

# OPERACIONES ARITMÉTICAS Y DIGITALES

PLC Control Continuo



## Índice

- **Objetivos**
  1. **Números hexadecimales.**
  2. **Codificación BCD.**
  3. **Operaciones Aritméticas**
  4. **Operaciones Digitales.**
- **Bibliografía**



## Objetivos

- **Identificar el principio de funcionamiento de las operaciones aritméticas y lógicas.**
- **Programar las operaciones aritméticas y lógicas.**



## CONTENIDOS

- **Números hexadecimales.**
- **Codificación BCD.**
- **Operaciones Aritméticas**
- **Operaciones Digitales.**
- **Enmascaramientos.**



## ¿Que sabemos hasta ahora?



- Sabemos que un **BIT**, puede únicamente ser: "0" ó "1"

1

- Que un **BYTE**, es un conjunto de 8 bits.

0 0 1 1 0 1 1 1

- Que una **PALABRA** es un conjunto de 16 bits, equivale también a 2 Bytes.

1 0 1 1 0 1 0 1 1 0 1 0 1 1 0 0

## ¿Que sabemos hasta ahora?



También sabemos que las palabras utilizadas en los PLC, contienen valores numéricos y que estos pueden ser, visualizados o ingresados en diferentes códigos de numeración:

código *decimal*, base 10, el de uso diario.

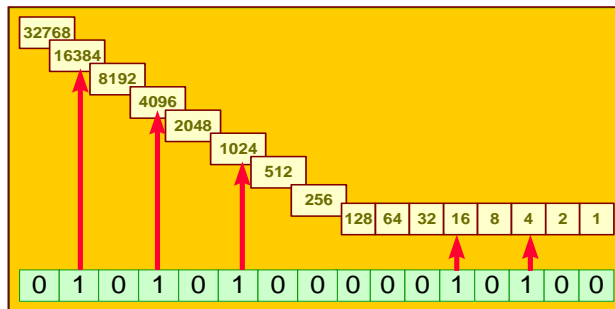
código *binario*, base 2.

código *hexadecimal*, base 16.

## ¿Que sabemos hasta ahora?



- En los sistemas binarios cada bit tiene un peso de acuerdo a su ubicación.



- Su valor es:

16384+

4096+

1024+

16+

4+

= 21524+

**0101010000010100 = 21524**

## ¿Que sabemos hasta ahora?



- El mayor número que se puede expresar con 16 bits en números binarios es:

**1111111111111111 = 65535**

## ¡Algo mas respecto a los números binarios!



Si queremos expresar números positivos y negativos. El bit más significativo (último de la izquierda) se atribuye al signo del valor codificado:

- Si es **0** el contenido de la palabra es positivo.
- Si es **1** el contenido de la palabra es negativo.



**0** 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 = **+32767**

## ¡Algo mas respecto a los números binarios!



- Entonces podemos codificar en una palabra, números binarios en el rango de:

**- 32768** <-----> **+ 32767**

**0** 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 = **+ 32767**

**1** 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 = **- 32768**

- El tratamiento de los valores negativos para el usuario es transparente. Por el momento no es necesario profundizar mas.



## 1. Código Hexadecimal

- En los PLC también es muy frecuente usar palabras con números codificados en **HEXADECIMAL (en base 16)**, lo cual facilita la programación.
- Mientras que en el sistema decimal solo se usan 10 caracteres ( del 0 al 9 ), en el sistema **HEXADECIMAL** se usan **16 caracteres alfanuméricos**.

## Código Hexadecimal



- Del cero al 9 es lo mismo que en codificación binaria.
- Del 10 al 15 se reemplazan por las letras: A, B, C, D, E, F.

HEXADECIMAL		0	0	0	0
	1	0	0	0	1
	2	0	0	1	0
	3	0	0	1	1
	4	0	1	0	0
	5	0	1	0	1
	6	0	1	1	0
	7	0	1	1	1
	8	1	0	0	0
	9	1	0	0	1

### HEXADECIMAL

A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1

## Código Hexadecimal



- La idea es mostrar la combinación de 16 bits que contiene una PALABRA, en tan solo 4 caracteres alfanuméricos. Uno por cada cuarteto de bits.

1 0 1 0 0 1 1 1 1 1 0 0 0 1 0 1

Numero binario

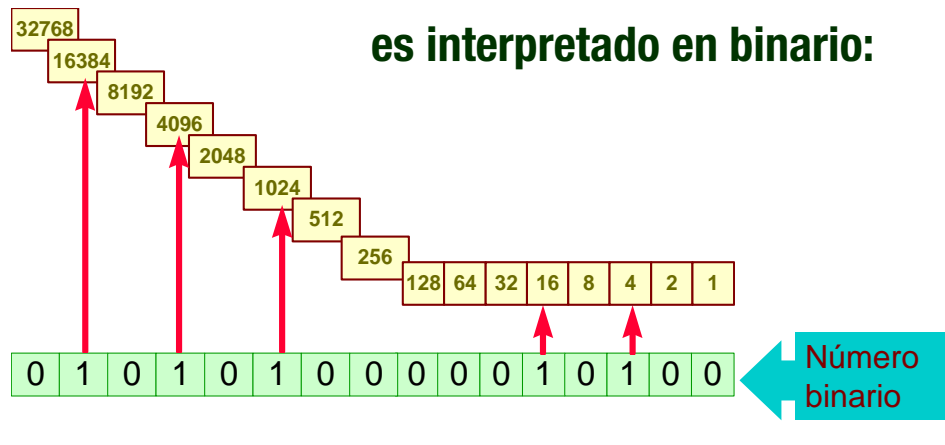
A 7 C 5

Valor HEXADECIMAL

## Código Hexadecimal



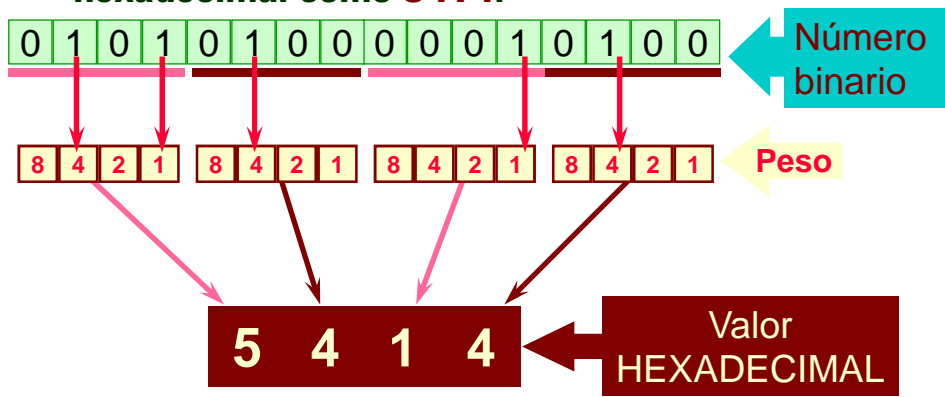
- Por ejemplo el valor decimal **21524** es interpretado en binario:



## Código Hexadecimal



- mientras que, **21524** es interpretado en valor hexadecimal como **5414**:





## Código Hexadecimal



El valor decimal **21524** es visualizado o ingresado en el PLC como:

- **Decimal:** simplemente **21524**
- **Binario:** **2#0101010000010100**
- **Hexadecimal:** **16# '5414'**

## Código BCD



[www.tecsup.edu.pe](http://www.tecsup.edu.pe)

## 2. Código BCD



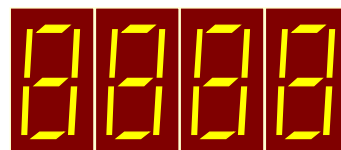
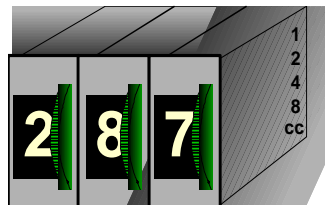
Existen también otro código intermedio entre el decimal y el binario. Este código esta definido por convención, no está regido por ninguna ley matemática.

Es el **BCD** (Binary Coded Decimal),  
valor decimal codificado en binario.

## SISTEMA BCD



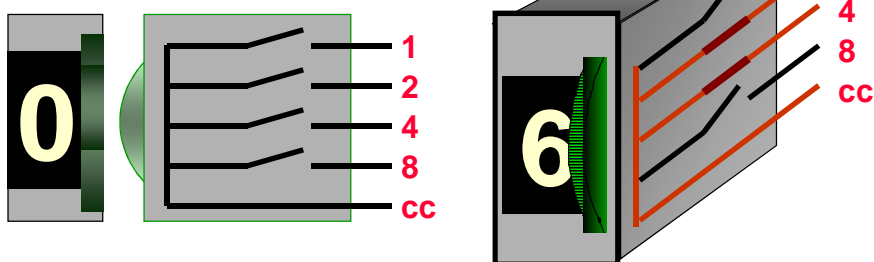
El sistema BCD es utilizado sobre todo para la visualización de informaciones en los visualizadores alfanuméricos (DISPLAY) o en la adquisición de valores desde las ruedas codificadoras (thumbleswitch).



## BCD a BINARIO



Estas ruedas codificadoras consisten en un tambor que a medida que gira va cerrando 4 interruptores de acuerdo al número visualizado.



## SISTEMA BCD



Codificación en DECIMAL	Codificación en BCD
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

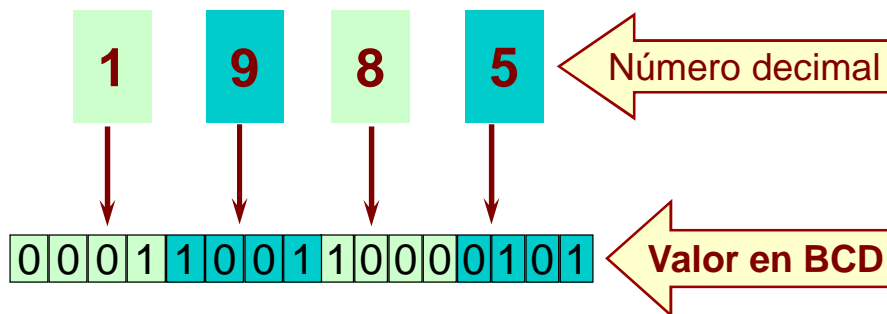
La palabra de 16 bits se desglosa en cuartetos, es decir 4 paquetes de 4 bits. Un paquete por cada cifra comprendida.

**SOLO ENTRE 0 Y 9**

## SISTEMA BCD



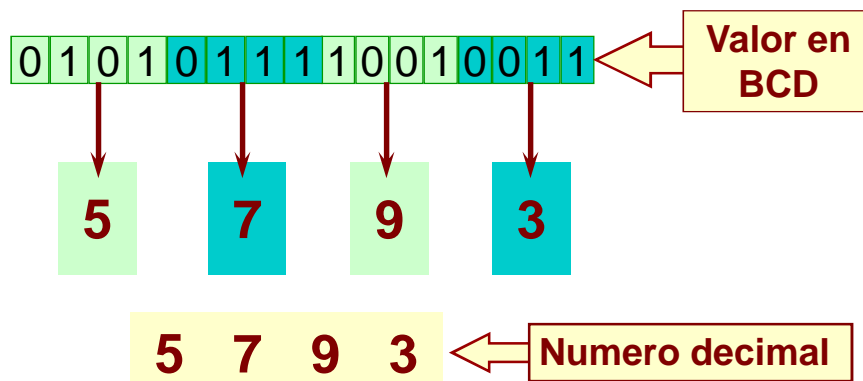
- Por ejemplo el número decimal **1985**, para expresarlo en BCD sería :



## SISTEMA BCD



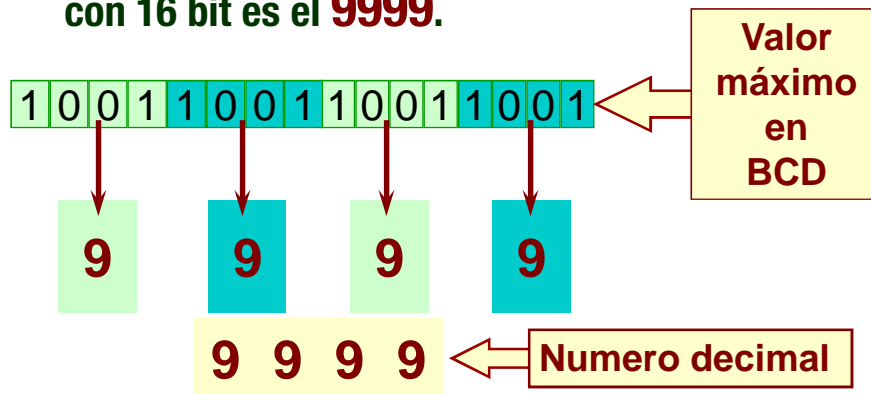
- Si tengo la combinación en BCD, es fácil interpretar que número decimal es:



## SISTEMA BCD



- El mayor valor que puedo expresar, en BCD con 16 bit es el **9999**.



## SISTEMA BCD

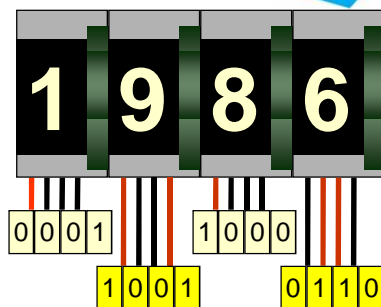


No es frecuente que un operador conozca los números binarios, ni la codificación en BCD.

Pero existen unas ruedas codificadoras en BCD que hacen transparente el ingreso de números al PLC.

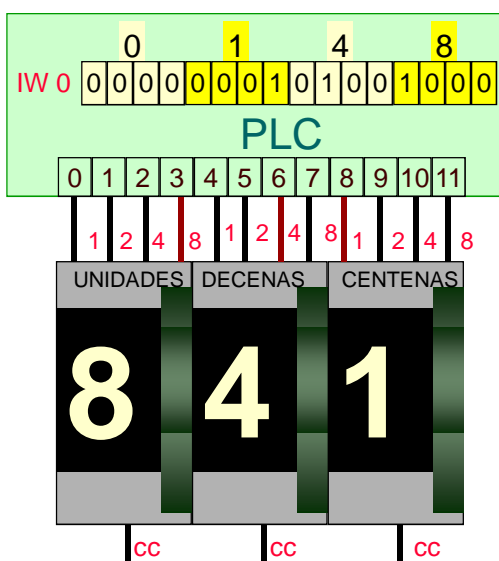
## BCD a BINARIO

Juntando 4 ruedas  
puedo tener hasta  
el numero 9999.



El valor decimal 1986 visualizado en las 4  
ruedas codificadoras, transmiten en sus 16  
salidas la información en código BCD

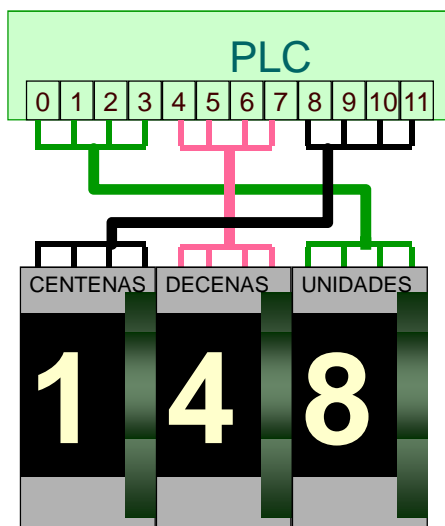
## BCD a BINARIO



Al cablear al PLC hay  
que tener cuidado  
con el orden de las  
ruedas.

En este ejemplo el  
número a ingresar  
es el 148

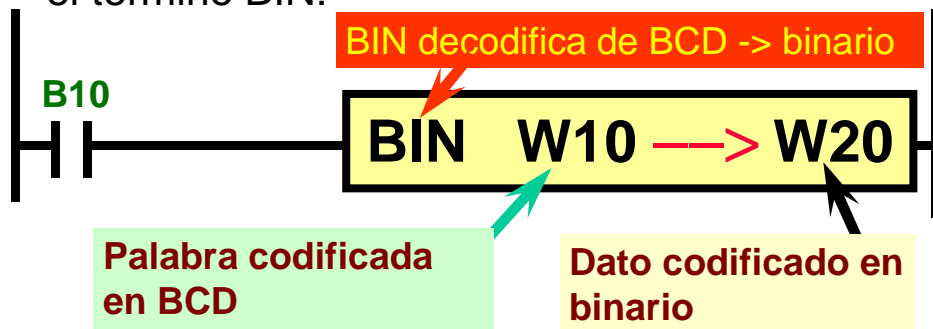
## BCD a BINARIO



El operador debe ver las ruedas en orden natural, para los cual el conexionado debe considerar el orden del bit menos significativo al más significativo.

## Transferencia de BCD a BINARIO

LA codificación de una palabra en binario a partir de su valor en BCD se obtiene utilizando un bloque de operación de transferencia con el termino BIN:



## BCD a BINARIO



En la palabra W10 se encuentra en BCD el valor de 2961.

W10 

		2			9			6			1				
0	0	1	0	1	0	0	1	0	1	1	0	0	0	0	1

 En BCD

BIN convierte este dato a un valor binario, que corresponda al numero 2961 y almacenarlo en la palabra W20:

W20 

0	0	0	0	1	0	1	1	1	0	0	1	0	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 En binario

## Transferencia de BINARIO a BCD



De igual manera, la codificación de una palabra en BCD a partir de su valor en binario se obtiene agregando el termino BCD:

BCD: decodifica de BINARIO -> BCD.





## BCD a BINARIO



En la palabra W3 se encuentra en binario el valor de 2961.

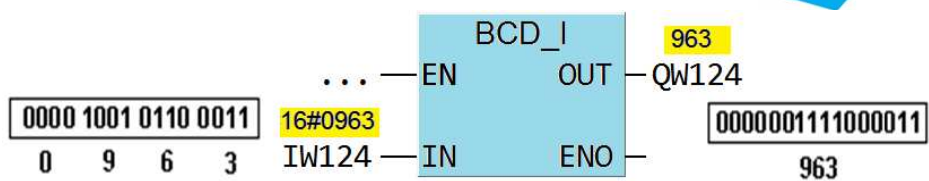
W3 0 0 0 0 1 0 1 1 1 0 0 1 0 0 0 1 En binario

BCD convierte de binario a BCD, el numero 2961 y lo almacena en la palabra W4:

W4 0 0 1 0 1 0 0 1 0 1 1 0 0 0 0 1 En BCD


2            9            6            1

## EJEMPLO EN PLC SIEMENS



	Address	Display	@Status value
1	IW 124	BIN	2#0000_1001_0110_0011
2	IW 124	DEC	2403
3	IW 124	HEX	W#16#0963
4	QW124	BIN	2#0000_0011_1100_0011
5	QW124	DEC	963
6	QW124	HEX	W#16#03C3





## OPERACIONES ARITMÉTICAS

**existen diferentes aplicaciones donde se tiene que sumar valores por ejemplo para totalizar la producción de los diferentes turnos de un día de trabajo. Restar valores para descontar los productos defectuosos, etc.**

**Las operaciones aritméticas entre palabras es transparente para los usuarios. Esto veremos a continuación.**

## OPERACIONES ARITMÉTICAS



### SUMA:

el valor de la palabra W1 se suma al valor de la palabra W2 y el resultado se transfiere a la palabra W3

operación  
de SUMA

Palabra  
destino

W1 + W2 → W3

## OPERACIONES ARITMÉTICAS



$$W1 + W2 = W3$$

W1 0 0 0 0 1 0 1 1 1 0 0 1 0 0 0 1 = 2961

+

W2 0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 1 = 651

=

W3 0 0 0 0 1 1 1 0 0 0 0 1 1 1 0 0 = 3612

## OPERACIONES ARITMÉTICAS



### RESTA:

Al valor de la palabra W1 se resta el valor de la palabra W2 y el resultado se transfiere a la palabra W3

operación  
de RESTA

Palabra  
destino

W1 - W2 → W3

## OPERACIONES ARITMÉTICAS



$$W1 - W2 = W3$$

W1 0 0 0 0 1 0 1 1 1 0 0 1 0 0 0 1 = 2961

-

W2 0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 1 = 651

=

W3 0 0 0 0 1 0 0 1 0 0 0 0 0 1 1 0 = 2310

## OPERACIONES ARITMÉTICAS



### MULTIPLICACIÓN:

el valor de la palabra W1 se multiplica por el valor de la palabra W2 y el resultado se transfiere a la palabra W3

operación de  
**MULTIPLICACIÓN**

Palabra  
destino

**W1 X W2 → W3**

## OPERACIONES ARITMÉTICAS



$$W1 \times W2 = W3$$

W1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 1 = 651

X

W2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 = 3

=

W3 0 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 = 1953

## OPERACIONES ARITMÉTICAS



### DIVISIÓN:

el valor de la palabra W1 se DIVIDE entre el valor de la palabra W2 y el resultado se transfiere a la palabra W3

operación  
de división

Palabra  
destino

W1 / W2 → W3

## OPERACIONES ARITMÉTICAS



$$W1 / W2 = W3$$

W1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 1 = 651

/

W2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 = 3

=

W3 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 = 217

## OPERACIONES ARITMÉTICAS



### REM:

El resto de la división de la palabra W1 entre el valor de la palabra W2 se registra en la palabra W3

operación de resto

Resto de la división

W1 REM W2 → W3

## OPERACIONES ARITMÉTICAS



W1 REM W2 = W3

W1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 1 = 651

REM

W2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 = 3

=

W3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 = 0

## OPERACIONES ARITMÉTICAS



Todas las operaciones aritméticas se efectúan sobre números enteros incluidos entre:

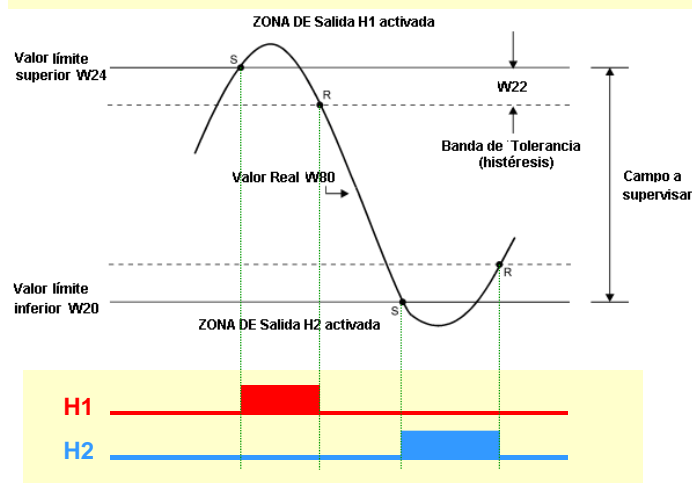
**- 32768** y **+ 32767**

El resultado de la operación debe estar incluido en el mismo intervalo.

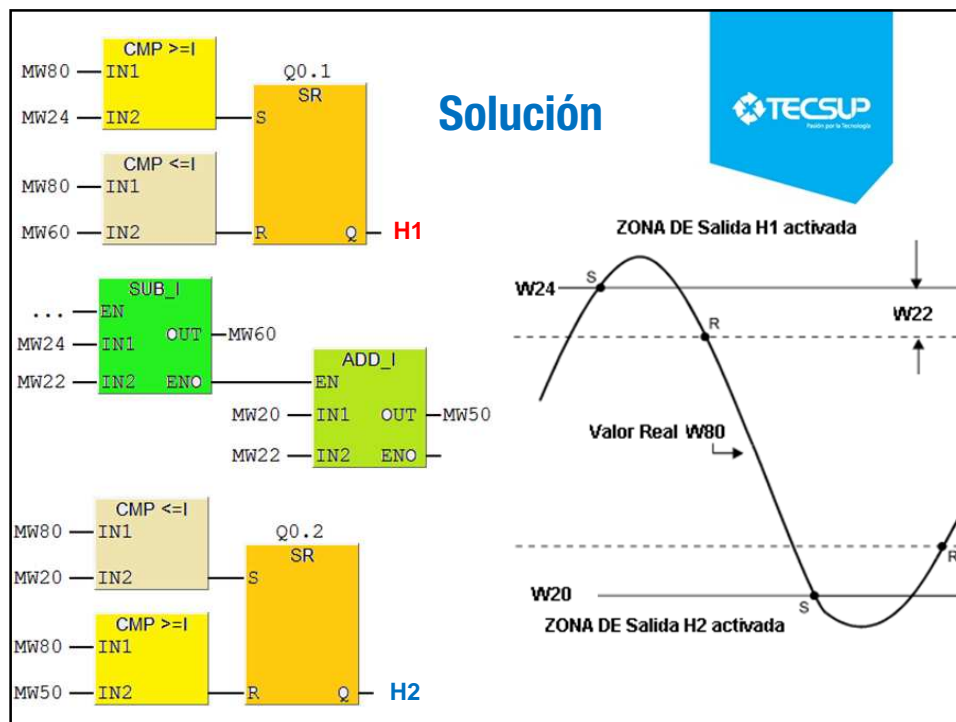
## OPERACIONES ARITMÉTICAS. Ejemplo



### VALORACIÓN DE LÍMITE CON HISTÉRESIS







## OPERACIONES DIGITALES



Las operaciones lógicas permiten efectuar entre palabras un:

**Y** lógico, **O** lógico, **O EXCLUSIVO** .

El resultado se transfiere a otra palabra o también a una cadena de bits. Esto veremos a continuación

## OPERACIONES DIGITALES



### AND:

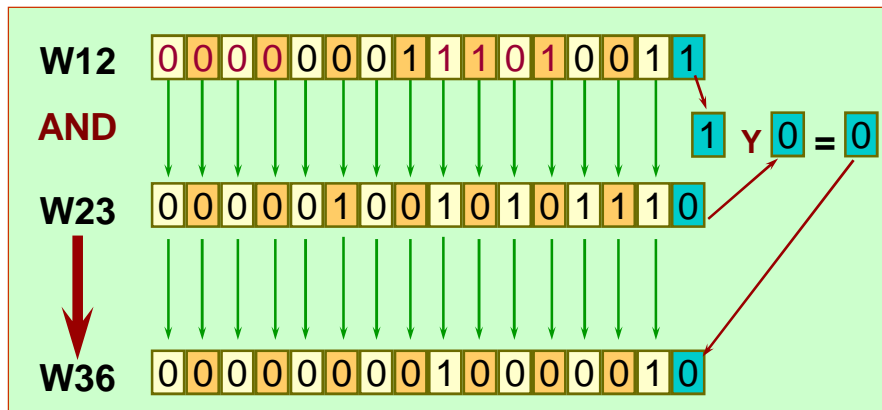
Ejecuta la operación lógica **AND (Y)** entre la cadena de bits de la palabra **W12** y la cadena de bits de la palabra **W23**. El resultado se transfiere a la cadena de bits de la palabra **W36**

**W12 AND W23 → W36**

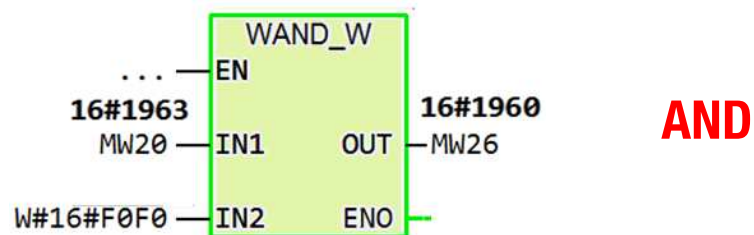
## OPERACIONES DIGITALES



La operación AND es bit a bit



## OPERACIONES ARITMÉTICAS



Path: PLC_IL_LAB_1\SIMATIC 300(1)\CPU 313C-2 DP					
	Address	Display format	@Status value	Modify value	
1	MW 20	BIN	2#0001_1001_0110_0011	2#0001_1001_0110_0011	
2	MW 26	BIN	2#0001_0000_0110_0000		
3	MW 20	HEX	W#16#1963		
4	MW 26	HEX	W#16#1060		

## OPERACIONES DIGITALES



### OR:

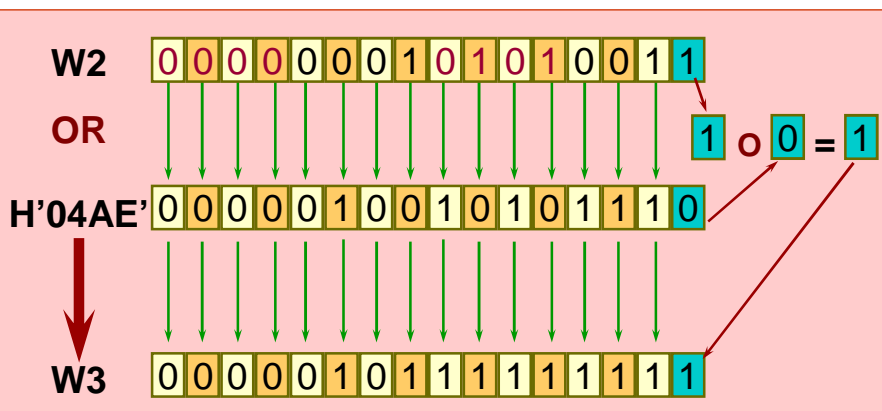
Ejecuta la operación lógica **OR (O)** entre la cadena de bits de la palabra **W2** y la cadena de bits del valor hexadecimal **H'04AE'**. El resultado se transfiere a la cadena de bits de la palabra **W3**.

**W2 OR H'04AE' → W3**

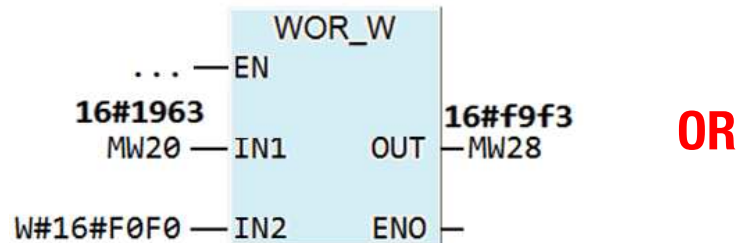
## OPERACIONES DIGITALES



La operación OR es bit a bit



## OPERACIONES ARITMÉTICAS



Path: PLC\_IL\_LAB\_1\SIMATIC 300(1)\CPU 313C-2 DP

	Address	Display format	@Status value	Modify value
1	MW 20	BIN	2#0001_1001_0110_0011	2#0001_1001_0110_0011
2	MW 28	BIN	2#1111_1001_1111_0011	
3	MW 20	HEX	W#16#1963	
4	MW 28	HEX	W#16#F9F3	

## OPERACIONES DIGITALES



### XOR:

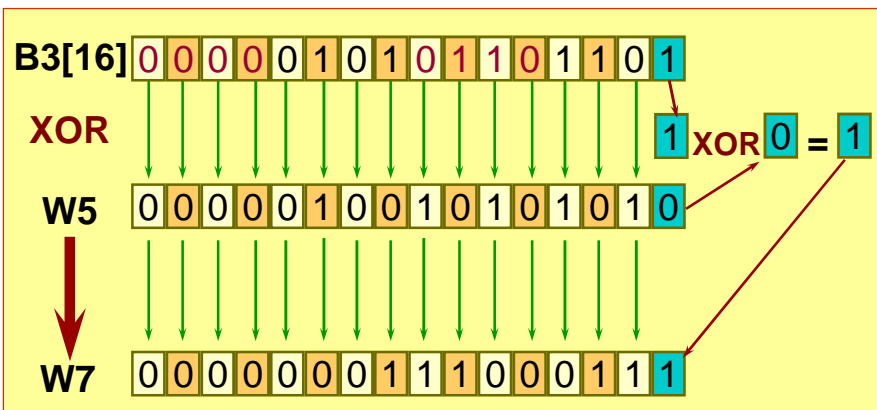
Ejecuta la operación lógica **XOR (0 exclusiva)** entre la cadena de bits **B3[16]** y la cadena de bits de la palabra **W5**. El resultado se transfiere a la cadena de bits de la palabra **W7**.

**B3[16] XOR W5 → W7**

## OPERACIONES DIGITALES



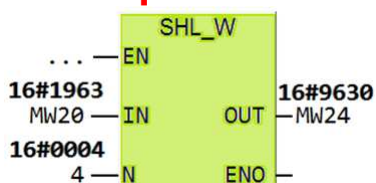
La operación XOR es bit a bit



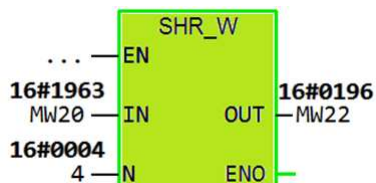
## FUNCIÓN DE CORRIMIENTO



Izquierda



Derecha



	Address	Display	@Status value	Modify value
1	MW 20	BIN	2#0001_1001_0110_0011	2#0001_1001_0110_0011
2	MW 22	BIN	2#0000_0001_1001_0110	
3	MW 24	BIN	2#1001_0110_0011_0000	
4	MW 20	HEX	W#16#1963	
5	MW 22	HEX	W#16#0196	
6	MW 24	HEX	W#16#9630	

## TIPOS DE DATOS



Denominación	Bits	Ejemplo	Descripción
BOOL	1	FALSE o TRUE	Variable binaria o lógica ( <i>Boolean</i> )
INT	16	-32768 .. 32767	Número entero con signo ( <i>Integer</i> )
DINT	32	$-2^{31} .. +2^{31} - 1$	Número entero doble con signo
REAL	32	0.4560	Número real
BYTE	8	0 .. 255	Conjunto de 8 bits
WORD	16	0 .. 65535	Conjunto de 16 bits
DWORD	32	$0 .. 2^{32} - 1$	Conjunto de 32 bits ( <i>Double Word</i> )
TIME	32	T#5d4h2m38s3.5ms	Duración
DATE	16	D#2002-01-01	Fecha
TIME_OF_DAY	32	TOD#15:35:08.36	Hora del día
S5TIME	16	S5T#2h2m38s	Duración
DATE_AND_TIME	64	DT#2002-01-01-15:35:08.36	Fecha y hora
CHAR	8	'A'	Carácter
STRING		'AUTOMATA'	Cadena de caracteres

Tabla 2.1. Principales tipos de datos del sistema de programación STEP7.

## IDENTIFICACIÓN DE VARIABLES:



	Normalizada IEC 1131-3	STEP7 inglesa (Internacional)	STEP7 alemana (SIMATIC)
<b>Variables predefinidas</b>			
Entradas	I, IX, IB, IW	I, IB, IW, ID	E, EB, EW, ED
Salidas	Q, QX, QB, QW, QD	Q, QB, QW, QD	A, AB, AW, AD
Marcas	M, MX, MB, MW, MD	M, MB, MW, MD	M, MB, MW, MD
<b>Operaciones lógicas básicas</b>			
Carga inicial	LD	A ó O	U ó O
Y	AND	A	U
NO-Y	ANDN	AN	UN
O	OR	O	O
NO-O	ORN	ON	ON
O-exclusiva	XOR	X	X
NO-O-exclusiva	XORN	XN	XN
<b>Operaciones con paréntesis</b>			
Y	AND(	A(	U(
NO-Y	ANDN(	AN(	UN(
O	OR(	O(	O(
NO-O	ORN(	ON(	ON(
O-exclusiva	XOR(	X(	X(
NO-O-exclusiva	XORN(	XN(	XN(
Cerrar paréntesis	)	)	)
<b>Terminar una cadena lógica</b>			
Asignar	ST	=	=
Desactivar	R	R	R
Activar	S	S	S
<b>Operaciones con flancos</b>			
Flanco negativo	LDF, ORF, ANDF	FN	FN
Flanco positivo	LDR, ORR, ANDR	FP	FP

Tabla 2.3. Denominaciones normalizadas IEC, STEP7 inglesa y STEP7 alemana, de las

## IDENTIFICACIÓN DE VARIABLES:



### *Variables de salida internas $M\ n.m$*

El término  $M$  (Marca) representa una variable lógica interna (elemento de memoria) y  $n$  y  $m$  tienen la misma correspondencia que en el caso de las entradas y salidas.

Las variables predefinidas pueden ser lógicas (bits) ( $X$  en los bits de los bloques de datos DB), octetos (B) [Bytes (8 bits)], palabras de 16 bits (W) (Words) y dobles palabras de 32 bits (DW) (Double Words), que constituyen datos del sistema de programación STEP7 (Tabla 2.1). En la tabla 2.4 se resumen los tipos de variables predefinidas de STEP7.

Entradas	Desde E 0.0 hasta E 65535.7
Salidas	Desde A 0.0 hasta A 65535.7
Marcas	Desde M 0.0 hasta M 65535.7
Temporizadores	Desde T 0 hasta T 65535
Contadores	Desde Z 0 hasta Z 65535

**Tabla 2.4.** Variables predefinidas del lenguaje de lista de instrucciones de STEP7.



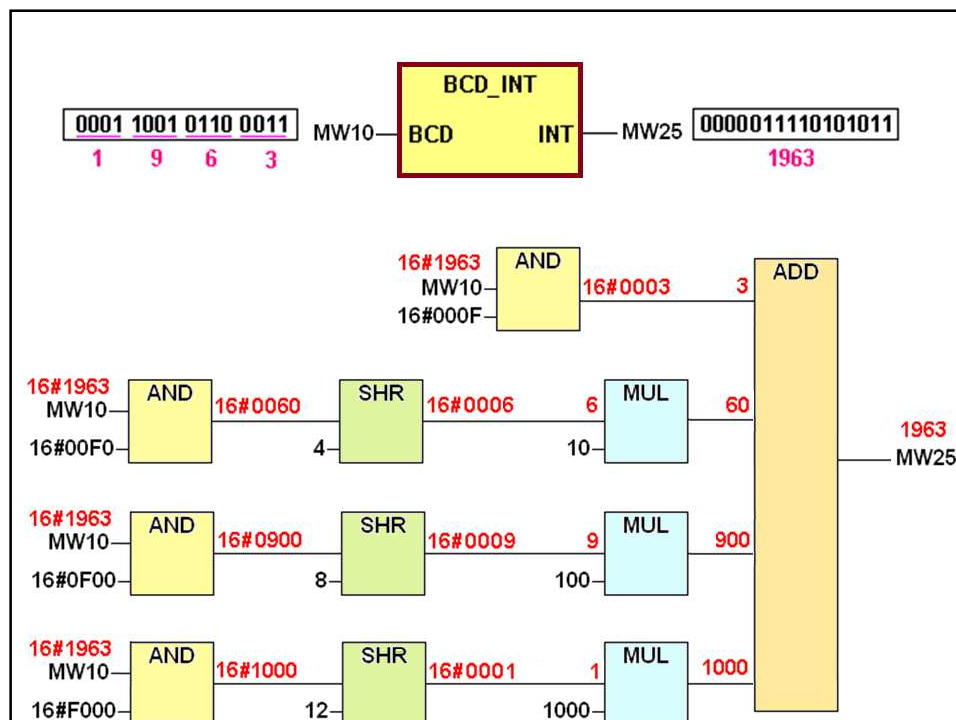
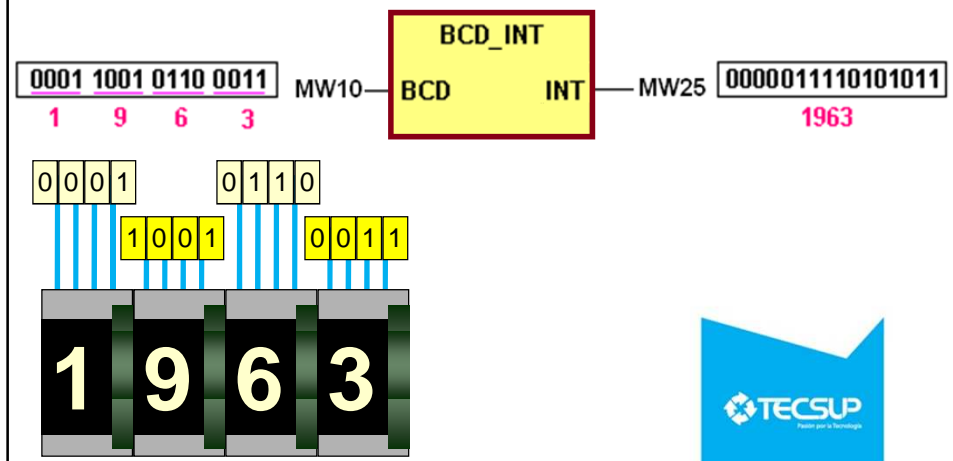
## Ejemplos Académico

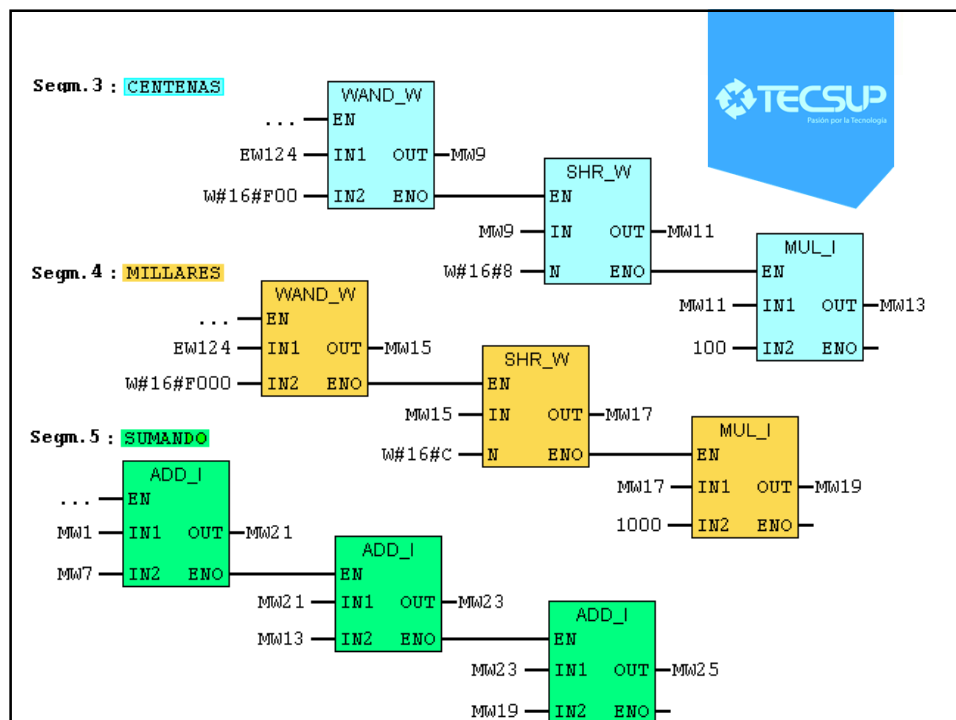
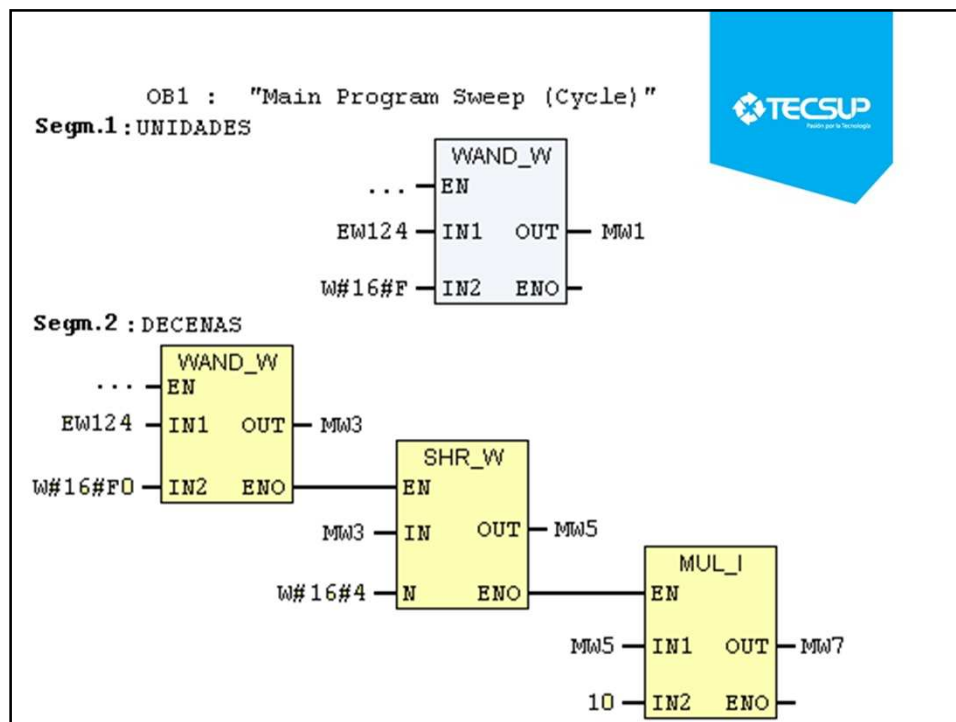
[www.tecsup.edu.pe](http://www.tecsup.edu.pe)



## 1. Conversión BCD a INT

Usando las funciones vistas hasta ahora Realizar un bloque de Conversión BCD a INT de 4 dígitos.

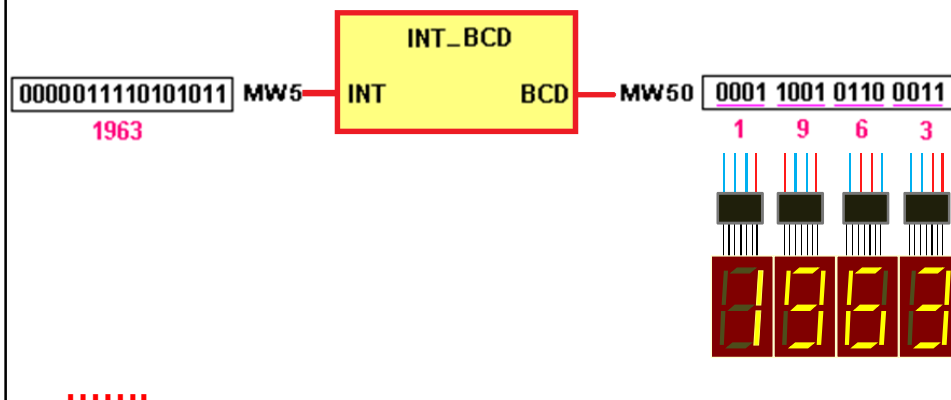




## 2. Problema planteado: Conversión INT a BCD



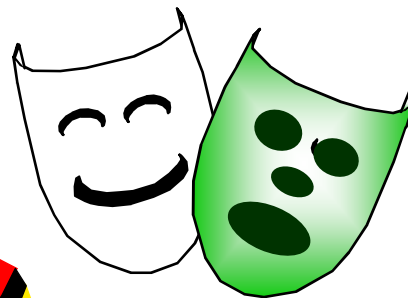
Usando las funciones vistas hasta ahora realizar un bloque de Conversión INT a BCD:



## 3. ENMASCARAR

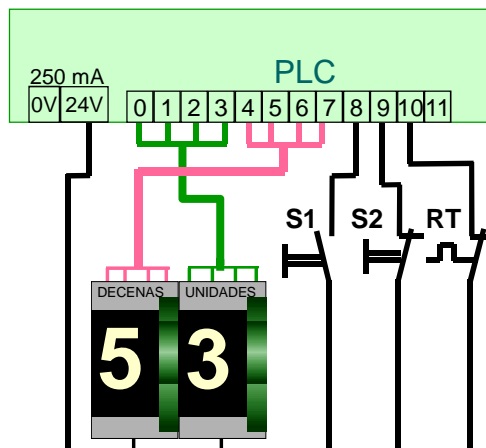


Dejar ver lo que se desea



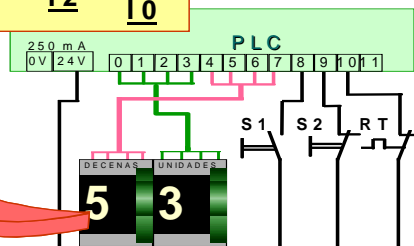
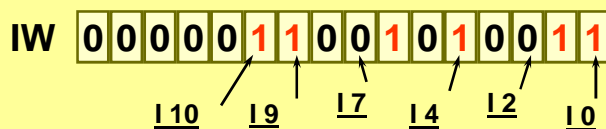
### 3. ENMASCARAR

Se tiene instaladas 2 ruedas codificadoras, los pulsadores S1, S2 y un contacto del relé térmico.



### 3. ENMASCARAR

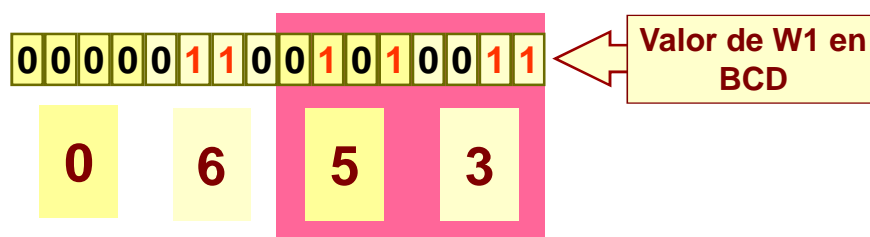
Solo me interesa leer el valor de las ruedas codificadoras.



### 3. ENMASCARAR



De la palabra IW solo nos interesa los 8 primeros bits. Pues si leo en BCD tendría el valor 653, que no corresponde al valor ingresado con las ruedas.

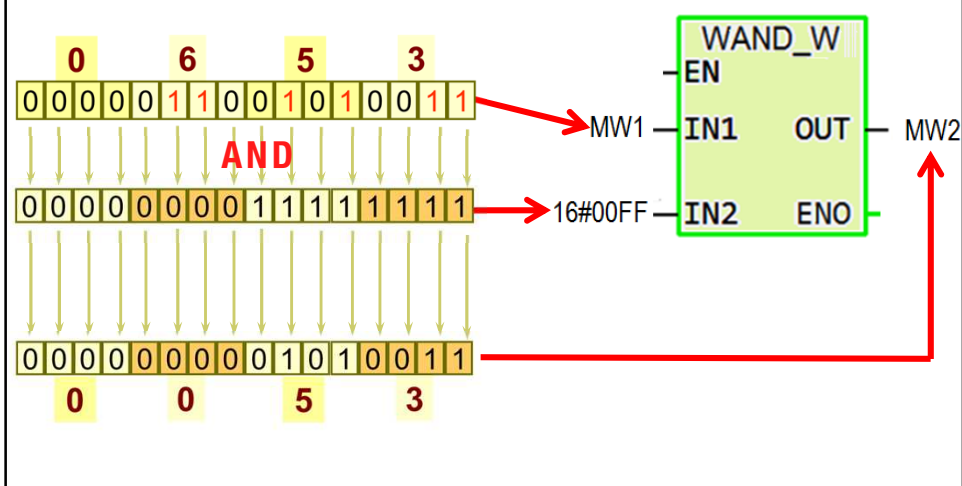


### ENMASCARAR



Esto implica usar la operación lógica AND con un valor constante (generalmente introducido en hexadecimal), de tal manera que los bit que no nos interesa se conviertan en ceros

### 3. ENMASCARAR

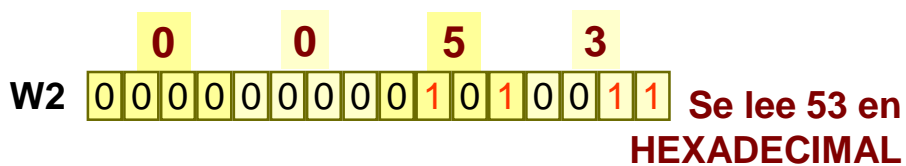


### EJEMPLO APLICATIVO



La palabra W2 contiene la combinación de ceros y unos que ingresaron por las ruedas codificadoras en BCD.

Si visualizamos el valor en HEXADECIMAL, veremos **0053**.



## EJEMPLO APLICATIVO



Pero si lo vemos como valor decimal, será **1619**. Esto es por que el PLC, interpreta esta combinación de ceros y unos, por defecto como binario

Se lee 53 en BCD

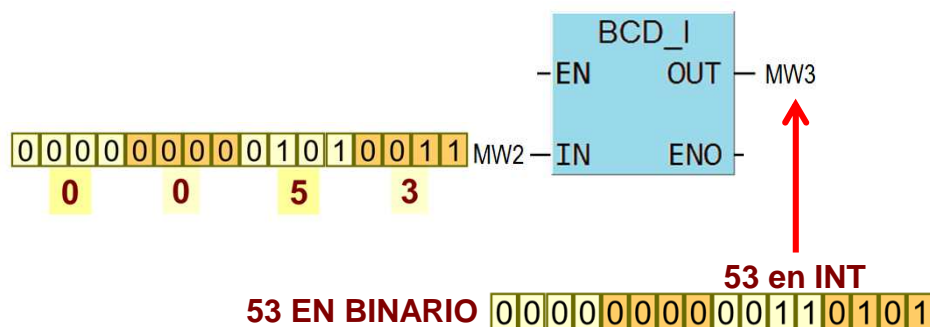
W2 00000000001010011 = 1619

Si no se hace algo este número es  
**1619**

## EJEMPLO APLICATIVO



Por lo tanto para tener el valor ingresado, se tendrá que usar el la función de conversión de BCD a BIN (**Binario**).



## Bibliografía

- Allen, Bradley (2001) Controllogix 5000 controllers common procedures programming manual. New York: Allen Bradley. (629.8PLC/A-2).
- Ramírez Quiroz, Elmer (1997) Controladores lógicos programables. Lima: CONCYTEC. (629.8PLC/R21)
- Siemens A.G. (1988) Autómata programable. (S5-100u) Simatic S5. Alemania.  
Siemens. (629.8PLC/S-199)



**Fin de la unidad**

