



## GUIA RAPIDA DE INICIO

PLC KDN-K3 Serie 304/306

www.caipe.com

### Crear un proyecto en Easy Prog PLC KDN

Ejecutaremos el EasyProg correspondiente a la version de PLC que deseamos usar, actualmente existen dos versiones V1.9.3 / V1.9.5.

En **(File, New Project)** pondremos nombre a nuestro nuevo programa.

La programacion se podra realizar, tanto en lenguaje de Instrucciones, como en Ladder **(Project, LD /IL)**.

En el **(Workspace, Hardware)** eligiremos el modelo de CPU, con las expansiones a utilizar.

---

### Instrucciones basicas en Easy Prog (Resumido)

Logicas	Comparacion	Traslado	Aritmeticas	Control de Prog	Contadores	Temporizadores
LD LDN ST S R R_TRIG F_TRIF OR NOR AND NAND	EQ LT GT	MOVE	ADD SUB MUL DIV INC DEC	CAL	CTU CTD	TON TOF TP

LD: es la instruccion de carga de un bit.

LDN: instruccion inversa a LD, si el bit a cargar esta en "0" nos devolvera "1".

ST: almacena el resultado de la instruccion anterior en la variable asignada.

S: si el resultado de la instruccion anterior, fue "1" pondra en "1" la variable asignada.

R: si el resultado de la instruccion anterior, fue "1" pondra en "0" la variable asignada.

R\_TRIG: se pondra en "1" solo en el flanco ascendente de una variable.

F\_TRIF: se pondra en "1" solo en el flanco descendente de una variable.

OR: suma logica entre dos variables del tipo bit.

NOR: suma logica entre dos variables del tipo bit, niega el resultado.

AND: multiplicacion logica entre dos variables del tipo bit.

NAND: multiplicacion logica entre dos variables del tipo bit, niega el resultado.

EQ: si el valor entre dos bytes es igual, da "1".

LT: si el valor entre del primer byte es menor que el segundo, da "1".  
GT: si el valor entre del primer byte es mayor que el segundo, da "1".

MOVE: copia el valor de una variable del tipo byte o, doble byte; en otra del mismo tipo.

ADD: suma una constante, o el valor de una variable en otra.  
SUB: resta una constante, o el valor de una variable en otra.  
MUL: multiplica una constante, o el valor de una variable en otra.  
DIV: divide una constante, o el valor de una variable en otra.  
INC: incrementa en 1 el valor de una variable.  
DEC: decrementa en 1 el valor de una variable.

CAL: llama a una subrutina.

CTU: contador ascendente.  
CTD: contador descendente.  
#Ver Contadores#

TON: transcurrido su tiempo programado, arroja "1".  
TOF: transcurrido su tiempo programado, arroja "0".  
TP: mientras transcurre su tiempo programado da "1".  
#Ver Temporizadores#

Las instrucciones son condicionales, es decir, seran ejecutadas siempre que el resultado de la instruccion anterior de como resultado "1".

## Mapa de memoria (Resumido)

### Tipo de memoria, características:

<b>I (entradas digitales)</b>	
TIPO DE ACCESO  Aqui se describe solo el acceso por bit, pudiendose hacer tambien por byte, word, doble word.	<b>Por bit (solo leemos una entrada digital por vez)</b> <b>%Ix.y</b> <b>x</b> :byte de direccion de la variable <b>y</b> :bit del byte de la variable (entre 0~7)  <b>Ej: %I0.0, %I0.7, %I1.0</b>
TIPO DE DATO	BOOL
ACCESO CORRECTO	Solo lectura

### **Q (salidas digitales)**

TIPO DE ACCESO	<b>Por bit (solo leemos/escibimos una salida digital por vez)</b> <b>%Qx.y</b> x:byte de direccion de la variable y:bit del byte de la variable (entre 0~7) <b>Ej: %Q0.0, %Q0.7, %Q1.0</b>
Aqui se describe solo el acceso por bit, pudiendose hacer tambien por byte, word, doble word.	
TIPO DE DATO	BOOL
ACCESO CORRECTO	Lectura/escritura

<b>M (Area de memoria interna, propicia para la operacion con bits)</b>	
TIPO DE ACCESO	<b>Por bit ( leemos/escibimos un bit )</b> <b>%Mx.y</b> x:byte de direccion de la variable y:bit del byte de la variable (entre 0~7) <b>Ej: %M0.0, %M0.7, %M1.0</b>
Aqui se describe solo el acceso por bit, pudiendose hacer tambien por byte, word, doble word.	
TIPO DE DATO	BOOL
ACCESO CORRECTO	Lectura/escritura

<b>V (Area de memoria interna, usada para almacenar gran cantidad de datos)</b>	
TIPO DE ACCESO	<b>Por bit ( leemos/escibimos un bit )</b> <b>%Vx.y</b> x:byte de direccion de la variable y:bit del byte de direccion de la variable(entre 0~7) <b>Ej: %V0.0, %V0.1, %V4.6</b>  <b>Por byte ( leemos/escibimos un byte )</b> <b>%VBx</b> x:byte de direccion de la variable <b>Ej: %VB0, %VB1, %VB14</b>  <b>Por word ( leemos/escibimos una word )</b> <b>%VWx</b> x:byte de comienzade direccion de la variable (debe ser numero par) <b>Ej: %VW0, %VW2, %VW14</b>  <b>Por dobleWORD, dobleINT ( leemos/escibimos una dWord o dINT )</b> <b>%VDx</b> x:byte de comienzo de direccion de la variable (debe ser par y ocupa 4 bytes) <b>Ej: %VD0, %VD4, %VD14</b>
TIPO DE DATO	DWORD, DINT: REAL (x debe ser en un rango especifico ver nota sobre numeros reales )
ACCESO CORRECTO	Lectura/escritura

### *Algunos ejemplos*

Logica Bit	Operaciones Logicas	Comparaciones	Operaciones Algebraicas
LD %I0.0 ST %Q0.0	LD %I0.0 OR %I0.1 ST %Q0.0	LD %I0.0 EQ %VW0, %VW2 ST %Q0.0	LD %I0.0 ADD 45, %VW2
LD %I0.1 S %Q0.1	LD %I0.0 AND %I0.1 S %Q0.0	LD %I0.0 LT %VW0, %VW2 S %Q0.0	LD %I0.0 ADD 45.38, %VD4000
LD %I0.2 R %Q0.1	LD %I0.0 ORN %I0.1 S %Q0.0	LD %I0.0 GT %VW0, %VW2 R %Q0.0	LD %I0.0 SUB 45, %VW8 R %Q0.0
LD %M0.0 ST %M0.1	LD %M0.0 ST %M0.1		

### USO DE ETIQUETAS

MAIN *	VAR_GLOBAL	ini	Hardware	
	Symbol	Address ▾	Data Type	Comment
1	marcha	%I0.0	BOOL	
2	parada	%I0.1	BOOL	
3	aux1	%M0.0	BOOL	
4	aux8	%M0.7	BOOL	
5	aux9	%M1.0	BOOL	
6	salida1	%Q0.0	BOOL	
7	salida2	%Q0.1	BOOL	
8	var1	%VD4000	REAL	
9	var2	%VD4004	REAL	
10	tiempo1	%VW0	INT	
▶ 11	tiempo2	%VW2	INT	

Se facilita el uso de variables, quedando de la sig. manera.

LD marcha  
ST salida1

LD parada  
S salida2

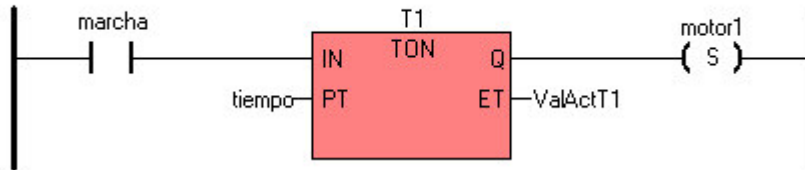
LD aux1  
ST aux8

## TEMPORIZADORES Y CONTADORES

		CPU 304	CPU 306
<b>Temporizadores</b>	Cantidades	64	128
	Nombres	T0--T63	T0--T127
	Base de Tiempo	T0--T3 1ms T4--T19 10ms T20--T63 100ms	T0--T3 1ms T4--T19 10ms T20--T127 100ms
	Tiempo Maximo	32767	32767
<b>Contadores</b>	Cantidades	64	128
	Nombres	C0--C63	C0--C127
	Cuenta Maxima	32767	32767

### Temporizador:

(\* Network 0 \*)  
(\* tiempo1(T1) \*)

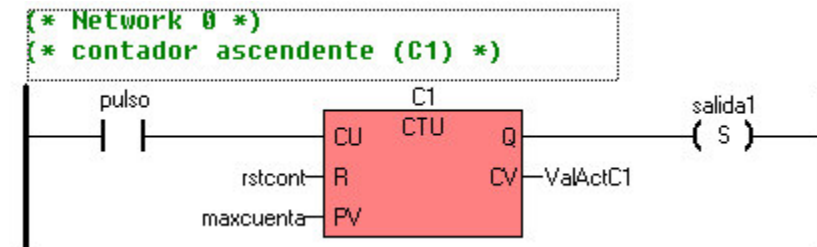


MAIN * VAR_GLOBAL				
	Symbol	Address	Data Type	Comment
1	marcha	%I0.0	BOOL	
2	motor	%Q0.0	BOOL	
3	tiempo	%VW0	INT	
4	ValActT1	%VW2	INT	
5				
6				

Con la siguiente tabla, tendremos el tipo de variables aceptadas en cada una de las entradas o salidas del bloque.

Operands	Input/Output	Data Type	Acceptable Memory Areas
<i>Tx</i>	-	Timer instance	T
<i>IN</i>	Input	BOOL	Power flow
<i>PT</i>	Input	INT	I, AI, AQ, M, V, L, SM, constant
<i>Q</i>	Output	BOOL	Power flow
<i>ET</i>	Output	INT	Q, M, V, L, SM, AQ

## Contador:



Symbol	Address	Data Type	Comment
1 pulso	%I0.0	BOOL	
2 rstcont	%I0.1	BOOL	
3 salida1	%Q0.0	BOOL	
4 maxcuenta	%VW0	INT	
5 ValActC1	%VW2	INT	
6			

Con la siguiente tabla, tendremos el tipo de variables aceptadas en cada una de las entradas o salidas del bloque.

Operands	Input/Output	Data Type	Acceptable Memory Areas
<i>Cx</i>	-	Counter instance	C
<i>CU</i>	Input	BOOL	Power flow
<i>R</i>	Input	BOOL	I, Q, M, V, L, SM, T, C
<i>CD</i>	Input	BOOL	Power flow
<i>LD</i>	Input	BOOL	I, Q, M, V, L, SM, T, C
<i>PV</i>	Input	INT	I, AI, AQ, M, V, L, SM, constant
<i>Q</i>	Output	BOOL	Power flow
<i>CV</i>	Output	INT	Q, M, V, L, SM, AQ

Los seteos de tiempos o conteo, tambien podrian ser valores constantes.

## **Carga de un programa al PLC KDN**

Ejecutaremos el EasyPog. Dentro de **(File, Open Project)** seleccionaremos el proyecto a ejecutar.

Una vez cargado el archivo en el editor, conectaremos el PLC con el cable de programación.

En **(Tools, Communications)** realizaremos un Search para detectar el equipo, una vez encontrado el PLC, daremos Ok.

Ahora si podemos realizar la carga al equipo, **(PLC, Download)**.

Nos preguntara que para realizar la carga, el PLC interrumpira la ejecucion del programa en curso, diremos que si, y en el cuadro de texto inferior nos anunciara si la carga fue exitosa.

### **Material adjunto en [www.CAIPE.com](http://www.CAIPE.com)**

- GUIA RAPIDA DE INICIO (castellano)-
- MANUAL INTRODUCTORIO (castellano)-
- HARDWARE MANUAL FOR KDN K3(ingles)-
- SOFTWARE AND INSTRUCCION MANUAL FOR KDN K3(ingles)-
- PID MANUAL FOR KDN K3(ingles)-