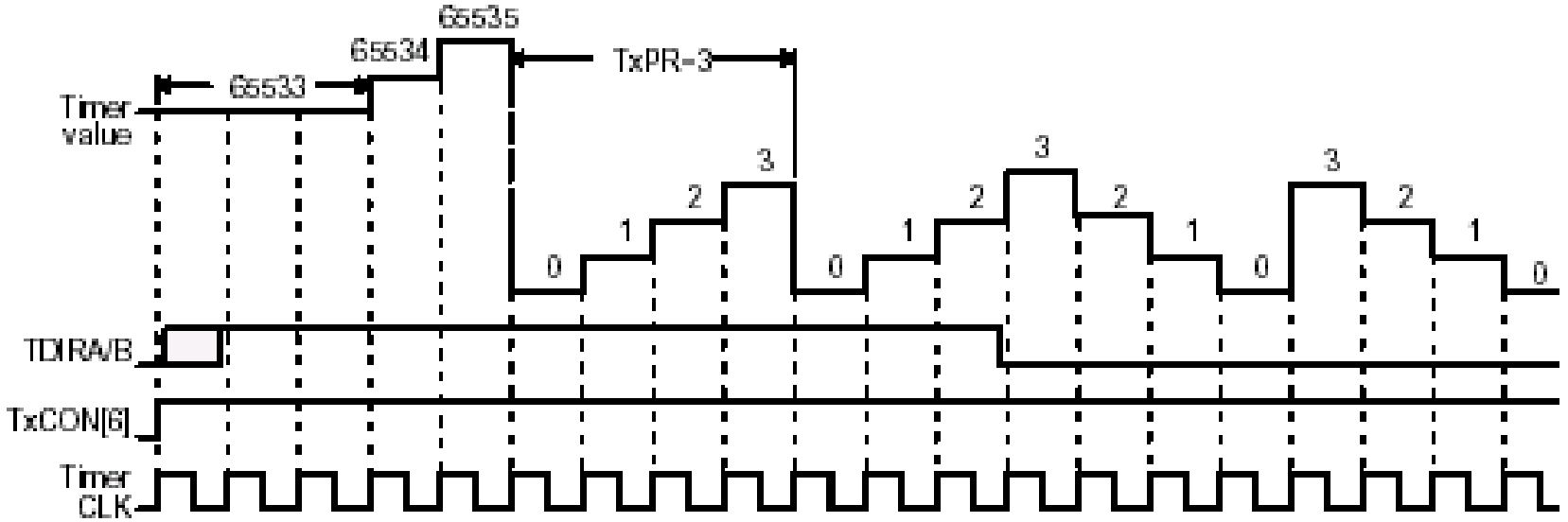


Periféricos

Temporizadores.

Modos de operación

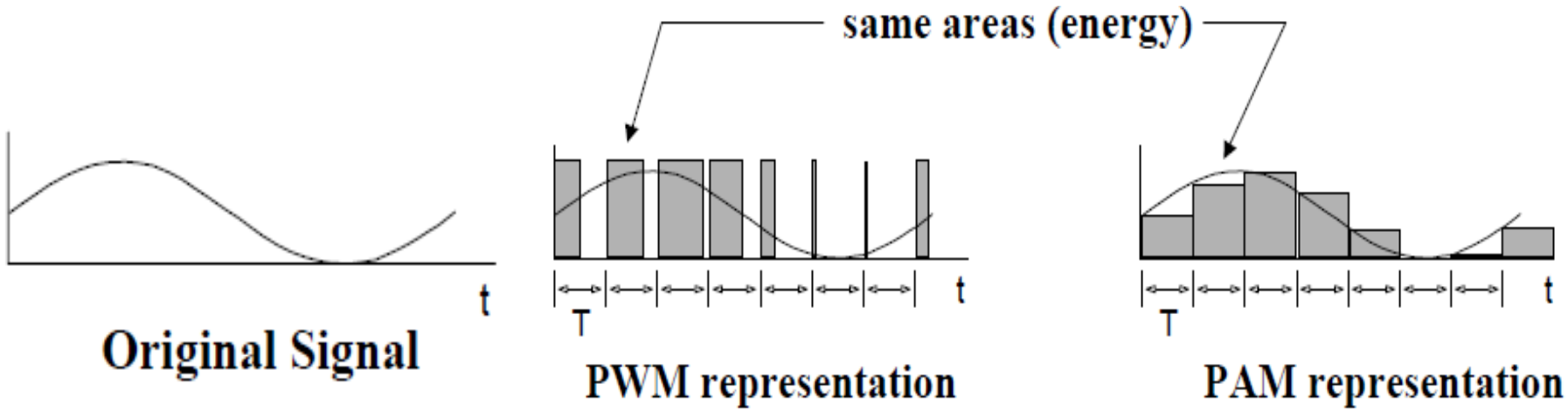
Con entrada de dirección



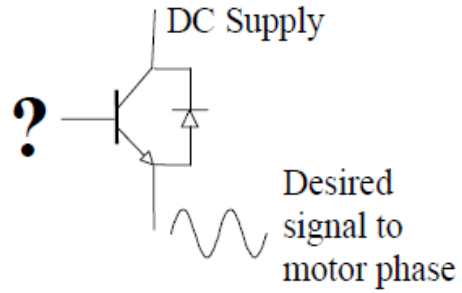
Periféricos

Salidas moduladas por ancho de pulso (PWM).

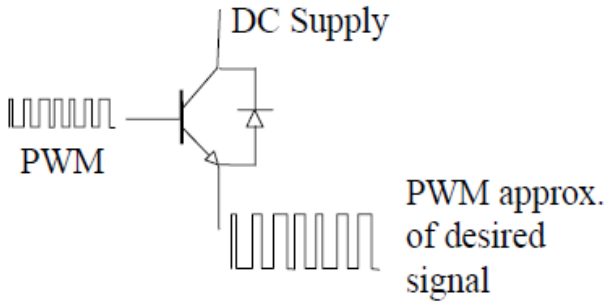
Continua PWM DAC



Original Signal PWM representation PAM representation



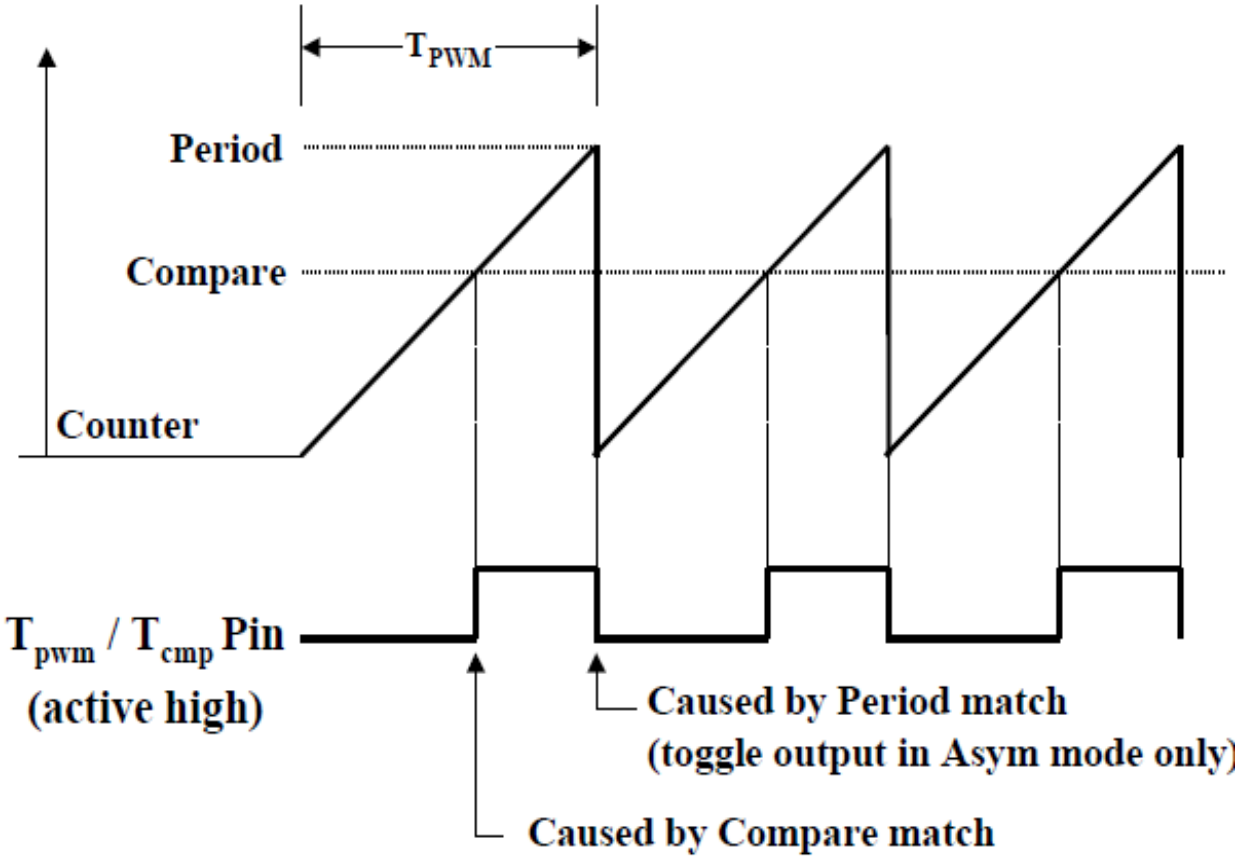
Unknown Gate Signal



Gate Signal Known with PWM

Periféricos

Salidas moduladas por ancho de pulso (PWM).



Periféricos

Salidas moduladas por ancho de pulso (PWM).

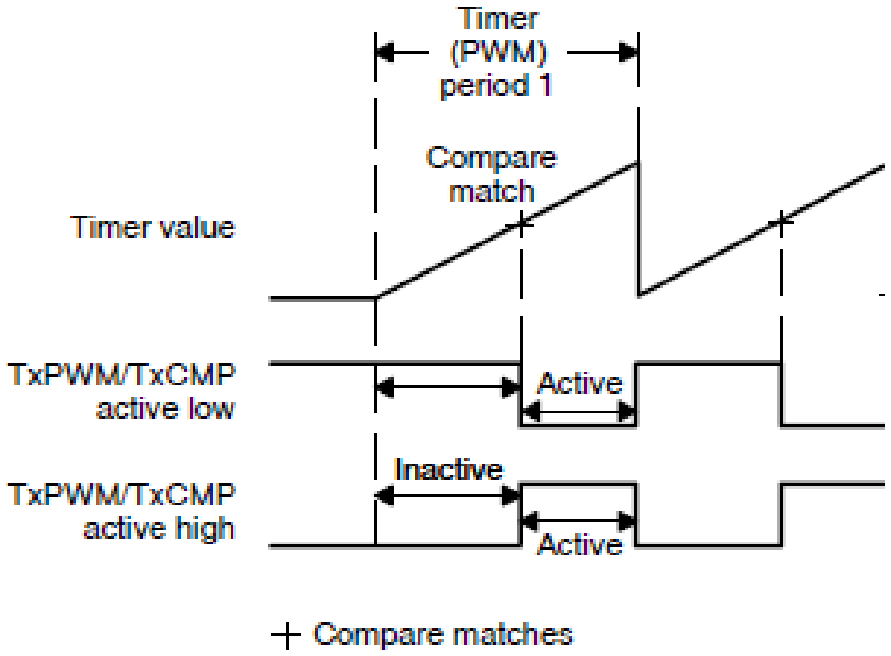
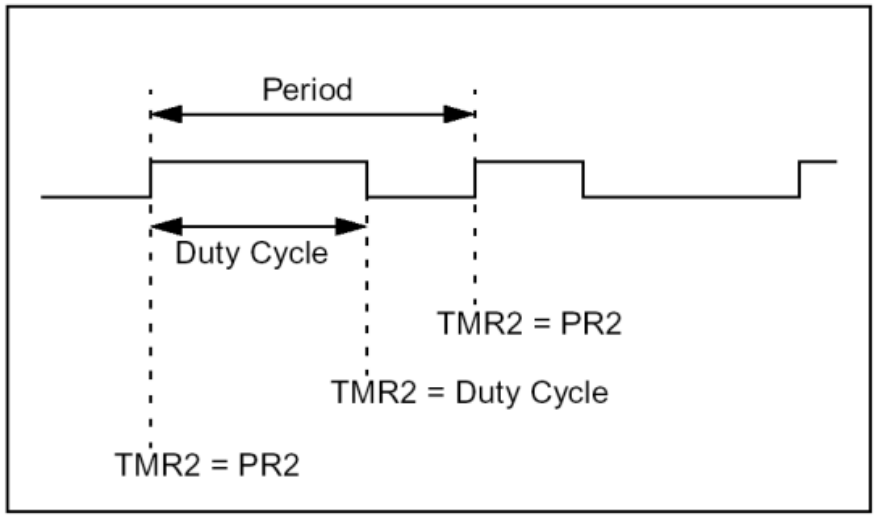


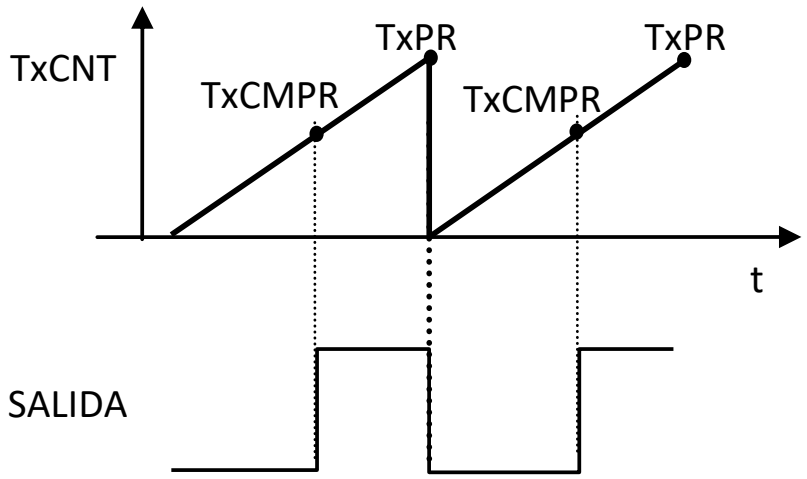
FIGURE 8-4: PWM OUTPUT



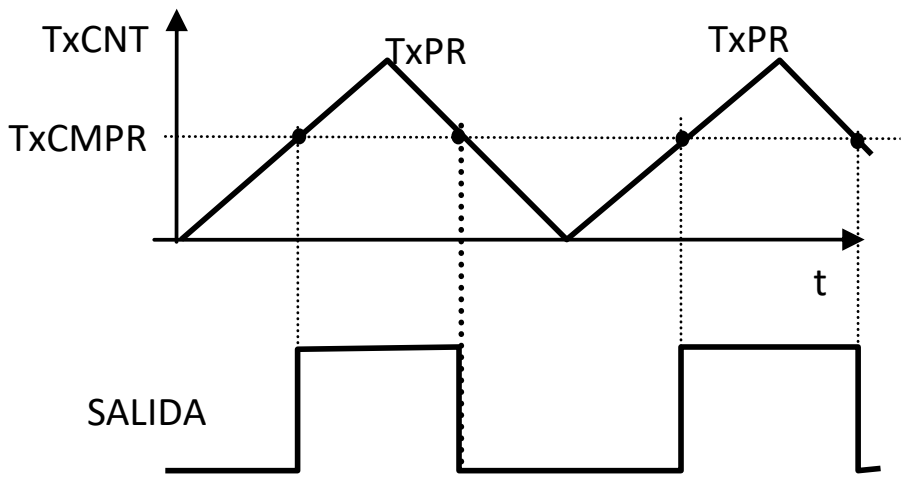
Periféricos

Salidas moduladas por ancho de pulso (PWM).

PWM ASIMÉTRICO

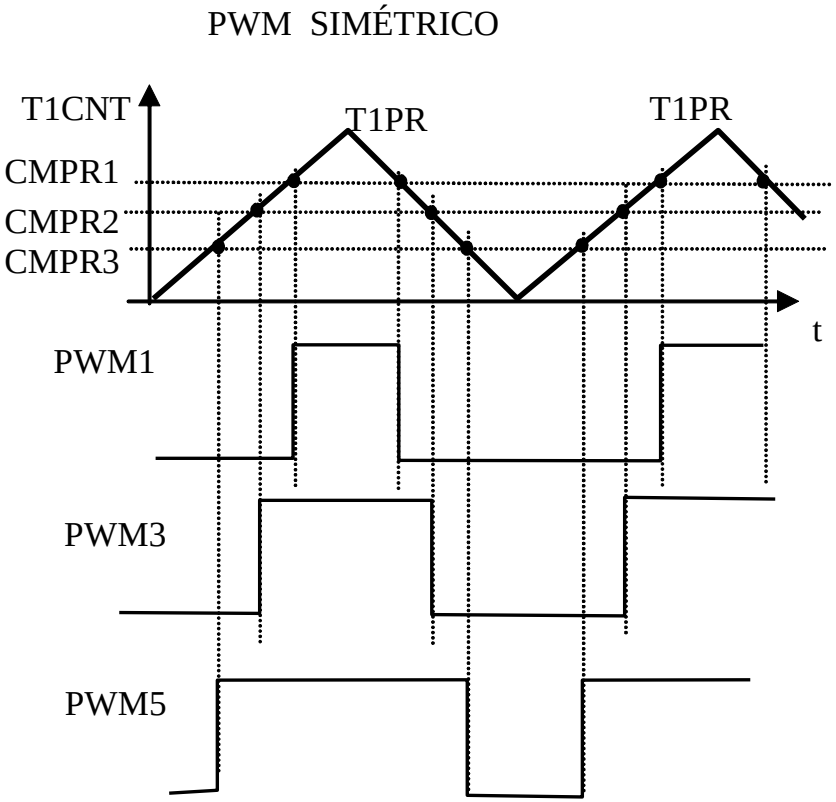


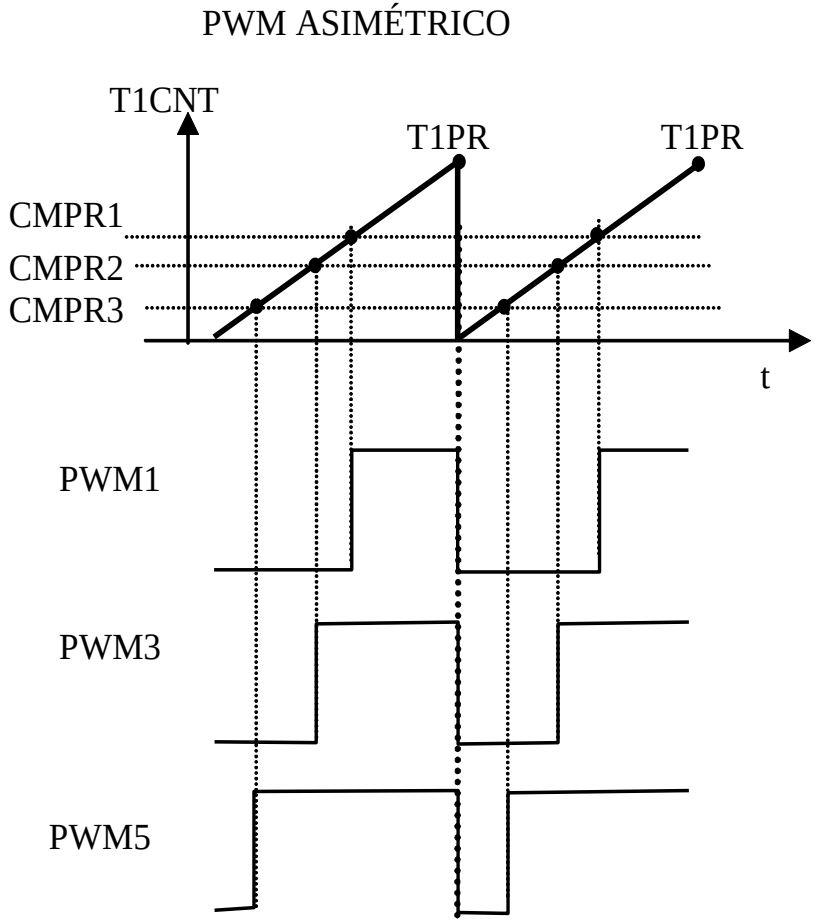
PWM SIMÉTRICO



Periféricos

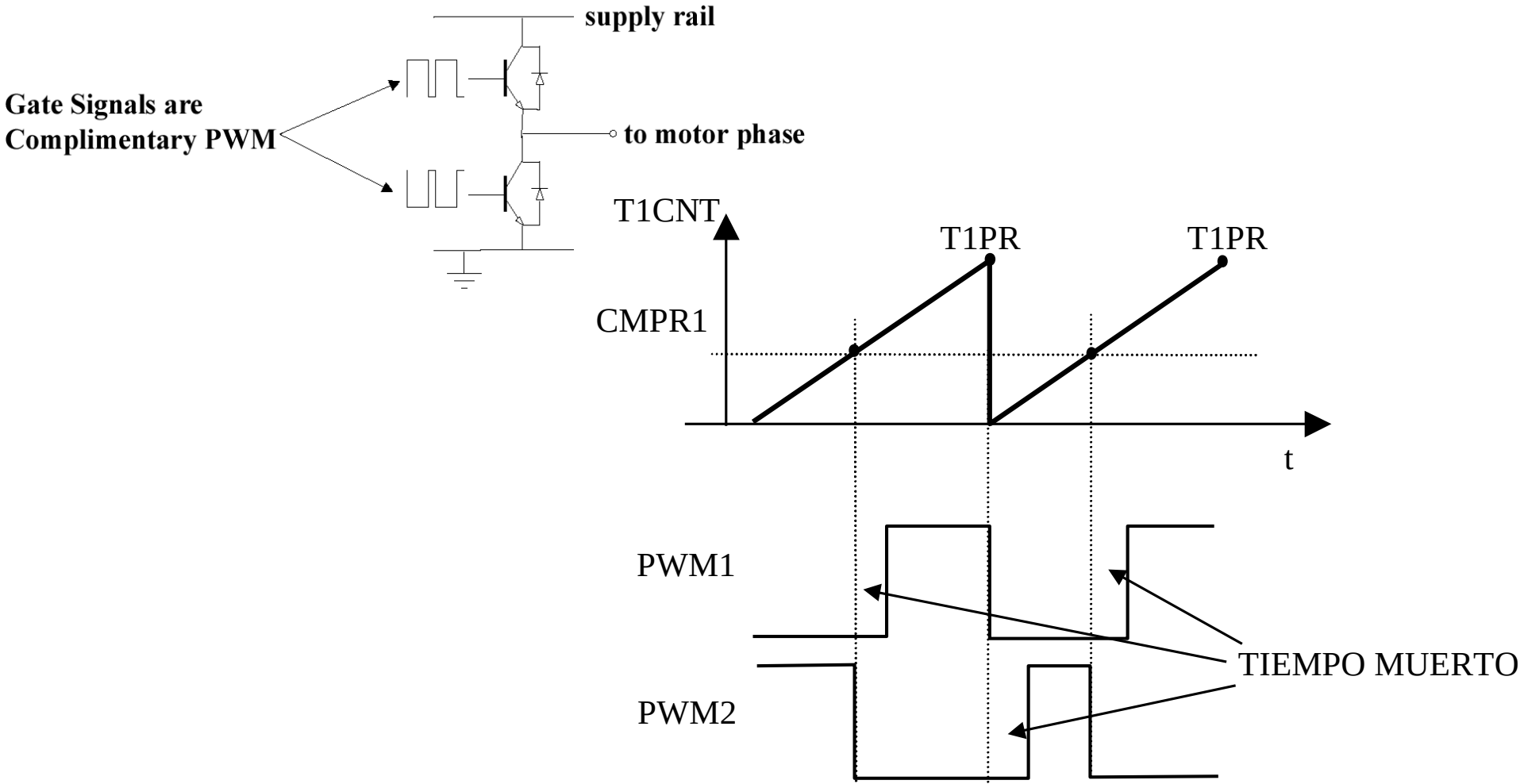
Unidad de comparación completa (6PWM).





Periféricos

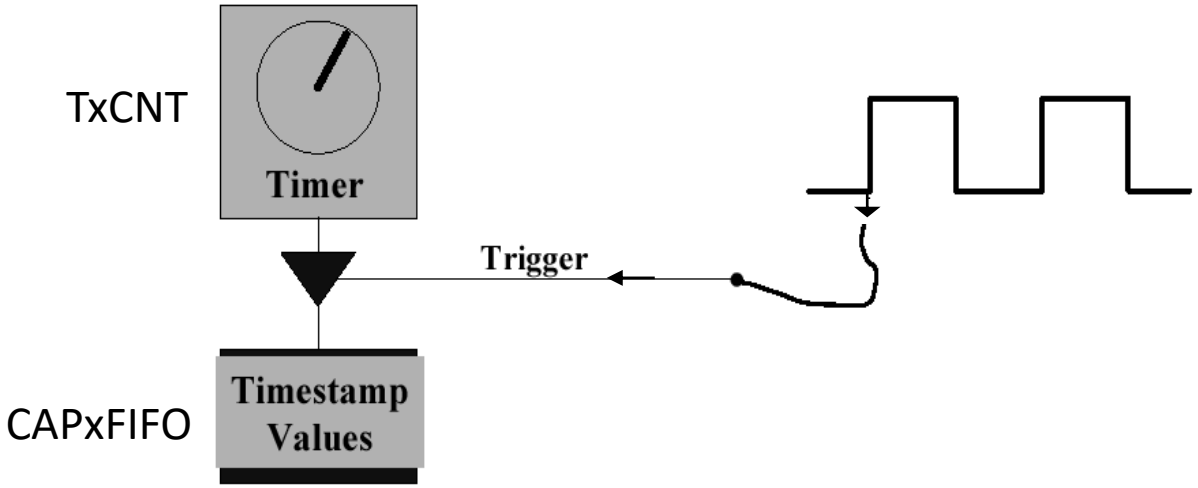
Unidad de comparación completa (6PWM).



Periféricos

Entradas de captura (CAP).

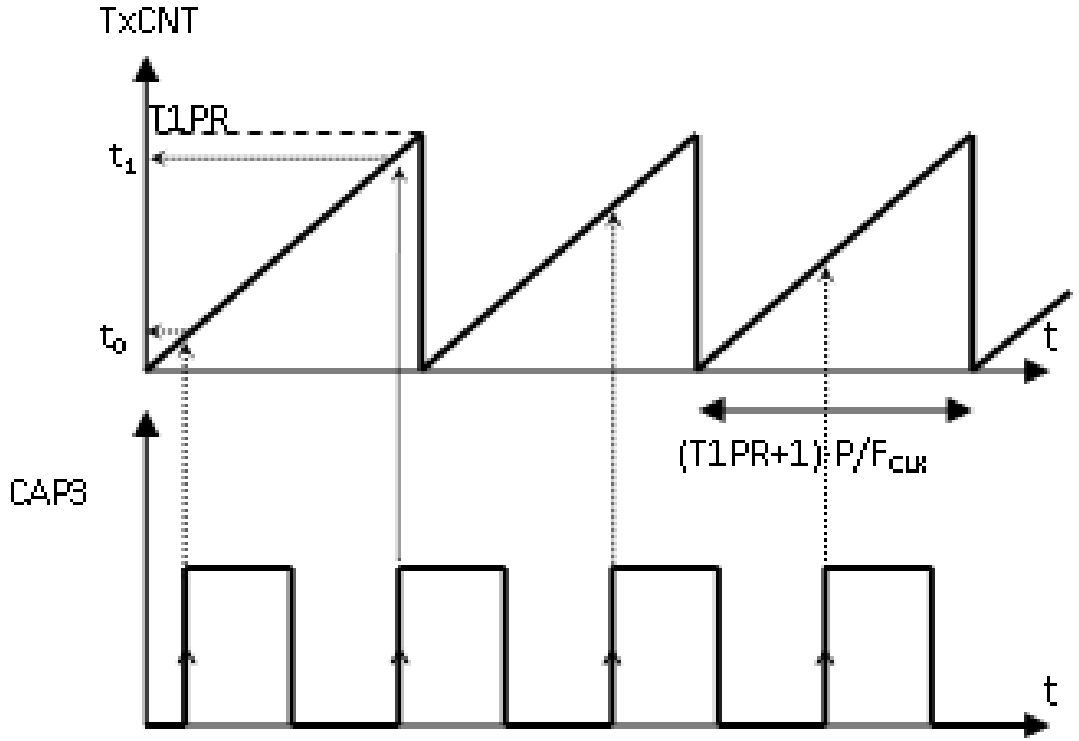
- Entrada digital .
- Detecta el instante en el que cambia una señal
- El temporizador puede tener preescalado
- Para medir periodos se restan dos capturas

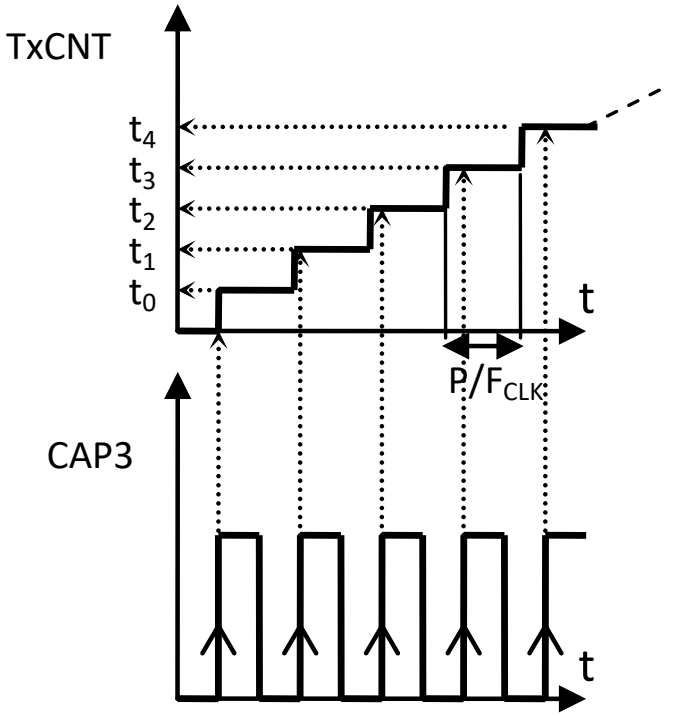


Periféricos

Entradas de captura (CAP).

- Rango de periodos y precisión (compromiso)

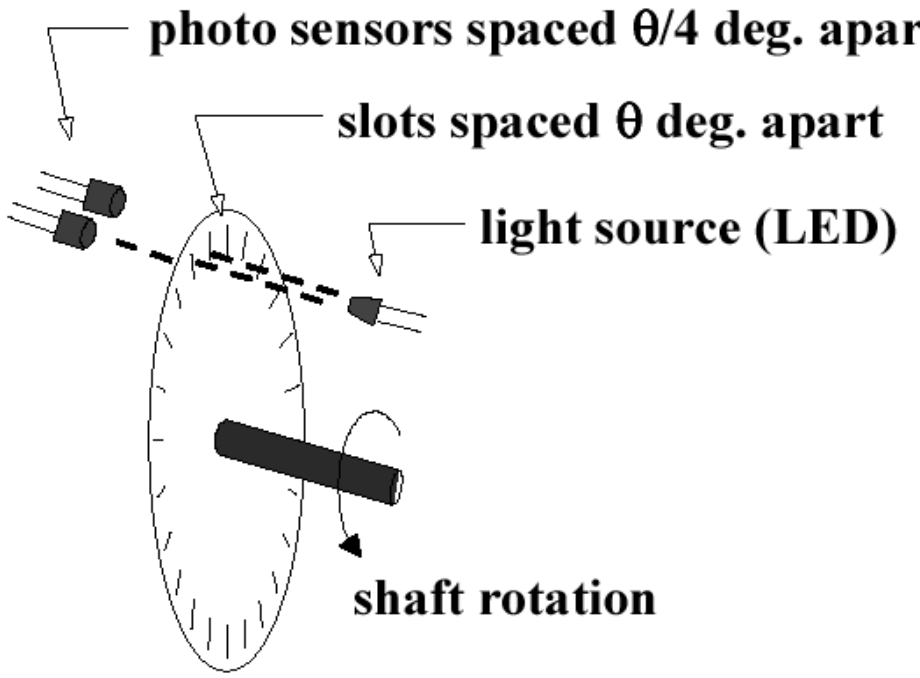




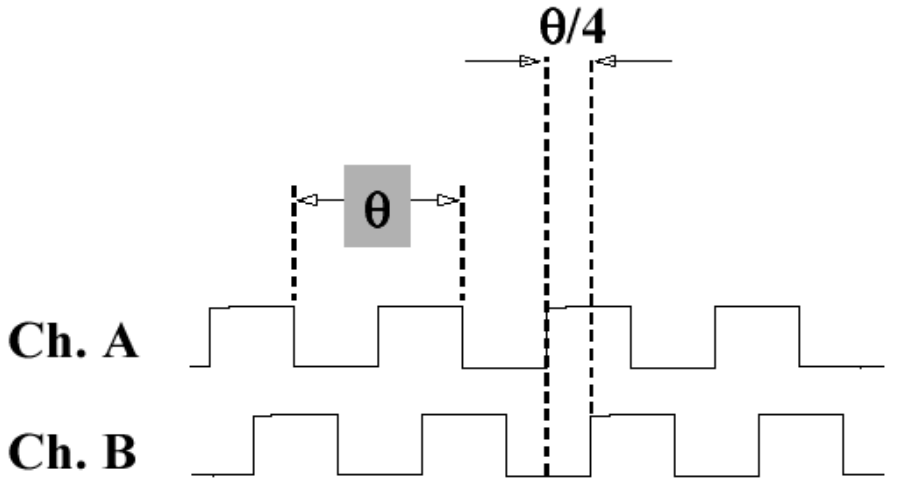
Periféricos

Entradas de encoder (QEP).

A digital (angular) position sensor



Incremental Optical Encoder



Quadrature Output from Photo Sensors

Periféricos

Entradas de encoder (QEP).

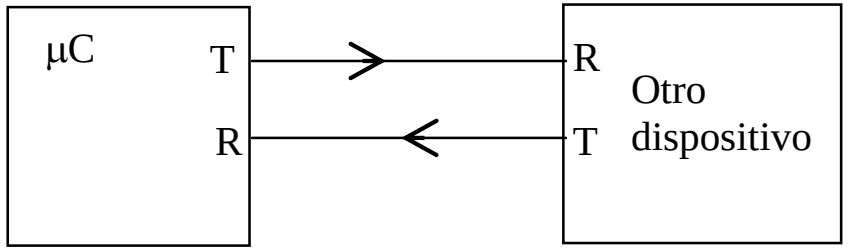
- Son 2 entradas digitales
- El encoder genera la señal de reloj (y diriección)
- Un temporizador cuenta los pulsos del encoder
- Precisión: la fija el encoder
  - Ranuras/vuelta
  - Grados/ranura
- Rango y resolución:
  - puede ser mayor que una vuelta
  - La fija el temporizador asociado (máximo número ctas.)

Periféricos

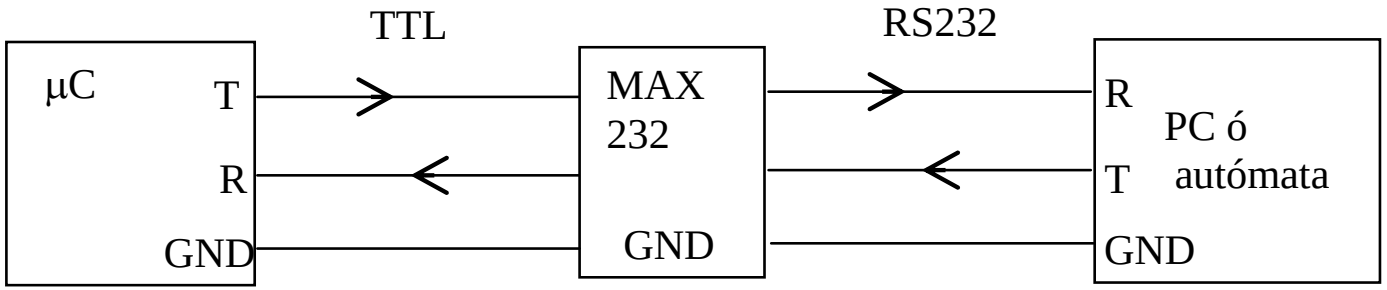
Puertos de comunicaciones

Comunicación serie asíncrona (UART)

- Recepción-transmisión asíncrona universal



- Adecuación a RS232

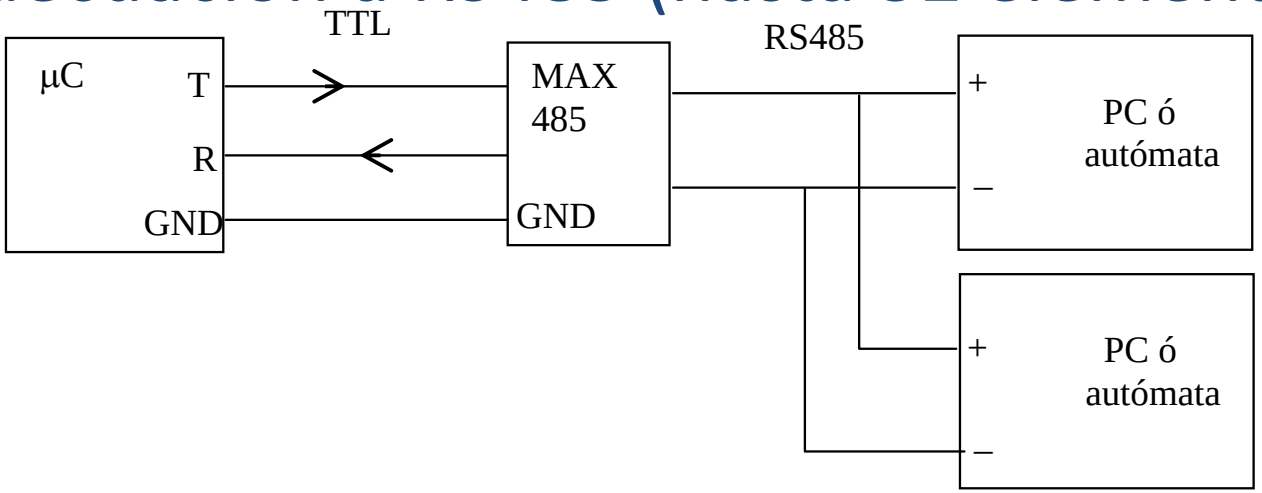


Periféricos

Puertos de comunicaciones

Comunicación serie asíncrona (UART)

- Adecuación a RS485 (hasta 32 elementos)



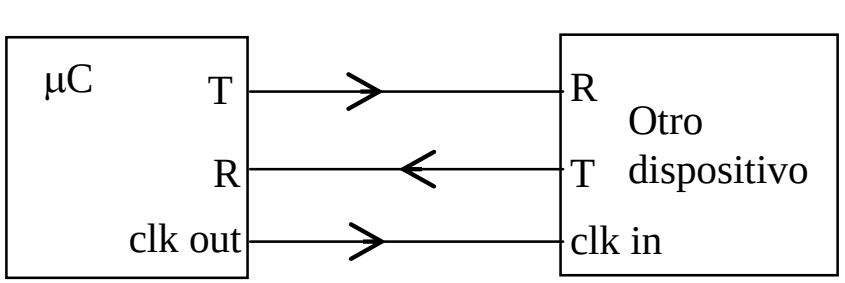
- Adecuación a USB: FTDI FT232R

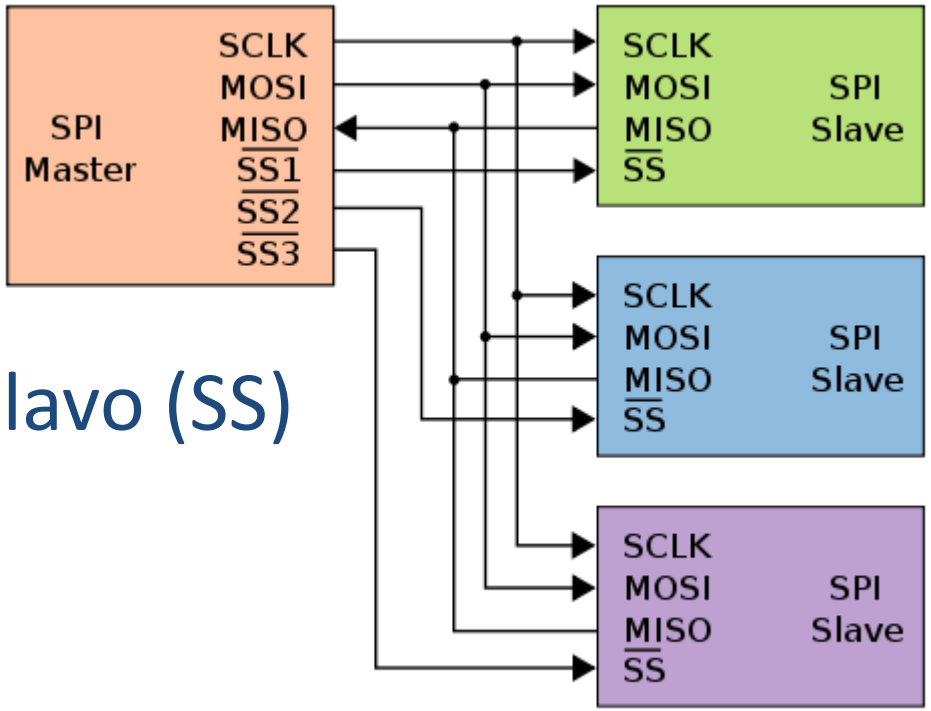
Periféricos

Puertos de comunicaciones

Comunicación serie síncrona (USRT)

- SPI (Serial Peripheral Interface)





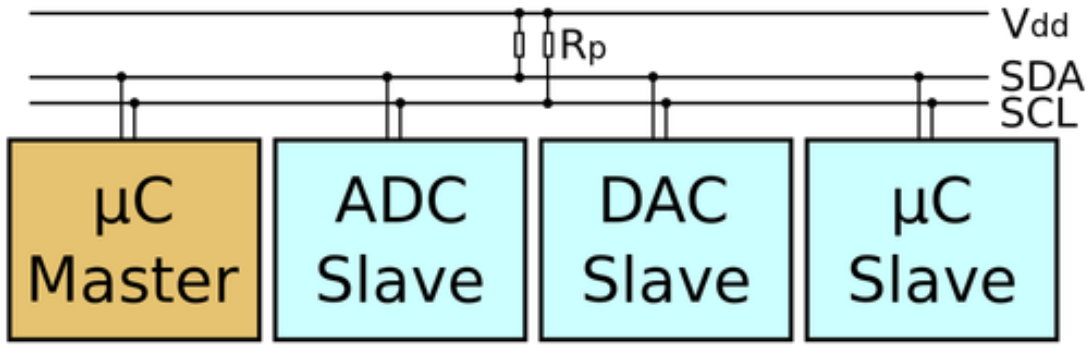
- Utiliza selección de esclavo (SS)

Periféricos

Puertos de comunicaciones

Comunicación serie síncrona (USRT)

- I2C (Inter-Integrated Circ.). 2 cables.
- Utiliza direccionamiento de 7 bits (127 disp.)



Periféricos

Puertos de comunicaciones

Comunicación CAN

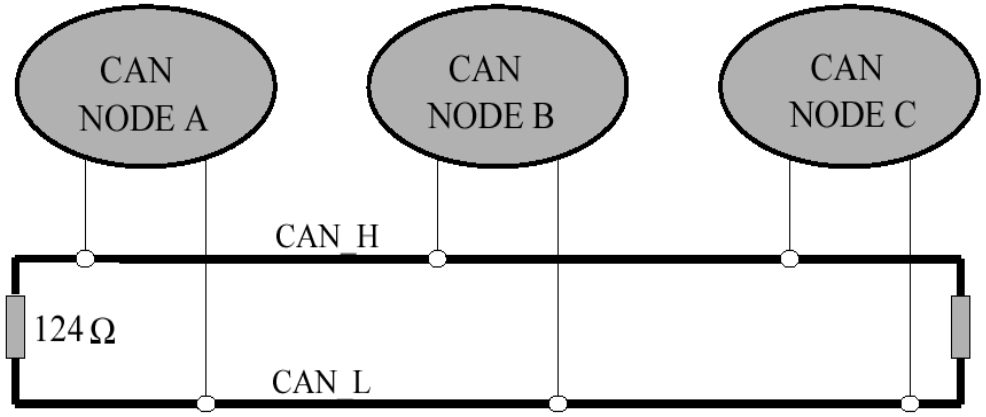
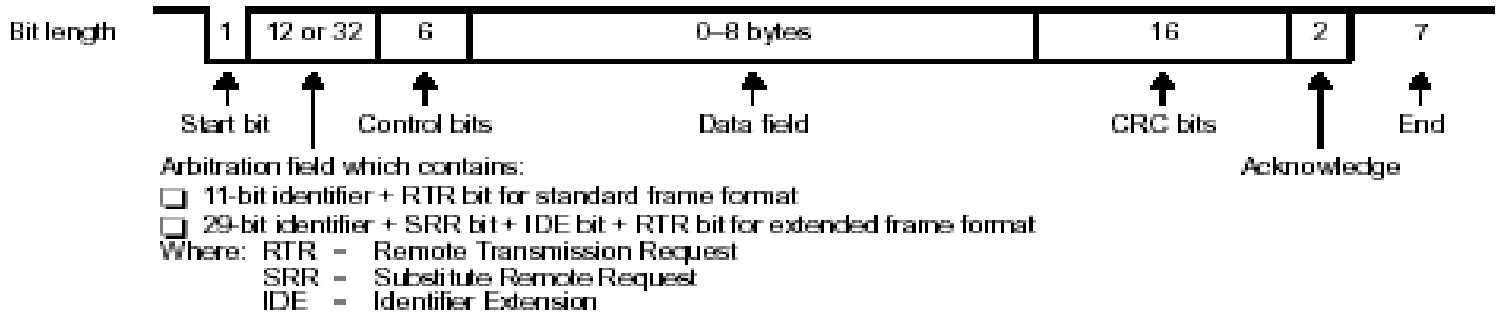


Figure 10–1. CAN Data Frame



Periféricos

Protocolos de comunicación. Buses de campo

- HART
- PROFIBUS
- FOUNDATION FIELDBUS
- MODBUS
- DEVICENET

Control secuencial vs. Control continuo.

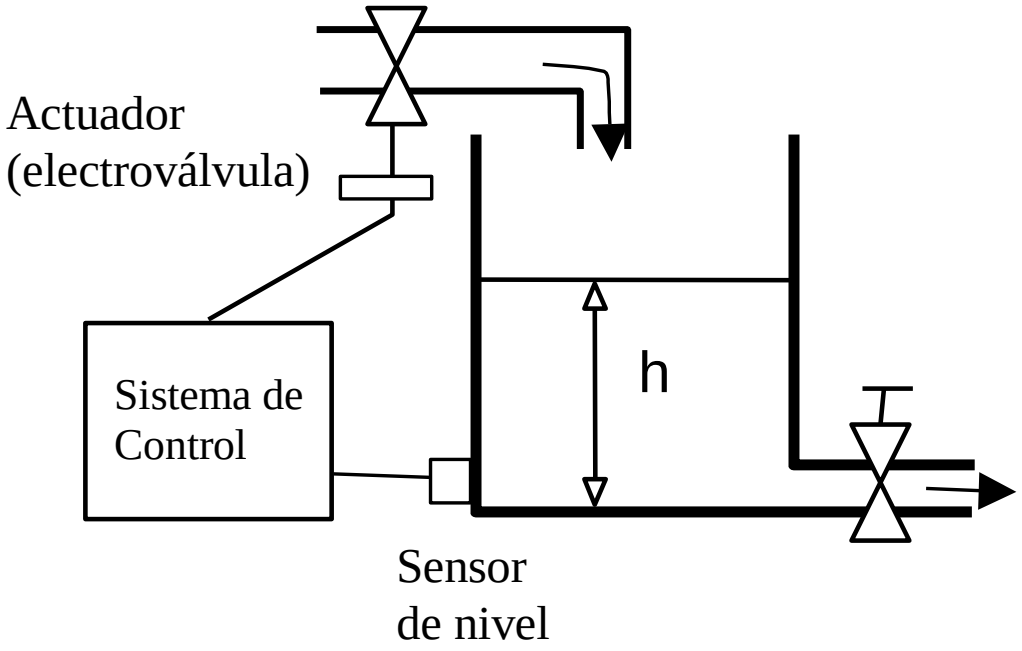
Control secuencial:

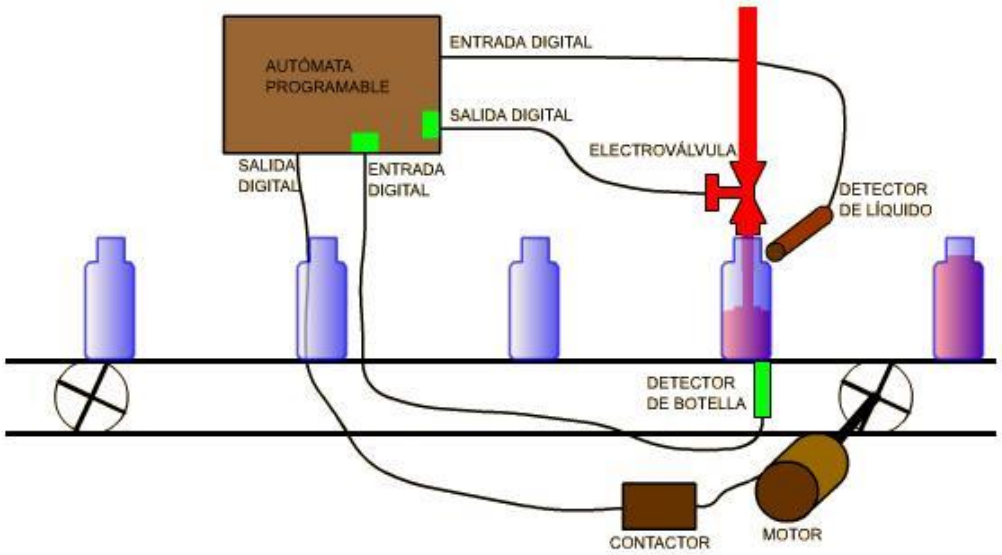
- Sistemas de eventos discretos
- Variables binarias: sensores y actuadores binarios
- El estado del proceso está definido mediante variables binarias

Control continuo:

- Variables con valor arbitrario dentro de un rango amplio
- Sensores y actuadores continuos

Control secuencial vs. Control continuo.

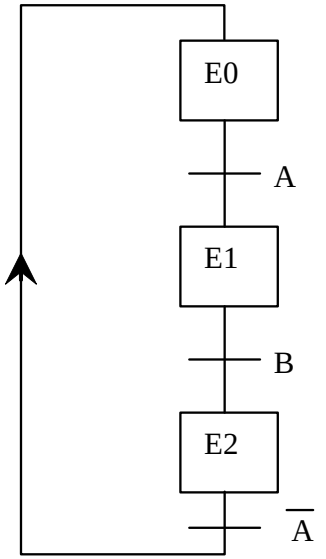




Grafcet

Grafcet: grafo orientado

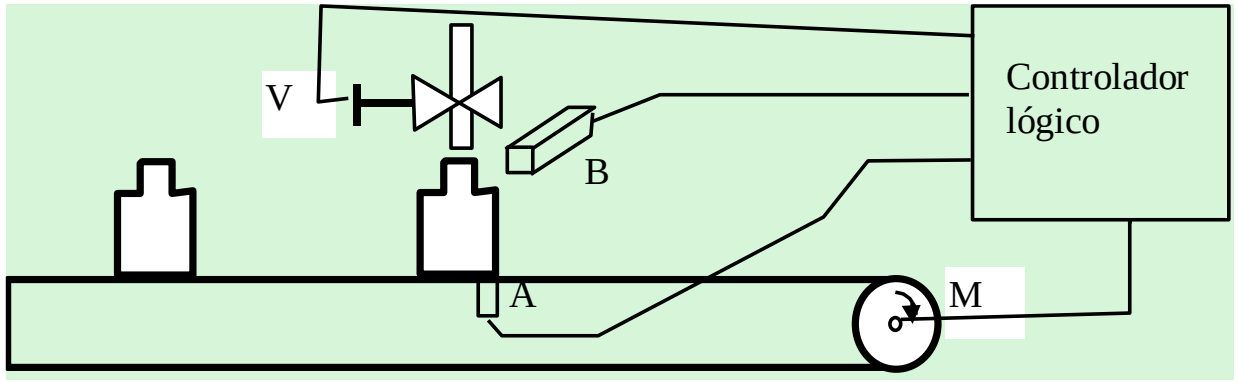
- Formado por dos tipos de nodos:
  - Etapas: Ex
  - Transiciones: —— A

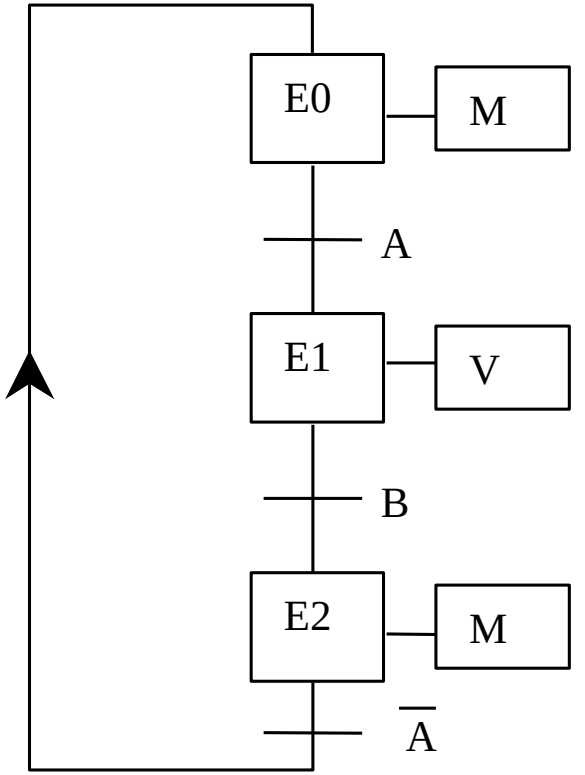


- Orientado de arriba abajo, salvo indicación
- Se une etapa con transición: no se pueden unir dos nodos del mismo tipo

Grafcet

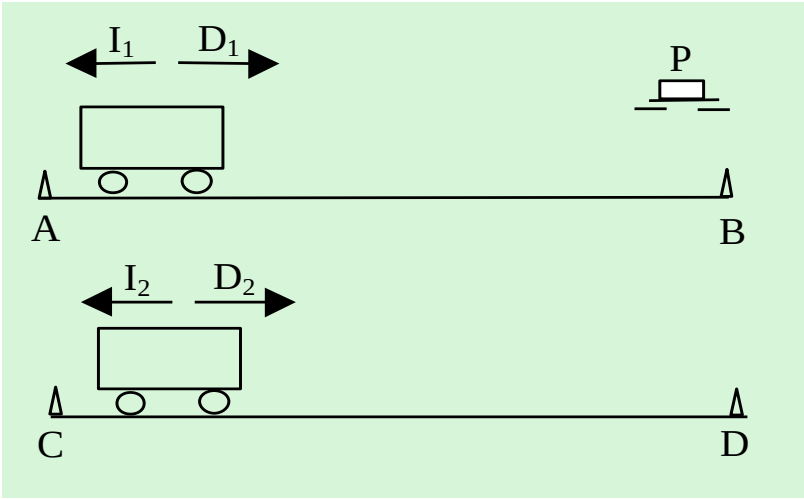
Ejemplo: llenado de botellas

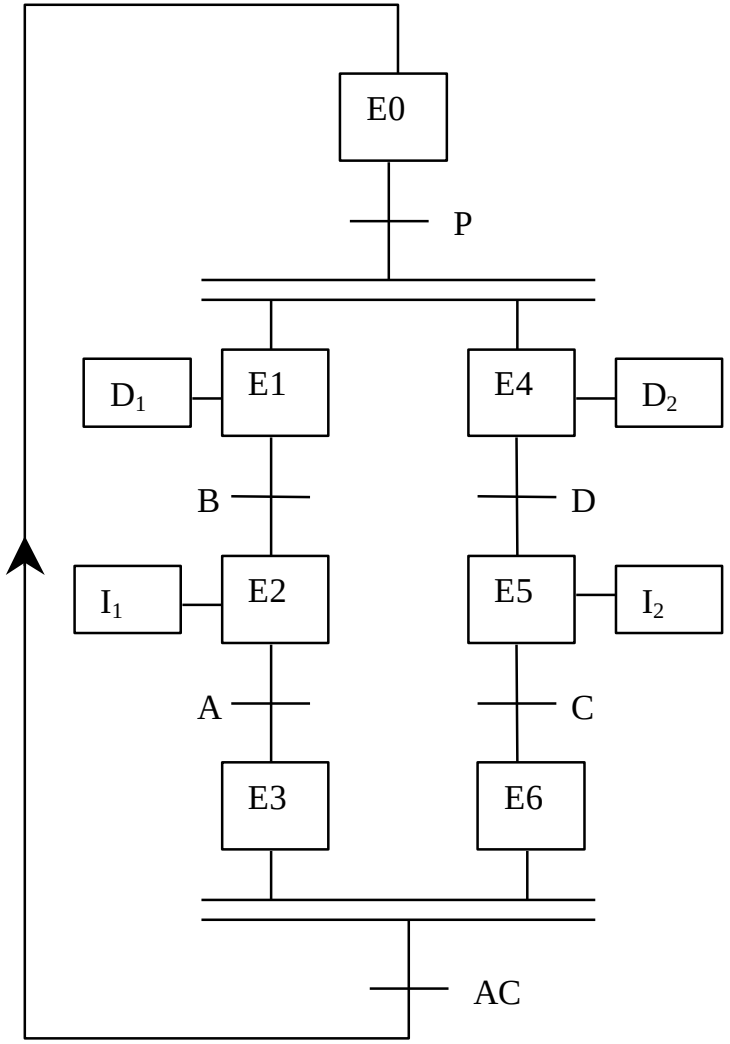




Grafcet

Ejemplo: carros (bifurcación)

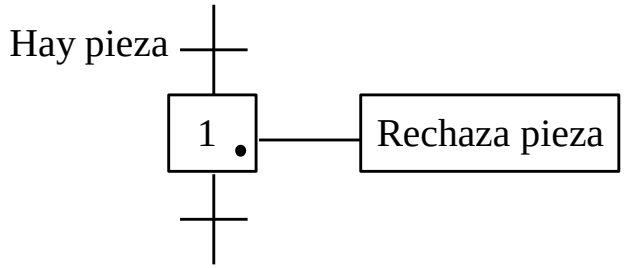




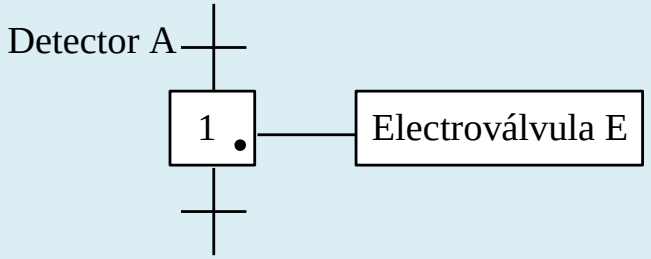
Grafcet

Niveles de descripción

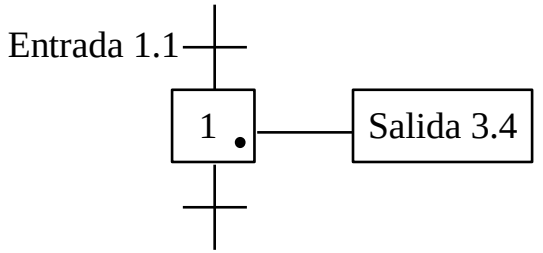
- Nivel 1:
  - Operaciones a realizar sin mencionar tecnología



- Nivel 2:
  - Se detallan las variables que se leen o activan



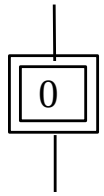
- Nivel 3:
  - Variables del autómata que se leen o activan



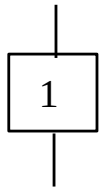
Grafcet

Elementos del Grafcet:

- Etapa:

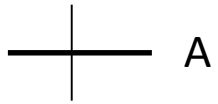


Etapa inicial

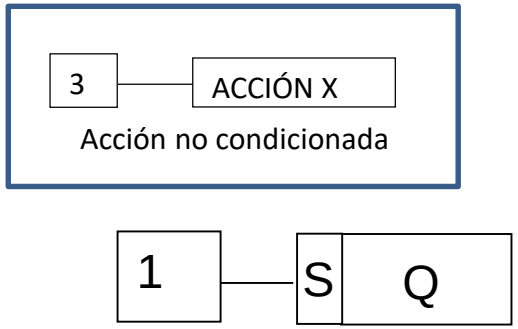


Etapa normal

Transición:

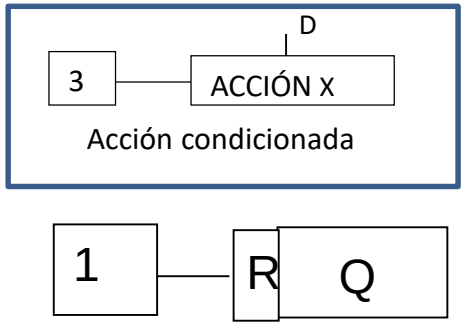


- Acción asociada a una etapa



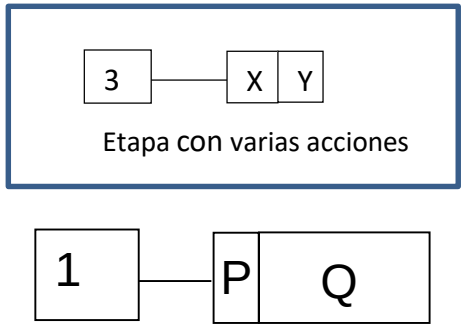
Acción no condicionada

Acción memorizada de set



Acción condicionada

Acción memorizada de reset



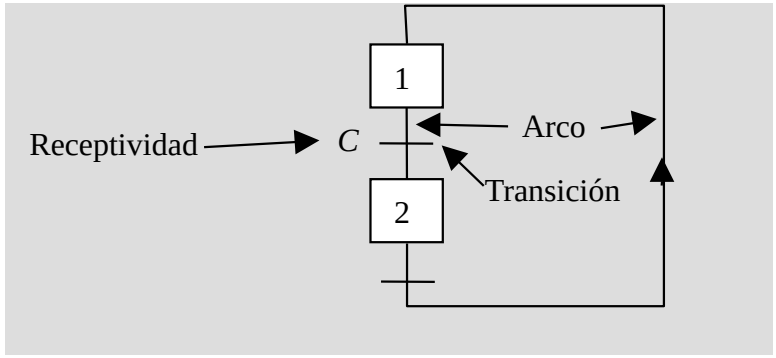
Etapa con varias acciones

Acción impulsional

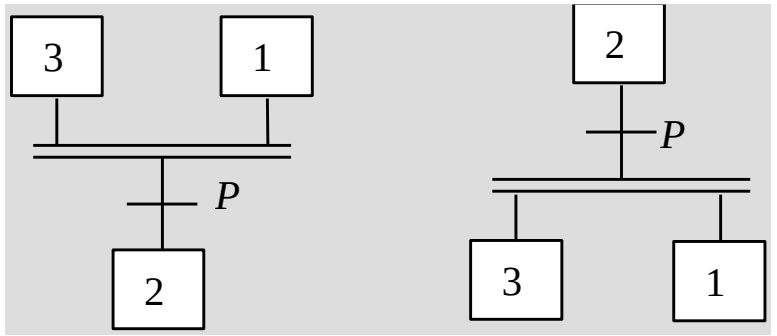
Grafcet

Elementos del Grafcet:

- Arco:



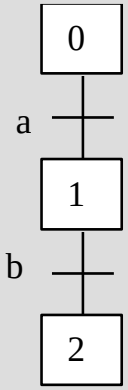
- Trazos paralelos



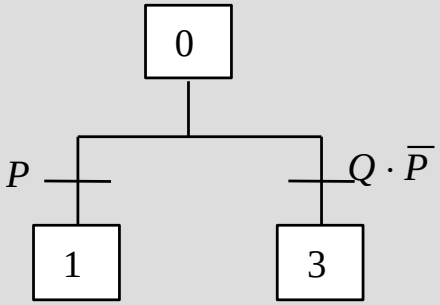
Grafcet

Grafcet. Estructuras:

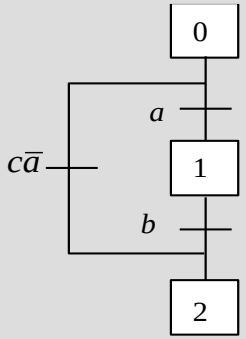
Secuencia



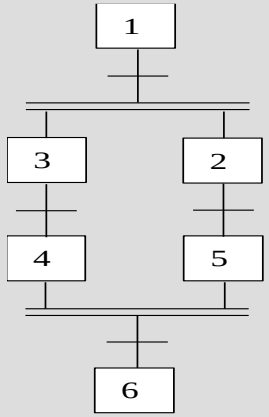
Selección entre secuencias



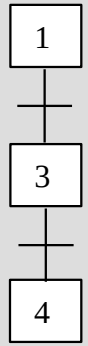
Salto



Paralelismo de secuencias

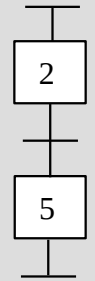


Etapa Fuente



Etapa Sumidero

Transición Fuente



Transición Sumidero

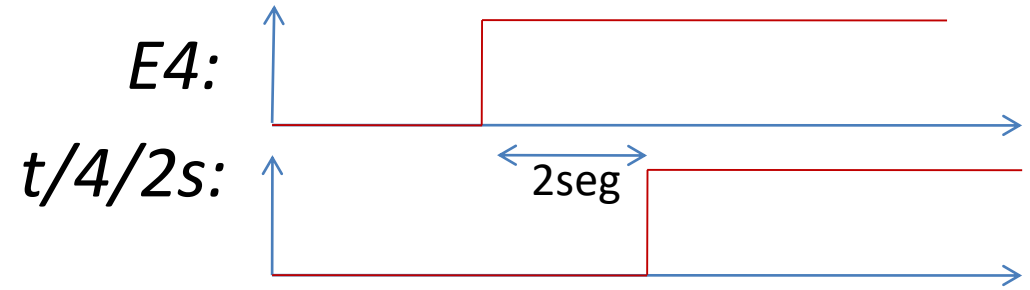
Grafcet

Acciones y receptividades temporizadas

- Son para la medición del tiempo.
- Notación:

t / Nºetapa / tiempo (s)

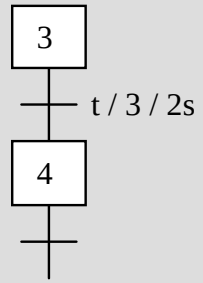
- Ejemplo:
  - ***t/4/2s*** *es una señal que se activa dos segundos después de que se active la etapa 4.*



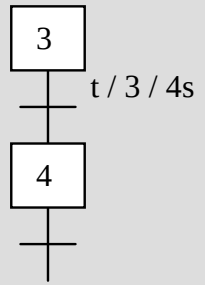
Grafcet

Acciones y receptividades temporizadas

Lograr que una etapa esté activa durante un tiempo determinado:

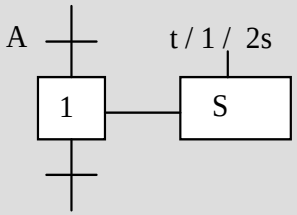


La etapa 3 estará activa durante 2 s

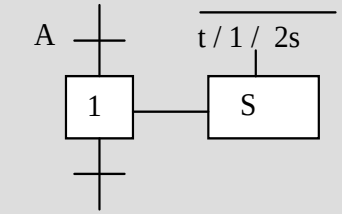


La etapa 3 estará activa durante 4 s

Condicionar una salida:



La salida S se activará 2 segundos después de que se active la etapa 1, desactivándose cuando se desactive la etapa 1.



La salida S se activará en cuanto se active la etapa 1, desactivándose 2 segundos después (si no se ha desactivado antes la etapa 1).

Grafcet

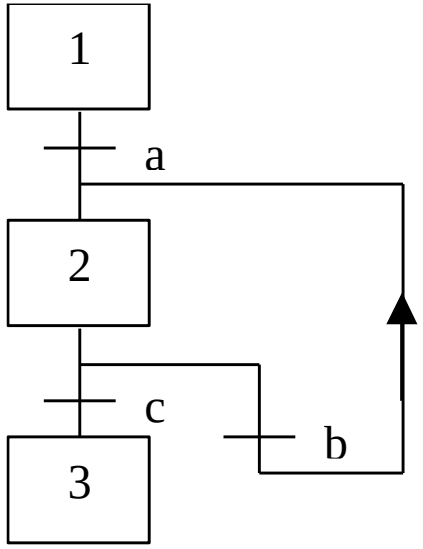
Grafcet. Reglas de evolución

- Inicialización:
  - Sólo se activan las etapas iniciales (doble cuadro).
- Evolución de las transiciones:
  - Si las etapas anteriores a una transición están activas y la receptividad asociada es cierta, la transición se dispara inmediatamente.
- Evolución de las etapas.
  - Cuando se dispara una transición todas las etapas inmediatamente anteriores se desactivan y simultáneamente se activan todas las posteriores: **prioridad en la activación**.

Grafcet

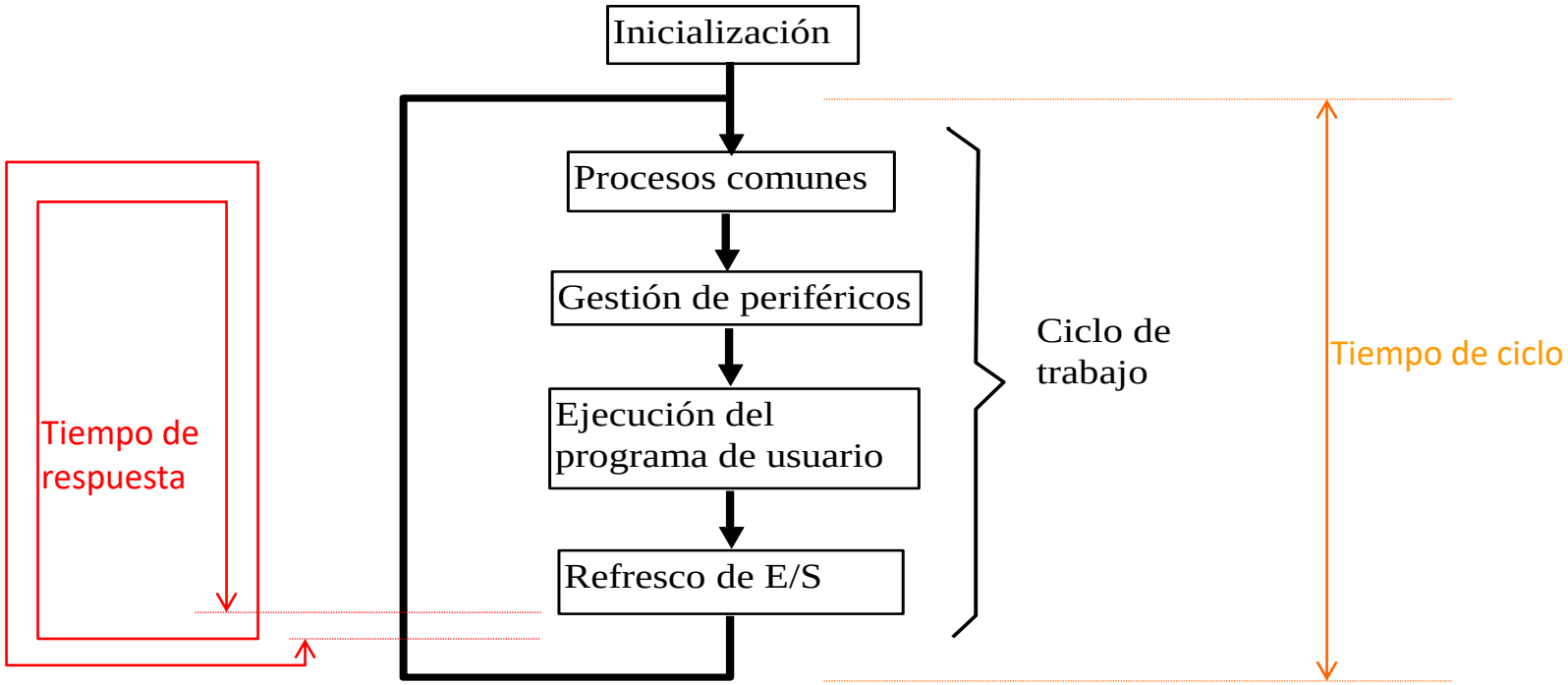
Grafcet. Reglas de evolución

- Prioridad en la activación:
  - Si una etapa debe activarse y desactivarse a la vez, quedará activa.



Programación del Grafcet

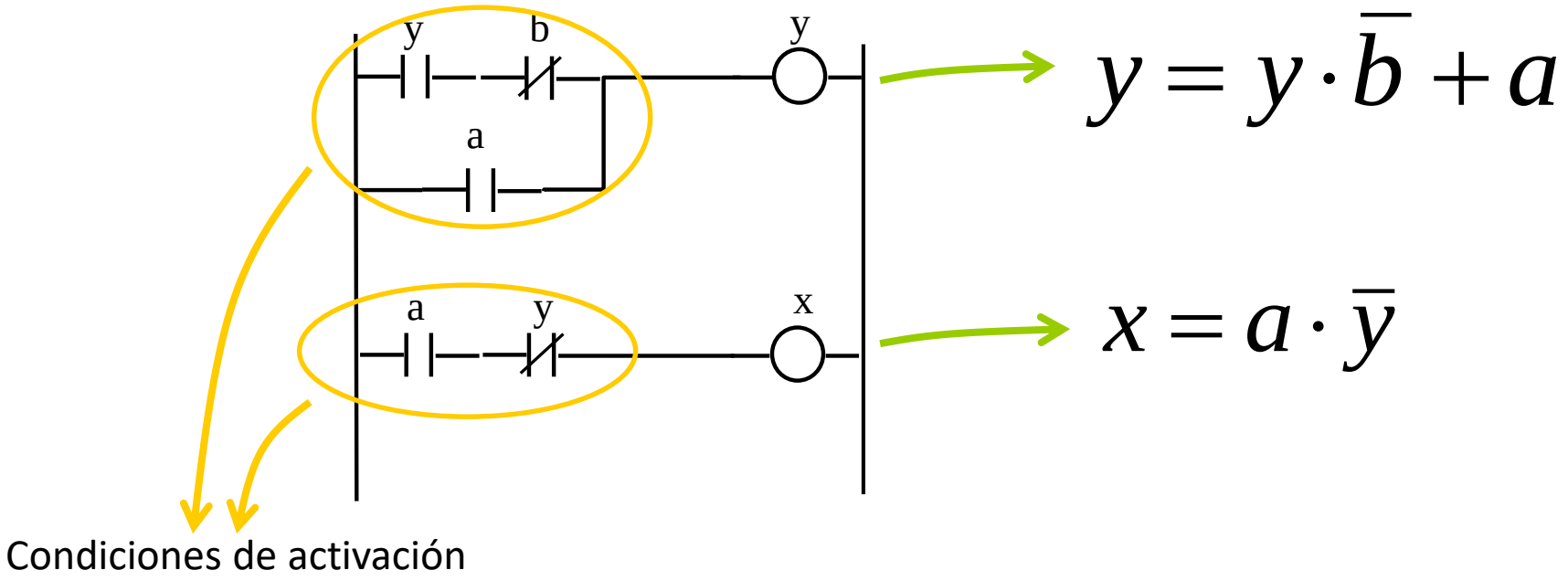
Ciclo de trabajo (SCAN) :



Tresp,máx=2Tciclo+Tout ≈ unos ms ó decenas de ms

Programación del Grafcet

Ejemplo de programa

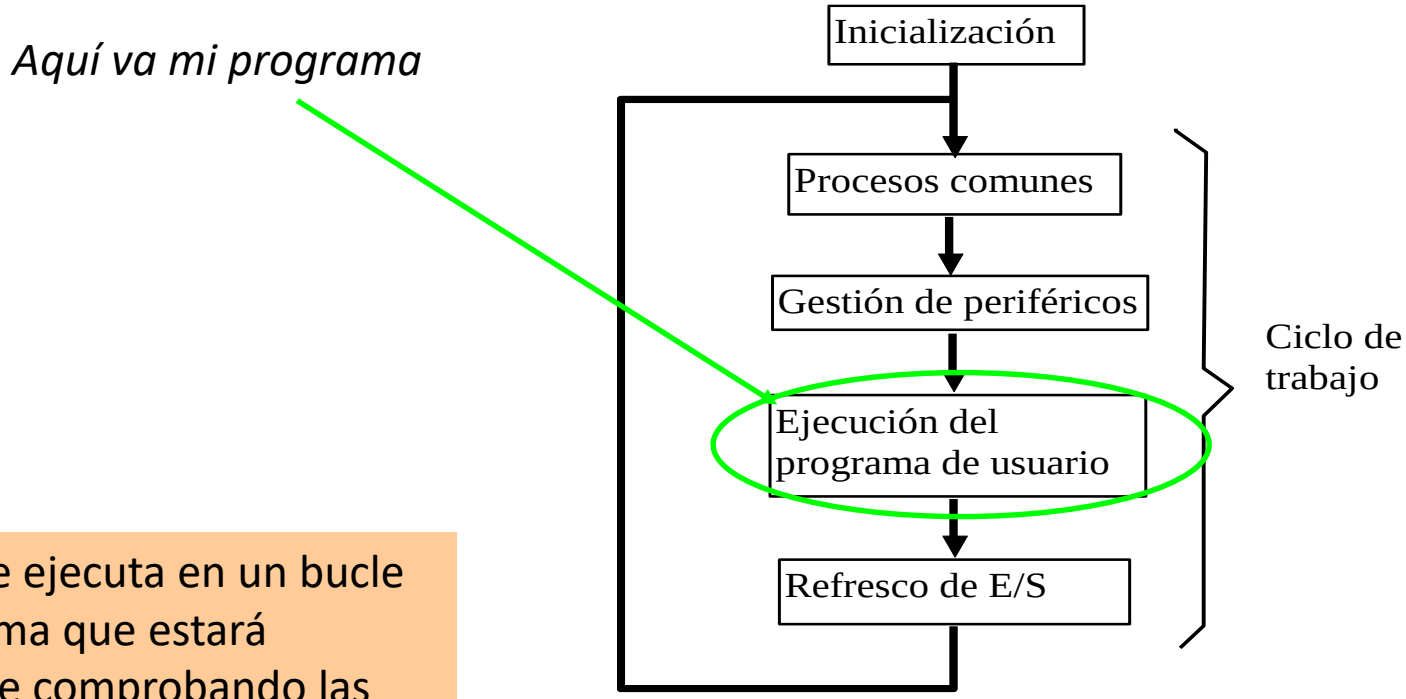


$$y = y \cdot \bar{b} + a$$

$$x = a \cdot \bar{y}$$

Programación del Grafcet

Recordar



El algoritmo se ejecuta en un bucle infinito de forma que estará continuamente comprobando las entradas para modificar las etapas y las salidas

Programación del Grafcet

Programa de usuario

1. **Inicialización** de las etapas (solo una vez).
2. **Lectura de entradas**
3. **Detección de flancos de entradas**
4. **Desactivación/activación de las etapas**
5. **Detección de flancos de etapas**
6. **Activación de salidas**
7. **Temporizadores y contadores**
8. **Actualización de salidas**

Programación del Grafcet

1. Inicialización

```
boolean E[3]={1,0,0};
const int pinP=2;
int P;
const int pinS=4;
int S;
void setup()
{
      pinMode(pinP,INPUT);
      pinMode(pinS,OUTPUT);
      Serial.begin(9600);

}
```

Programación del Grafcet

2. Lectura de entradas

```
P=digitalRead(pinP);
A=analogRead(pinA);
```

```
if (Serial.available() > 0) {
      ByteEntrante = Serial.read();
      }
```

Programación del Grafcet

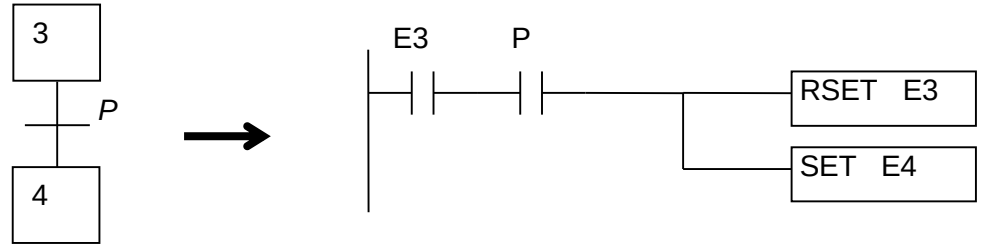
3. Detección de flancos de entradas

```
Flanco_A = (!Copia_A) && A;
Copia_A =A;
```

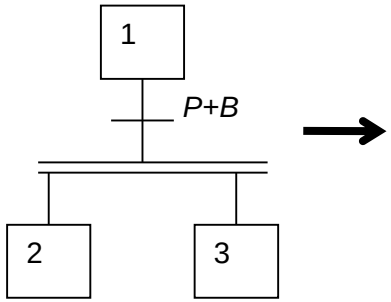
```
Flanco_bajada_A = (Copia_A) && !A;
Copia_A =A;
```

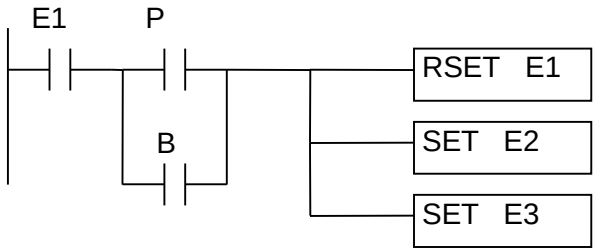
Programación del Grafcet

4. Desactivación/Activación etapas




```
if(E[3] && P)
{
        E[3]=0;
        E[4]=1;
}
```

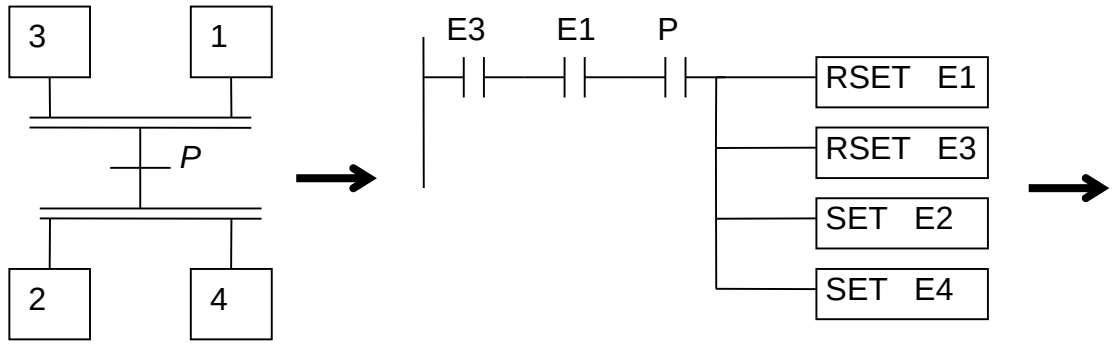




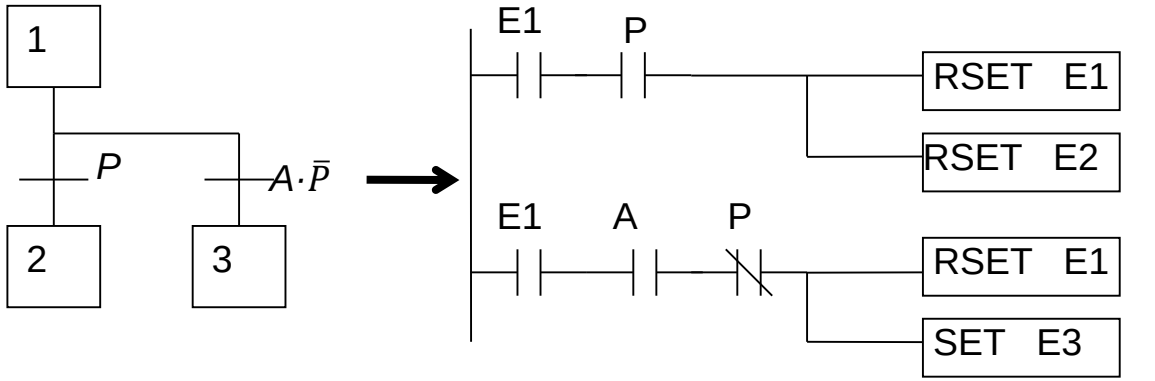

```
if(E[1] && (P || B))
{
        E[1]=0;
        E[2]=1;
        E[3]=1;
}
```

Programación del Grafcet

4. Desactivación/Activación etapas




```
if(E[3] && E[1] && P)
{
        E[1]=0;
        E[3]=0;
        E[2]=1;
        E[4]=1;
}
```

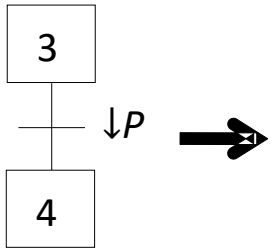



```
if(E[1] && P)
{
        E[1]=0;
        E[2]=1;
}
if(E[1] && A && !P)
{
        E[1]=0;
        E[3]=1;
}
```

Programación del Grafcet

4. Desactivación/Activación etapas

Transiciones por flanco

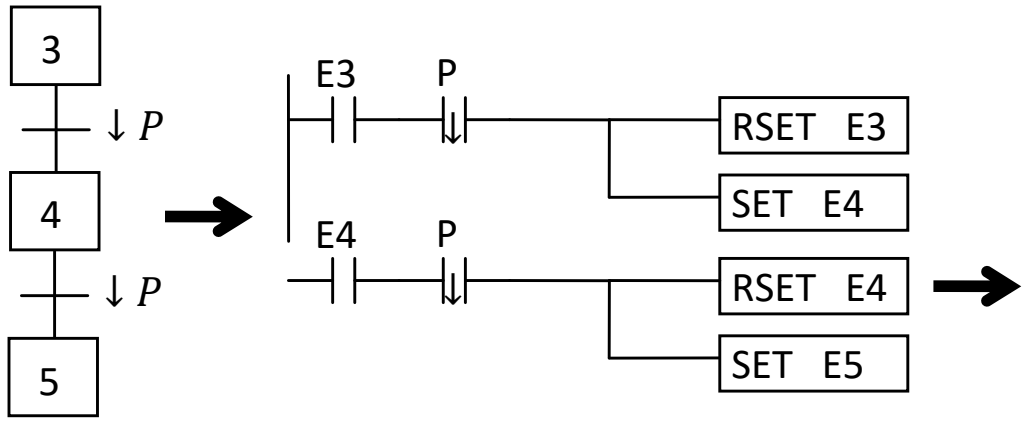



```
Flanco_P=(Copia_P)&&!P;
Copia_P =P;
if(E[3] && Flanco_P){
      Flanco_P=0;
      E[3]=0;
      E[4]=1;
}
```

Programación del Grafcet

4. Desactivación/Activación etapas

Transiciones por flanco




```
Baja_P=(Copia_P)&&!P;
Copia_P =P;
if(E[3] && Baja_P){
      E[3]=0;
      E[4]=1;
      Baja_P=0;
}
if(E[4] && Baja_P){
      E[4]=0;
      E[5]=1;
      Baja P=0;
```

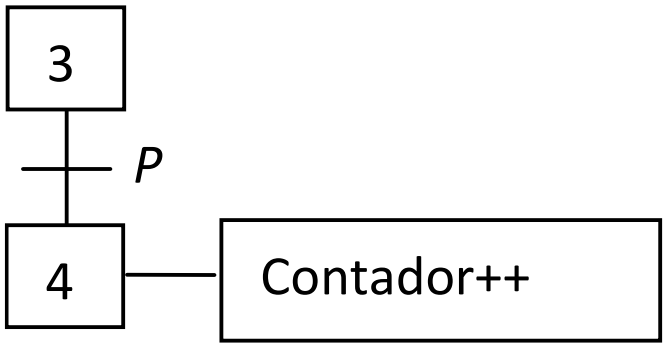
Programación del Grafcet

4. Desactivación/Activación etapas

Activación de salidas impulsionales o memorizadas

(contadores, SET, RESET, etc.)

Se implementan justo al llegar a la nueva etapa




```
if(E[3] && P){
      E[3]=0;
      E[4]=1;
      Contador++;
}
```

Programación del Grafcet

5. Definición de temporizadores

```
long tiempo_ms, tiempo_us;
//variables enteras de 32 bits.
tiempo_ms=millis();
// contaje actual en ms
tiempo_us=micros();
// contaje actual en us
```

Programación del Grafcet

5. Definición de temporizadores

Ejemplo (t/4/3s condiciona E5)

```
unsigned long fin_t_4_3s=0; // Definición variables
boolean t_4_3s =0;
```

En la desactivación/activación de las etapas (paso 4, transiciones):

```
if(E[3] && A){
            E[3]=0;
            E[4]=1;
            fin_t_4_3s = millis()+3000;
}
```

Para comprobar si ha pasado el tiempo (paso 5):

```
t_4_3s = E[4]&&(millis() >= fin_t_4_3s);
```

Para evitar desbordamientos:

```
t_4_3s = E[4]&&((long) millis() - (long) fin_t_4_3s >=0);
```

Programación del Grafcet

5. Definición de temporizadores

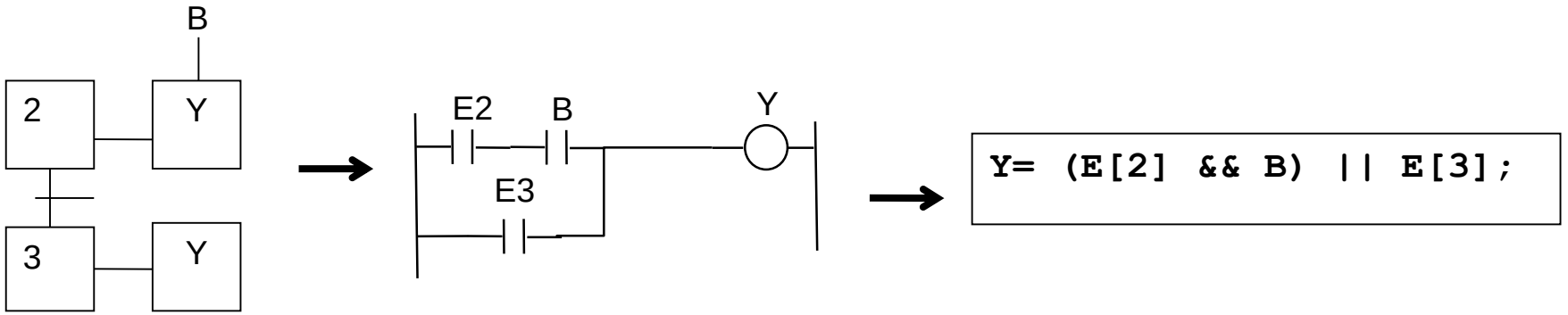
Ejemplo (t/4/5s condiciona E5)

En el paso 4 (activación/desactivación):

```
if(E[4]  && t_4_ts){
          E[4]=0;
          E[5]=1;
}
```

Programación del Grafcet

6. Activación de las salidas



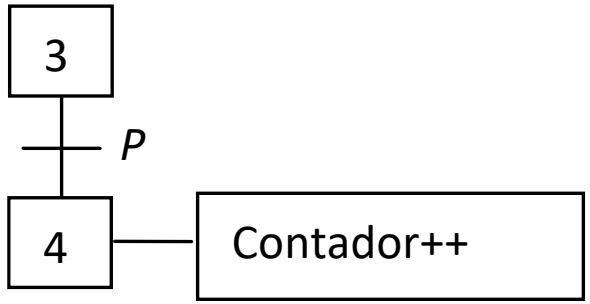

```
Y= (E[2] && B) || E[3];
```

Programación del Grafcet

6. Activación de las salidas

Acciones impulsionales (contador, p.ej.)

Se incluyen junto a la activación de la etapa




```
if(E[3] && P){
        E[3]=0;
        E[4]=1;
        Contador++;
}
```

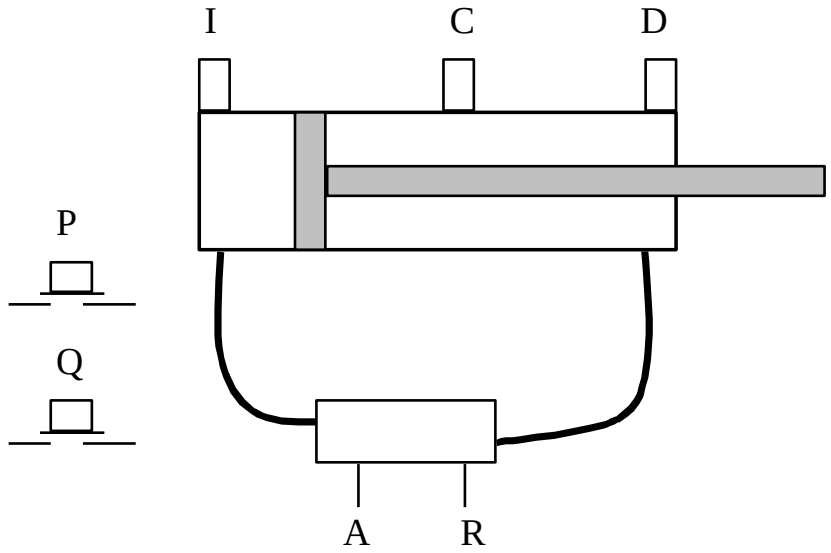
Programación del Grafcet

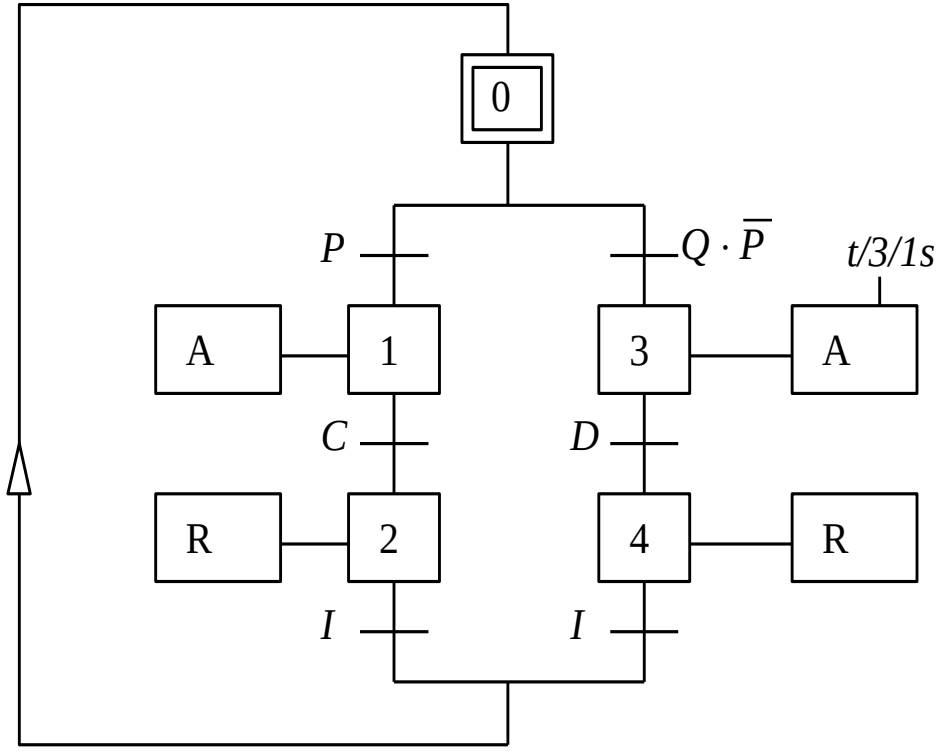
7. Actualización de salidas

```
digitalWrite(pinS,S);

analogWrite(pinS,valor);
```

Programación del Grafcet

Ejemplo





Programación del Grafcet

Ejemplo

```
// Definición de variables e inicialización
const int pinP=0, pinQ=1, pinI=2, pinC=3, pinD=4, pinA=5, pinR=6;
boolean P=0, Q=0, I=0, C=0, D=0, A=0, R=0;
boolean E[5]={1, 0, 0, 0, 0},
boolean t_3_1s=0;
unsigned long fin_t_3_1s=0;
```

Programación del Grafcet

Ejemplo

```
void setup(){
    // Configuración de entradas y salidas
    pinMode(pinP,INPUT);
    pinMode(pinQ,INPUT);
    pinMode(pinI,INPUT);
    pinMode(pinC,INPUT);
    pinMode(pinD,INPUT);
    pinMode(pinA,OUTPUT);
    pinMode(pinR,OUTPUT);

    // Inicialización de las salidas
    digitalWrite(pinA,0);
    digitalWrite(pinR,0);
}
```

Programación del Grafcet

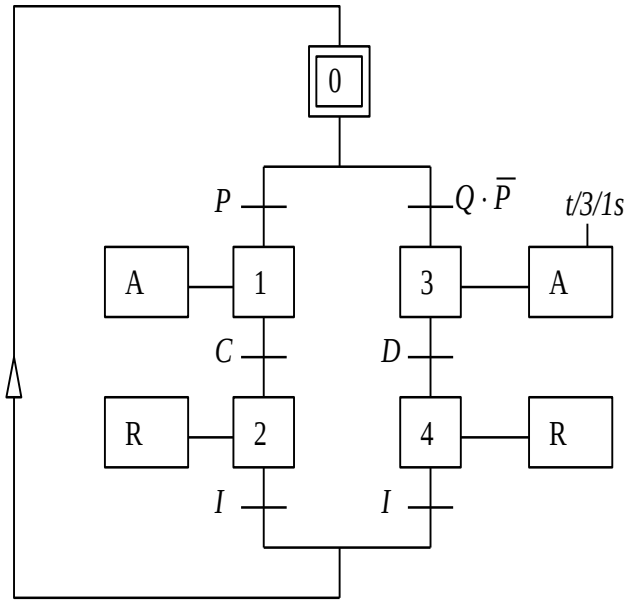
Ejemplo

```
void loop(){
    //Lectura de entradas
    P=digitalRead(pinP);
    Q=digitalRead(pinQ);
    I=digitalRead(pinI);
    C=digitalRead(pinC);
    D=digitalRead(pinD);
```

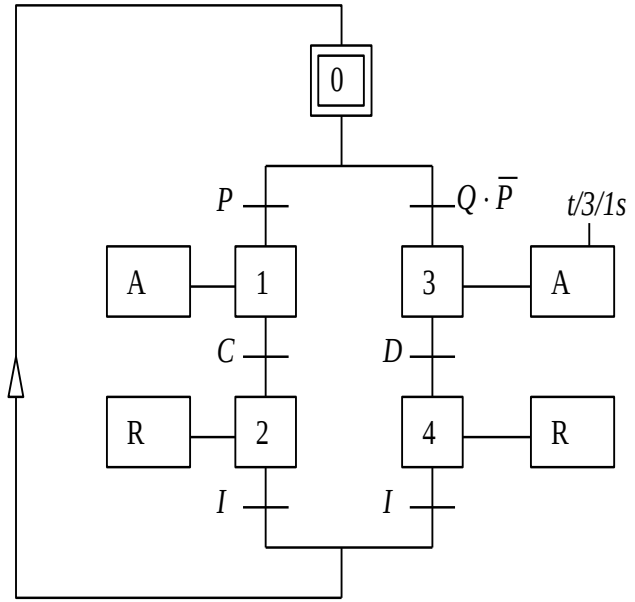
Programación del Grafcet

Ejemplo

```
//Desactivación/Activación de etapas
if(E[0] && P){E[0]=0; E[1]=1;}
if(E[1] && C){E[1]=0; E[2]=1;}
if(E[0] && !P && Q){E[0]=0; E[3]=1;
fin_t_3_1s = millis()+1000;}
if(E[3] && D){E[3]=0; E[4]=1;}
if(E[2] && I){E[2]=0; E[0]=1;}
if(E[4] && I){E[4]=0; E[0]=1;}
```



Programación del Grafcet

Ejemplo

```
// Definición de temporizadores
t_3_1s = E[3] && (millis()>=fin_t_3_1s);

// Activación de salidas
    A = (E[1] || (E[3] && t_3_1s));
    R = (E[2] || E[4]);

// Actualización de salidas
    digitalWrite(pinA,A);
    digitalWrite(pinR,R);
}
```