Sistemas numéricos

En este apéndice...

SISTEMAS NUMÉRICOS

Introducción a sistemas numéricos

En este apéndice describiremos algunos de los sistemas numéricos que son usados en los PLCs *Direct*Logic. Como cualquier PLC, éstos almacenan y manipulan números en forma binaria, esto es, sólo usan 1s y 0s, y entonces los unos y ceros tienen que tener una cierta convención para representar un número. Es por eso que debemos describir qué sistemas numéricos usan estos PLCs.

Describiremos el sistema decimal, el binario, el sistema octal, el sistema BCD, el sistema hexadecimal, los números reales con formato de punto flotante y el complemento de 2 y sus propiedades y como dato curioso el código Gray, que es usado con ciertos encoders.

Los PLCs ofrecen una cantidad fija de recursos dependiendo del modelo y de la configuración. Usamos la palabra "recursos" para incluir memoria variable, puntos de entradas y salidas, temporizadores, contadores, etc.

Los PLCs *Direct*Logic usan grupos de entradas y salidas en grupos de 8, es decir, todos los recursos de los PLCs son contados en el sistema octal. Es fácil para computar que los computadores cuenten en grupos de 8 porque 8 es una potencia de 2.

Sistema decimal

El sistema numérico decimal es el que se usa corrientemente y significa simplemente que se cuenta en grupos de 10 a un tiempo esto es 0 a 9, 10 a 19, 20 a 29, etc.

Este sistema es generado como una serie de números cuya base es el numero 10. Esto viene de la numeración arábiga que usaba el cero, como mejoría de la numeración romana.

Para formar el número 12345, lo que realmente se hace es:

$1x10^4 + 2x10^3 + 3x10^2 + 4x10^1 + 5x10^0 = 12345$

Llamamos **peso** al número que multiplica cada dígito por la potencia de 10. Esto es, el peso del digito más a la derecha (el menos significativo) es 1, el peso del segundo digito hacia la izquierda es 10; el tercer digito es 100, y así sucesivamente. Este concepto se usará posteriormente.

Esta explicación es necesaria para entender como funcionan los otros sistemas numéricos.

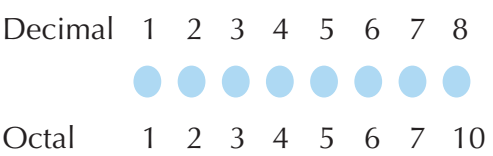
Sistema numérico octal

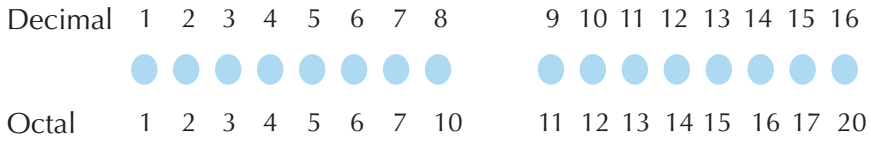
El sistema numérico OCTAL significa simplemente que se cuenta en grupos de 8 a un tiempo esto es 0 a 7, 10 a 17, 20 a 27, etc. usando el 8 como base.

Esto es, un número octal estará formado como sigue:
$1x8^4 + 2x8^3 + 3x8^2 + 4x8^1 + 5x8^0$ que en la numeración decimal es

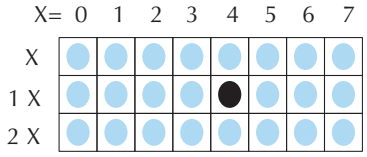
$4096 + 1024 + 192 + 32 + 5$ que equivale a 5357

Note que el peso es basado en una potencia de 8.





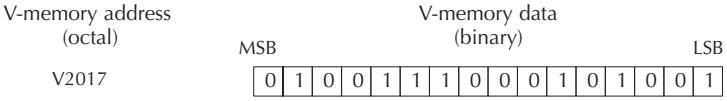
Vea la figura adyacente, donde tenemos dos grupos de 8 círculos. Contando en octal los círculos tenemos "2 0" círculos significando dos grupos de 8 y 0 grupos individuales.



Ahora los recursos del PLC se cuentan de 0 hasta siete que resulta también en un grupo de 8, el número 0 significa algo a un computador de modo que no lo saltamos; contamos de 0 a 7 y ahí entenderá que si éstos fueran contadores CT14 correspondería a lo que es la localización del círculo negro.

Memorias de datos y memorias para configuración del PLC

Las direcciones de memoria usan el mismo sistema octal, por ejemplo V2073 es una localización o dirección válida mientras que V1983 no es válida (9 y 8 no son dígitos válidos octales)

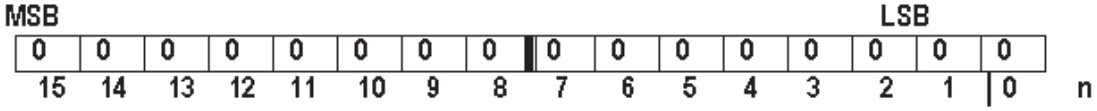


Cada localización de memoria tiene una palabra de datos y cada palabra contiene 16 bits.

Cada palabra tiene 2 **bytes,** es decir 2 grupos de 8 bits cada uno.

Cada byte tiene 2 **nibbles**, es decir, 2 grupos de 4 bits cada uno.

Los bits se muestran diagramáticamente en el diagrama abajo y el bit menos significativo (LSB) estará a la derecha, y el bit más significativo (MSB) a la izquierda. Usamos la palabra más significativa refiriéndose al peso de cada bit.



El peso de cada tipo es una característica que permite determinar el valor equivalente decimal, siendo el sistema decimal el sistema numérico que usamos los humanos.

Los datos de memorias de 16 bits son almacenados en forma binaria, pero nosotros raramente programamos las memorias de datos colocando cada bit. En vez de eso, usamos instrucciones o usamos herramientas de visualización que nos permiten trabajar con números binarios, números decimales, números BCD, números hexadecimales o incluso números reales de punto flotante. Todos estos números son convertidos y almacenados como binarios, es decir un conjunto de ceros y unos.

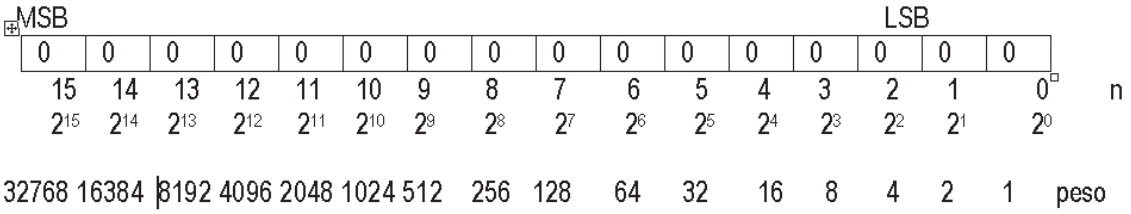
Una pregunta normalmente es, como se sabe si son números binarios, octales, decimales o hexadecimales? La respuesta es que no podemos decir cual numero es sólo mirando al conjunto de ceros y unos, pero realmente no es importante. Lo importante es que la fuente o el

mecanismo que escribe los datos en una localización o una dirección de memoria y el objeto que luego lee ellos deben tener el mismo tipo de datos o usar la misma convención. La localización de memoria es solamente una caja de almacenamiento, no convierte o mueve datos por sí mismo.

Sistema numérico binario

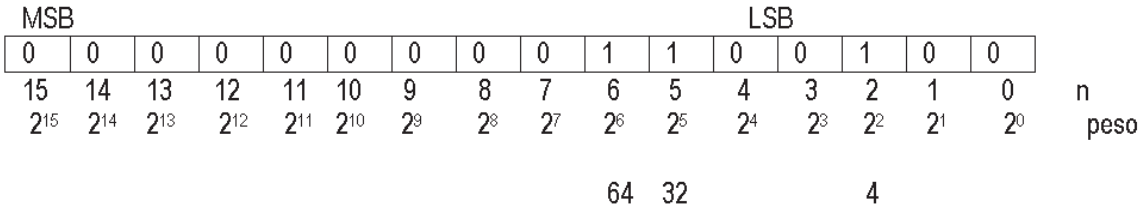
Un número en base 2 puede expresar cualquier número equivalente en decimal. Describimos como número binario a un número que es compuesto de 16 bits, cada uno teniendo un **peso** de **2** elevado a **n** siendo **n** la posición relativa de los bits; vea en la figura de abajo más explicaciones.

Un numero binario es formado por la suma de los valores que contienen un 1 en el bit correspondiente.



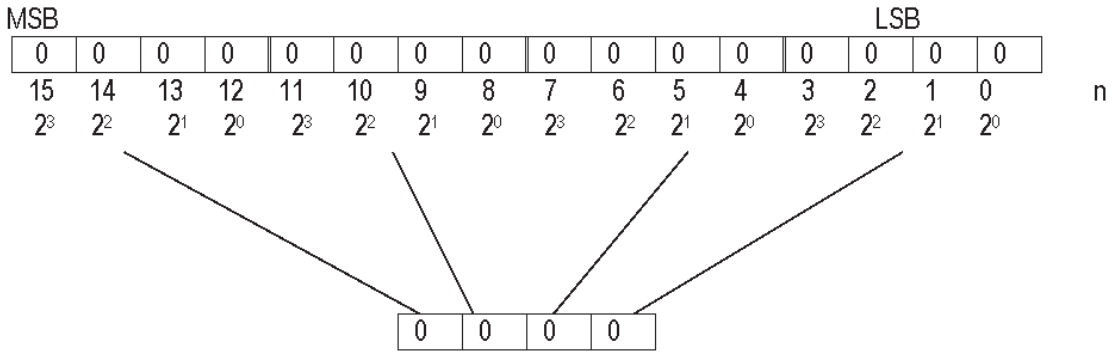
Por ejemplo, el número 100 decimal corresponde a un número en bits con un 1 en los bits 2, 5 y 6, como muestra el diagrama abajo. Aquí se suma 64+32+4=100. Los bits que son 0 no se suman.

Un número binario también puede contener 15 bits, 12 bits, o un valor dado por convención.

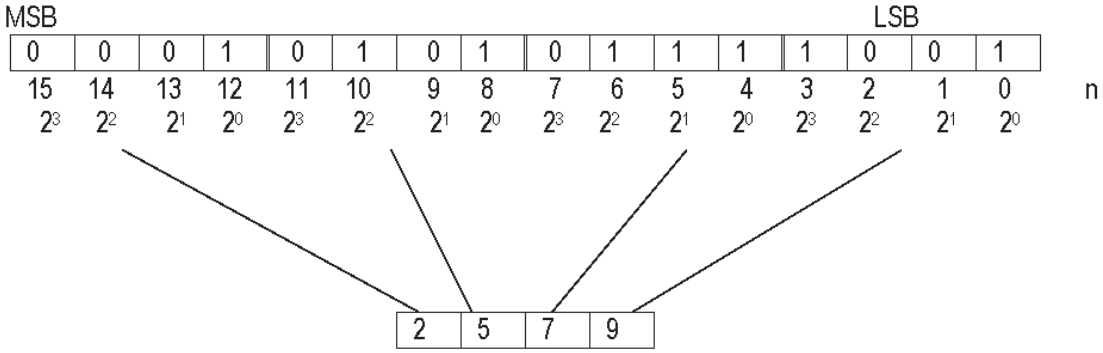


Sistema numérico BCD (Binary coded decimal)

Ya que los humanos naturalmente cuentan en el sistema numérico decimal, preferimos entrar y ver datos en el PLC en valores decimales. Sin embargo los computadores son más eficientes al usar números binarios puros. Una solución de compromiso entre los dos es la representación BCD. Un dígito BCD tiene el rango de 0 a 9 y es almacenado como cuatro bits (llamado un nibble). Esto permite que cada localización de memoria almacenen cuatro dígitos BCD, con un rango de números decimales de 0000 hasta 9999.



En un sentido binario puro, una palabra 16 bits representa un número de 0 a 65,536. Al almacenar números BCD, el rango es reducido a 0 hasta 9999. Muchas instrucciones aritméticas en el PLC usan datos BCD en que el rango es reducido a 0 a 9999. Muchas instrucciones aritméticas usan datos BCD e incluso *Direct*SOFT nos permite entrar y ver datos en el sistema BCD. *Direct*SOFT tiene instrucciones que nos permiten convertir de BCD a binario o viceversa.Vea en el ejemplo siguiente como es representadop el número 2579/



Sistema numérico hexadecimal

Los números hexadecimales son similares a los números BCD, excepto que ellos utilizan todos los números binarios en cada nibble. Estos son números en base 16 de modo que se necesitan 16 dígitos diferentes. Para extender los números decimales de 0 a 9 se usan las letras A hasta F , como se muestra abajo

Decimal 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Hexadecimal 0 1 2 3 4 5 6 7 8 9 A B C D E F

Un numero hexadecimal de 4 nibbles puede representar hasta 65536 valores en una palabra de 16 bits. El rango va desde 0000_h hasta $FFFF_h$, siendo el sufijo $_h$ la indicación de que el número está representado como hexadecimal. A veces los PLCs necesitan todo este rango, por ejemplo, para datos de sensores con señales análogas. El sistema hexadecimal es sólo una forma conveniente de ver datos almacenados en forma binaria.

Sistema numérico real de punto flotante

El estándar de IEEE 754 de números de punto flotante es la representación más común hoy para números reales en computadoras, inclusive PCs basados en Intel, Macintoshs, y la mayoría de las plataformas Unix.

Hay varias maneras de representar los números reales en computadoras. El sistema de coma fija (o punto como es usado en USA) coloca una coma en algún lugar entre los dígitos, por convención y es equivalente a usar enteros que representan las porciones de alguna unidad. Por ejemplo, uno quizás represente un centécimo de una unidad; si usted tiene cuatro dígitos de decimal, usted podría, por ejemplo, representar 10,82, o 00,01.

Otro enfoque seria usar números racionales, y representar cada número como la razón de dos enteros.

La representación del punto flotante representa básicamente un número real en notación científica. La notación científica representa los números como un número base y un exponente.

Por ejemplo, 123,456 podría ser representado como 1,23456 X 10^2. En hexadecimal, el número 123, ABC podría ser representado como 1,23ABC x 16^2.

¿Cual es la convención con IEEE para punto flotante de 32 bits?

Los números de punto flotante tienen tres componentes básicos: el signo, el exponente, y la mantisa. La mantisa se compone de una fracción y un dígito delantero implícito (explicado abajo). La base (2) del exponente es implícita y no hay necesidad de almacenarla. La tabla siguiente muestra la disposición los valores de punto flotantes de precisión de 32 bits. Se muestra el número de bits para cada campo (los rangos de bits están en paréntesis cuadrados):

	Signo	Exponente	Fracción	Bias
Precisión de 32 bits	**1[31]**	**8[30-23]**	**23 [22-0]**	**127**

•El bit de signo

El bit del signo es muy sencillo. 0 denota un número positivo; 1 denota un número negativo.

•El exponente

El campo del exponente necesita representar tanto exponentes positivos y negativos. Para hacer esto, se añade un numero o bias al exponente verdadero para obtener el exponente almacenado. Para precisión de 32 bits, este valor es 127. Así, un exponente de cero significa que se almacena 127 en el campo del exponente. Un valor almacenado de 200 indica un exponente de (200-127), o 73. Para razones no discutidas aquí, los exponentes de -127 (todos 0s) y +128 (todos 1s) son reservados para números especiales.

•La mantisa

La mantisa representa los bits de precisión del número. Se compone de un bit delantero implícito y los bits de la fracción. Para averiguar el valor del bit delantero implícito, considera que cualquier número se puede expresar en notación científica de muchas maneras diferentes. Por ejemplo, el número cinco pueden ser representado como cualquiera de éstos:

$$5{,}00 \text{ X } 10^0$$

$$0{,}05 \text{ X } 10^2$$

$$5000 \text{ X } 10^{-3}$$

Para llevar al máximo la cantidad de números representables, los números de punto flotante se almacenan típicamente en forma normalizada. Esto pone básicamente la coma después del primer dígito que no sea cero. En forma normalizada, cinco es representado como 5,0 X 10^0.

Una optimización gradable está disponible en base dos, ya que el único dígito que no es cero posible es 1. Así, acabamos de asumir un dígito delantero de 1, y no necesitamos representarlo explícitamente. Como resultado, la mantisa tiene efectivamente 24 bits de resolución, y 23 bits de fracción.

Para hacer un resumen:

1.El bit del signo es 0 para positivo, 1 para negativo.

2.La base del exponente es dos.

3.El campo del exponente contiene 127 más el exponente verdadero con precisión de 32 bits

4.El primer bit de la mantisa se asume típicamente ser 1, f, donde f es el campo de bits de fracción.

Rangos de números de punto flotante

Consideremos por un momento números de punto flotante de precisión de 32 bits. Note que tomamos esencialmente un número de 32 bits y re-distribuimos los campos para cubrir un rango más amplio. Algo tiene que ceder, y es la precisión. Por ejemplo, enteros de 32 bits regulares, con toda precisión centrada en cero, puede almacenar precisamente enteros con 32 bits de resolución. Un número de punto flotante de precisión de 32 bits, por otro lado, es incapaz de lograr esta resolución con 24 bits. Sin embargo, se aproxima este valor al truncar los valores más bajos. Por ejemplo:

11110000 11001100 10101010 00001111 //entero de 32 bits =

+1,1110000 11001100 10101010 x 231de número de punto flotante de precisión de 32 bits

= 11110000 11001100 10101010 00000000 // valor correspondiente

Esto se aproxima al valor de 32 bits, pero no nos da una representación exacta. Por otro lado, además de la habilidad de representar los componentes fraccionarios (que los enteros no pueden hacer), el valor de punto flotante puede representar números alrededor de 2127, comparado con el valor de 32 bits del máximo de enteros alrededor de 232.

Mucho más se podría hablar sobre este sistema, pero en este libro eso es lo suficiente para los propósitos a ser alcanzados. Note que en realidad no es necesario conocer la convención ya que las operaciones toman cuenta de los cálculos en forma transparente.

Número en representación Gray

El código Gray es una sucesión binaria con la propiedad que sucede sólo un cambio de bit entre cualquiera de dos elementos consecutivos.

El código Gray se puede usar para convertir la posición angular de un disco a la forma digital (con un encoder, por ejemplo). Una línea radial de sensores lee el código desde la superficie del disco y si el disco está en el medio entre dos posiciones, cada sensor quizás lea su bit de ambas posiciones al mismo tiempo pero ya que sólo hay un bit de diferencia entre entre las dos posiciones, el valor leído es garantizado ser uno de los dos valores válidos antes que alguna combinación de un tercero (inválido).

Un algoritmo posible para engendrar una sucesión de código Gray deberá cambiar un bit que tiene como resultado un código nuevo cada vez. Aquí está una sucesión Gray de cuatro bits de código engendrada de esta manera:

Decimal	No. Gray
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111

6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

Algunos encoders usan la convención Gray para entregar la información y entonces es necesario entender que es este sistema numérico para poder operarlo adecuadamente. Varios de los PLCs DirectLogic tienen una instrucción que maneja este código.

Valores numéricos en módulos análogos

Volviendo al sistema BCD o binario, Ud. puede haber entrado la configuración de un módulo análogo como número BCD, o Ud. puede configurar los módulos para traer los valores en un sistema numérico binario. Todo lo que Ud. debe saber es que, si Ud. usará este valor de este módulo análogo como un valor prefijado de temporizador o de un contador, los temporizadores y los contadores funcionan con BCD.

De modo que si va a usar el valor como un valor prefijado de un temporizador, los va a tener que entrar como BCD. Pero si ellos entran como binario, la CPU tiene una conversión dentro para convertir un valor binario en BCD. Esto se usa para convertir el modelo binario para hacerlo el modelo correcto. Es su preferencia de cómo Ud. quiere que ellos entren.

Ahora hay alguna confusión entre clientes. Cuándo usa software de *Direct*SOFT y Ud. va a Data View, si Ud. baja el menú y Ud. lo pide en binario, *Direct*SOFT le dará ceros y unos. Le dará una serie de ceros y unos, le mostrará exactamente lo que se está en esa palabra. Si Ud. teclea la flecha hacia abajo y Ud. escoge tipo BCD/Hex sumará toda la serie de ceros y unos en la palabra que Ud. seleccionó y le dirá el valor en un número BCD. Le dará de vuelta el número a que equivalen estos bits. El PLC cuida de la operación para Ud.

Ud. no tiene que recordar estos sistemas numéricos semejantes a éstos. La computadora lo hace por Ud. Los suma. Si trata de evaluar una palabra en un sistema numérico binario que usa 0s y 1s, (que *Direct*SOFT se refiere a como sistema numérico decimal).

Si escoge binario en Data View de DirectSOFT, Ud. verá 0s y 1s. Si escoge decimal va a ver el equivalente de decimal, equivalente a ceros y unos en formato binario.

Representación del complemento de 2

Veamos a continuación la definición de lo que es el complemento de 2, cómo calcular este, como sumarlos, como restarlos, como multiplicarlos y como dividirlos.

La representación del complemento de 2 es usada por números con signo en la mayoría de los computadores. Esta notación le permite a un computador sumar y restar números usando la misma operación (de modo que no es necesario implementar sumadores y restadores). En este caso se necesita tener un numero fijo de bits y el bit más significativo es el bit de signo. Esta misma representación es usada para representar números positivos y negativos.

Propiedades

La representación del complemento de 2 permite el uso de operaciones aritméticas binarias en enteros con signo, permitiendo resultados correctos del complemento de 2's.

Los números positivos del complemento de 2's se representan como números binarios simples.

Los números negativos del complemento de 2's se representan como números binarios de tal modo que cuando sumados a un número positivo de la misma magnitud resulta en un cero.

Vea la tabla a continuación para entender el concepto: Es este caso se hace con 8 bits, pero puede hacerse con la cantidad de bits que sea conveniente o que se desee.

Número Entero Complemento de 2

Con signo		Sin Signo
5	5	0000 0101
4	4	0000 0100
3	3	0000 0101
2	2	0000 0010
1	1	0000 0001
0	0	0000 0000
-1	255	1111 1111
-2	254	1111 1110
-3	253	1111 1101
-4	252	1111 1100
-5	251	1111 1011

Nota: el bit más significativo (al extremo izquierdo) indica el signo del entero; por lo tanto se llama a veces el bit del signo

Si el bit del signo es cero => entonces el número es mayor que o es igual a cero, o positivo.

Si el bit del signo es uno => entonces el número es menor que cero, o negativo.

El cálculo del complemento de 2

Para calcular el complemento de 2 de un entero, invierta el equivalente binario del número cambiando todos los unos a ceros y todos los ceros a unos (llamado también el complemento de 1's), y luego sume uno.

Por ejemplo,

0001 0001 (binario 17) => 1110 1111 (complemento de 2 -17)

invertir (0001 0001) => 1110 1110 (Invierta los bits)

1110 1110 + 0000 0001 = >1110 1111 (Sume 1)

Suma del Complemento de 2

Sigue las mismas reglas de la adición binaria.

Por ejemplo,

```
5 + (-3) = 2      0000 0101 = +5
                + 1111 1101 =  -3
                ---------------
                  0000 0010 = +2
```

Substracción del Complemento de 2

La resta del complemento de 2 es la adición binaria del minuendo al complemento del 2's del sustraendo (agregar un número negativo es lo mismo que restar un 1 positivo).

Por ejemplo,

```
7 - 12 = (-5)   0000 0111 =  +7
              + 1111 0100 = -12
                1111 1011 =  -5
```

Multiplicación del Complemento de 2

Sigue las mismas reglas de la multiplicación binaria.

Por ejemplo,

```
(-4) × 4 = (-16)      1111 1100 =  -4
                    × 0000 0100 = +4
                    ---------------
                      1111 0000 = -16
```

División del Complemento de 2

La división del complemento de 2 es la sustracción repetida del complemento de 2. Se calcula el complemento de 2 del divisor y luego es añadido al dividendo. Para el próximo ciclo de sustracción, el cociente reemplaza el dividendo. Esto se repite hasta que el cociente sea demasiado pequeño para la sustracción o sea cero, entonces llega a ser el resto.

La respuesta final es el suma de ciclos de sustracción más el resto.

Por ejemplo,

7 ÷ 3 = 2 resto 1	0000 0111 = +7	0000 0100 = +4
	+ 1111 1101 = -3	+ 1111 1101= -3
	0000 0100 = +4	0000 0001 = +1 (resto)

Para hacer la extensión del signo para extender un entero con signo de 8 bits a uno de 16 bits o de 16 bits a uno de 32 bits, añada los bits adicionales en el lado izquierdo del número. Llene cada bit extra con el valor del número más pequeño los bits más significativos (el bit del signo).

Por ejemplo,

Entero con signo		Representación de 8 bits	Representación de 16 bits
- 1	1111 1111	1111 1111	1111 1111 1111 1111
+1	0000 0001	0000 0000	0000 0000 0000 0001

Esperamos que este apéndice le deje claro los sistemas numéricos que son usados en los PLCs.

Se pueden consultar muchas más informaciones en libros especializados de matemáticas y debemos decir que mayores explicaciones están fuera del alcance de este manual.

Productos de AUTOMATIONDIRECT.COM y tipos de datos

PLCs *Direct*LOGIC

La familia de PLCs *Direct*LOGIC usa el sistema de numeración octal para toda las entradas y salidas. Todas las memorias V se almacenan en formato BCD a menos que sea cambiada específicamente por el programador. Esto significa que todas las operaciones aritméticas usadas en programas de lógica ladder se deben hacer con operadores BCD. Se tiene también la opción de usar números binarios o reales así como valores de doble palabra BCD en algunos de los modelos del PLC.

Para cambiar el tipo de formato los datos, usted debe usar una instrucción de programa. Por ejemplo, para cambiar un valor de formato BCD a formato binario, use una instrucción BIN. También, al cambiar el formato desde BCD a un número real o de punto flotante, use una instrucción BTOR.

Un número en formato BCD o un número en formato binario no se puede sumar a un número verdadero, o un número en formato BCD a un número de formato binario, y luego va a obtener un resultado correcto. Los formatos de datos deben ser iguales .

Éstos son algunos asuntos que debe saber en relación con que todos los datos de memoria V que están en formato BCD por defecto.

- Uno es que los módulos análogos se pueden configurar para dar resultados em formato binarios o en BCD, de modo que es necesario saber como se está usando el módulo.
- PID es otra área donde no todos los valores están en formato BCD. De hecho, casi todos los parámetros de la tabla de PID se almacenan en la memoria del PLC como números de formato binario.

NOTA: *El algoritmo de PID usa magnitud más signo para los números binarios negativos, mientras que las funciones estándares usan el complemento de dos. Esto puede causar confusión mientras se buscan errores de un lazo de PID.*

- Por último, cuando se usa está utilizando Data View en *Direct*SOFT.
  Valores en formato binario, hexadecimal y decimales se almacenan de la misma manera en el PLC y son todos son llamados formato binario. La única diferencia entre todos es siempre que se vean usando Data View. Asegúrerse que ha seleccionado el formato apropiado en la ventana de Data View. También note que el valor BCD está llamado BCD/Hex.

  Recuerde de la tabla 8 (página I-6), BCD y hexadecimal son realmente iguales aunque las letras A a F no están implicadas; comparten un formato de representación aunque los valores son diferentes. Aquí es donde es crucial la buena documentación del formato de datos almacenado en memoria.