ESCUELA SUPERIOR POLITECNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACION

"DISEÑO Y CONSTRUCCIÓN DE UN BRAZO MECÁNICO MEDIANTE EL MICROPROCESADOR CPU-214 DE LA SIEMENS ALEMANIA, PARA SER UTILIZADO EN LA MECANIZACIÓN DE PROCESOS EN UNA FABRICA DE PLÁSTICOS EN LA CIUDAD DE GUAYAQUIL"

TESIS DE GRADO

Previa a la obtención del Título de:

INGENIERO EN ELECTRICIDAD

Especialización ELECTRONICA

Presentada por:

FRANCISCO JAVIER BUSQUET TRES

GUAYAQUIL - ECUADOR

1999

AGRADECIMIENTO

Ing. Hugo Villavicencio

director de tesis.

Ing. Miguel Yapur por su

ayuda.

DEDICATORIA

A Dios

A mi Esposa Talitha

A mi hija Clarita

A mis Padres.

TRIBUNAL DE GRADUACION

ING. ARMANDO ALTAMIRANO
SUB DECANO DE LA FACULTAD DE ING.
ELECTRICA Y COMPUTACION

ING. HUGO VILLAVICENCIO
DIRECTOR DE TESIS

ING. MIGUEL YAFUR
VOCAL

ING. NORMAN CHOOTONG
VOCAL

DECLARACIÓN EXPRESA

" La responsabilidad por los hechos, ideas y doctrinas expuestos en esta tesis, me corresponden exclusivamente; y el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL"

(Reglamento de Exámenes y Títulos profesionales de la ESPOL)

FRANCISCO J. BUSQUET TRES

RESUMEN

Primero doy la idea ¿qué es un Autómata? Para comprender la lectura posterior de lo que será el controlador de mi brazo mecánico. Para lograr eso, voy describiendo sus partes internas fisicas y lógicas, al igual que sus partes externas.

Segundo explico la manera en que este procesador maneja los periféricos de entrada y salida de acuerdo al programa que alberga en su memoria , la diferencia marcada que hay entre su funcionamiento lógico con una computadora. Explicamos además la manera en que es mantenido el programa, como cargarlo, como descargarlo, como modificarlo en memoria.

Como tercera parte explicamos la forma, lenguaje y consideraciones al programar además de poner ejemplos después de casi todas las instrucciones a fin de que esté bien claro. En el anexo pondremos los comandos más comunes. Luego daremos una tabla de los principales comandos de la familia S7 de la serie SIEMENS.

Por último presentaremos el proyecto aun que es sumamente limitado en un margen muy estrecho del poderío de esta herramienta (autómata) dentro del automatismo, sin embargo es suficiente para dar una idea de las múltiples formas que puede ser programado el autómata. Para ello diseñé el brazo cuyo proceso de fabricación está descrito, luego los pasos de su programación, diagramas de flujo, declaración de variables, referencias cruzadas y programa.

INDICE GENERAL

ABREVIATURAS

Abreviación	Descripción
A, Q	Salidas
AB	Byte de salida
AI	Cantidad de entradas analógicas que deben de leerse
AKKU1,2	Acumulador del CPU. Al cargar AKKU1, se desplaza su contenido al 2
ANZ 0 / 1	Indicación de resultado 0/1
AW	Palabra de salida
AWL	Lista de instrucciones STEP5
BF	Constante de Byte
BS	Zona de datos del sistema
D	Dato (1 bit)
DB	Módulo de datos
DL	Palabra de datos (byte izquierdo)
DR	Palabra de datos (byte derecho)
DW	Palabra de datos
E, I	Entradas
EB	Byte de entrada
EF	Buzón de recepción
EW	Palabra de entrada
FB	Módulo funcional

FUP	Esquema de funciones
I, E	Entradas
IN	Alarma de flanco negativo
INP	Alarma de flancos negativo y positivo
IP	Alarma de flanco positivo
IPN	Alarma con flanco positivo y negativo
KB	Constante 1 byte
KBE	Parámetro del DB1. (emisión)
KBS	Parámetro del DB1. (recepción)
KC	Constante (2 caracteres)
KF	Constante de coma fija
KH	Constante hexadecimal
KM	Configuración binaria 2 bytes
KOP	Esquema de contactos
KT	Constante (temporizador)
KY	Constante 2 bytes
KZ	Constante (valor contador)
M	Marca
MB	Byte de Marca
MW	Palabra de marca
NT	Cantidad de temporizadores procesados
OB	Módulos de organización para aplicaciones especiales
OBA	Identificador de bloque en DB1 para entradas analógicas integradas
OBC	Identificador de bloque en DB1 para contadores integrados
OBI	Identificador de bloque en DB1 para alarma integrada
OHE	Liberar contador horas operación
OHS	Ajustar contador horas operación

OP	Aparato de operación
OV	Indicador de desbordamiento
PAA	Imagen de proceso de salidas
PAE	Imagen de proceso de entradas
PB	Módulo de programa
PB o PY	Byte de periferia
PG	Aparato de programación
PW	Palabra de periferia
Q, A	Salidas
SAV	Salvar hora tras la última transición de STOP a RUM
SB	Módulo de paso
SET	Ajustar hora, fecha
SF	Situación del buzón de emisión
SM	Marcas especiales de memoria
STP	Actualizar hora en STOP
STW	Situación de la palabra de estado (reloj – calendario integrado)
T	Temporizadores
TFB	Indicador de bloque en DB1 para módulo funcional de temporizador.
TIS	Parámetro del DB1 para ajustar hora de alarma
VKE	Resultado de la combinación
Z	Contadores

INDICE DE GRÁFICOS

FIGURA 23 FORMA DE LA MANO 153

FIGURA 24 FORMA DE LOS DEDOS 154

FIGURA 25 MUESTRA LA POSICIÓN DE LA PINZA CUANDO ESTÁ CERRADA Y ABIERTA 154

FIGURA 26 MECANISMO PARA ACCIONAR EL PISTÓN DE LA MANO 155

FIGURA 27 PLACA DEL CIRCUITO 157

FIGURA 29 DIAGRAMA ESQUEMÁTICO DEL CIRCUITO (IMAGEN EXPORTADA DEL ORCAD) 158

INTRODUCCIÓN

Este proyecto presentará una herramienta muy usada hoy en día en el área de la automatización de procesos industriales en los que el hombre está realizando procesos repetitivos y a su vez cíclicos. Para evitar ese proceso monótono se ha lanzado esta herramienta de gran utilidad llamada PLC, que son también conocidos como autómatas programables. Estos tienen como base un microprocesador, con otros componentes que serán a lo largo de este trabajo descritos y usados, que pueden ser programados para esa función cíclica.

El conocimiento de esta herramienta sustituye en muchos casos circuitos digitales complejos. Todo está en la programación y uso de los recursos disponibles en él.

Para esto, **primero** presentamos al PLC describiendo sus partes tangibles, formas de alimentación, capacidad de almacenamiento, riesgos, secuencia de trabajo, sistema de almacenamiento, módulos con que cuenta, la entrada y salida. **Segundo** lo que es la programación, consideraciones al programar, características del programa, su lenguaje de programación y las instrucciones debidamente ilustradas con ejemplos. **Tercero** presentamos lo que es el proyecto, la construcción del brazo que será posteriormente programado para esa tarea repetitiva.

CAPITULO I

1. ¿QUE ES UN AUTOMATA?

1.1. EL AUTÓMATA PROGRAMABLE

La estructura electrónica del autómata programable está basada sobre un microprocesador, lo cual permite la capacidad de ser programable y de comunicarse, mediante módulos de entradas y salidas, con el mundo exterior.

La estructura da una flexibilidad muy útil en el mundo industrial. Los autómatas actualmente son modulares, fenómeno que facilita más aun su manejo; además de adecuar mejor los presupuestos a la necesidad del automatismo.

1.2. ESTRUCTURA DEL AUTÓMATA:

Basándonos en la figura[1] que presentamos a continuación explicaremos de qué módulos se compone dicha configuración:

1.2.1 La fuente de poder, que alimenta entradas salidas, y CPU.

1.2.2 Unidad Central de Procesamiento.

1.2.3 Módulos periféricos.

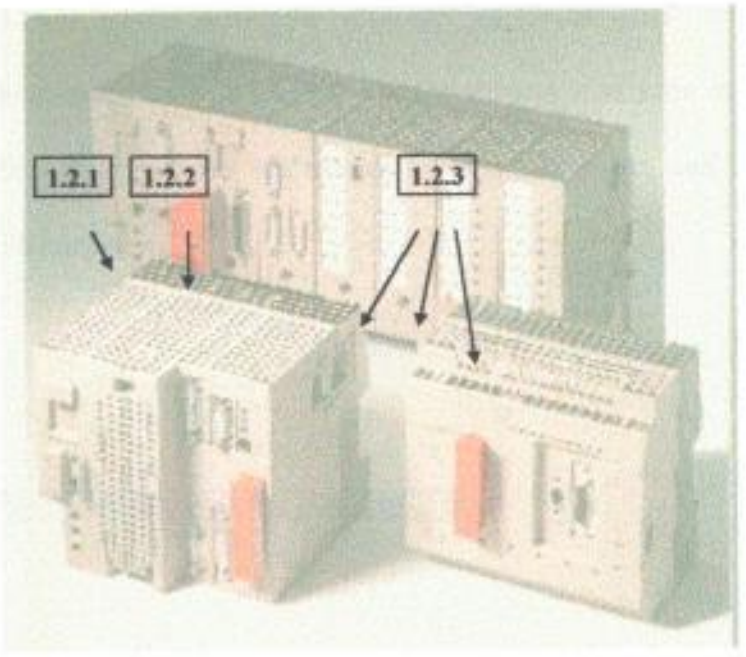


FIGURA 1 *ESTRUCTURA DE UN AUTÓMATA*

1.2.1. LA FUENTE DE ALIMENTACIÓN:

Opera directamente con la red 110/220 mediante una pequeña fuente de poder, o se conecta directamente una alimentación de 24 Vcc para alimentar al CPU. En caso como el autómata de la familia S7-200 que lleva el microprocesador 214, no cuenta con una fuente de poder, razón por la cual puede ser reemplazada por una fuente regulada de 24V, a 1Amp. Esta alimentación de un amperio es suficiente como para alimentar al CPU y a los periféricos de entrada y salida si es que no cuenta con otros módulos digitales

(1) La figura presenta la estructura externa básica del autómata.

ni analógicos adicionales que ya la carga sea grande, al igual que en caso contar con los autómatas cuyas salidas son transistores a colector abierto, no se deberá de cargar a todas las salidas al mismo instante con carga ni inductiva, ni resistiva.

1.2.2. UNIDAD CENTRAL DE PROCESAMIENTO:

Ejecuta el programa de mando. Cuando falla la alimentación, una batería interna alimenta la memoria RAM a fin de que el programa permanezca en ella durante el corte energético. El programa puede correr de dos formas diferentes, conectando el autómata directamente con un computador el cual le va dando la secuencia de las instrucciones del programa cargándolo en la RAM, o de lo contrario una vez programado puede grabarse en una memoria EPROM, (en algunos autómatas se usa una EEPROM) interna al autómata e independizar de esta forma del computador. En ese momento el autómata, una vez grabado el programa en la memoria EPROM tiene el potencial de ejecutarse o quedar almacenado allí y cada vez que se enciende el autómata, puede cargarse a la RAM el contenido de la EPROM para ser ejecutado en forma cíclica el programa. Es de tener claro que el principio de

funcionamiento es cíclico, chequea las entradas y condiciona las salidas de acuerdo a las instrucciones del programa.

Los estados de señal de los módulos de entrada se almacenan en una zona reservada de la memoria RAM denominada imagen de proceso de las entradas. (PAE).

Las informaciones que la CPU quiere transmitir a los módulos de salida se almacena en una zona reservada de la memoria RAM denominada imagen de proceso de las salidas (PAA).

Para poderse comunicar el autómata con el ordenador, se usa un puerto serie en el que se conecta el aparato de programación, este puede ser o un computador o un teclado propio de los autómatas, que dependiendo de la marca varía de uno a otro. Es de señalar que cada marca de autómatas trae su propio lenguaje de programación.

1.2.3. MÓDULOS PERIFÉRICOS:

Permiten el intercambio de la información entre la CPU y la periferia del proceso (emisores de señal, actuadores y convertidores), también

dependiendo de la marca y del modelo que se cuente hay mayor cantidad de salidas y entradas que facilitan o dificultan la programación, hay desde 4 entradas y salidas hasta los actuales que permiten la incorporación de entradas y salidas adicionales a las 32 que traen.

- **EL MÓDULO DIGITAL DE ENTRADA Y SALIDA.-** Son adecuados para tareas sencillas de mando en las cuales solo aparecen los estados de señales 0 y 1 que son conocidos como niveles lógicos de verdadero o falso, los cuales se determinan en función de un rango de voltajes que determina entre que voltajes puede considerarse como verdadero y entre cuales falso. Los límites superior e inferior normalmente están establecidos como los topes Vcc y cero, luego en margen depende de un fabricante a otro el rango que quiera dejar, que viene a ser como grados de precisión o libertad que deje.

- **MÓDULO ANALÓGICO DE ENTRADA Y SALIDA.-** Permiten detectar y generar magnitudes variables tales como tensiones y corrientes.

- **MÓDULOS TEMPORIZADORES.-** Permite ajustar temporizadores sin modificar el programa. En algunas marcas el autómata cuenta con un reloj interno a tiempo real, lo que permite que el programador pueda darle utilidad a esos temporizadores ya sea para aplicaciones internas o externas al autómata.

- **MÓDULOS CONTADORES:** Permiten procesar impulsos de frecuencias normalmente hasta Hz, sirve esta aplicación para el control de motores paso a paso, los cuales al emitir un tren de pulsos posicionan al motor en cualquiera de sus posiciones dependiendo el número de pasos que cuente.

- **MÓDULO CONTADOR RÁPIDO / LECTURA DE RECORRIDO:** El contador rápido puede usarse para captar impulsos de alta frecuencia y para tareas de posicionamiento.

- **MÓDULOS COMPARADORES:** Permite vigilar si se sobrepasa el límite ajustado, con ellos podemos establecer límites a ciertos parámetros que queremos controlar que se ciñan en cierto margen o que queden en un

valor fijo. Pueden hacerse control de si una variable es mayor, menor o igual que un valor prefijado. Este puede ser un parámetro ya sea de corriente o de voltaje.

- **MÓDULO DE DIAGNOSIS:** Permite el control del funcionamiento del bus periférico.

- **INTERFACES PARA MEMORIA:** Permite listar mensajes con hora y fecha a través de periféricos, como la impresora, en algunos casos puede establecerse mensajes hablados dependiendo de la flexibilidad en el manejo de esta interface.

Existen otros elementos que componen el autómata, tal como: el bus con **bloques de conexión** por pinzas o tornillos (unen el CPU con los módulos periféricos. En cada elemento del bus es posible enchufar dos módulos periféricos), **Interfaces** (permite configurar el autómata en varias filas), **Carril normalizado** (Sobre él se monta el autómata).

1.3. FUNCIONAMIENTO DEL AUTÓMATA:

La CPU tiene integrados temporizadores y contadores, que pueden cargarse, borrarse, arrancarse y pararse desde el programa.

Los valores de tiempo (temporización) y de cómputo (valor de un contador) se guardan en zonas reservadas de la memoria RAM. Otra zona de esta misma memoria permite almacenar resultados intermedios. Estas posiciones de memoria se denominan marcas.

El sistema operativo incluye programas del sistema que fijan la ejecución del programa de usuario, la gestión de entradas y salidas, la división de la memoria, la gestión de datos, y similares.

La unidad aritmética lógica se compone de dos acumuladores, AKKU 1 y 2, que procesan las operaciones por bytes y por palabras. También dispone de un AKKU de bits para procesar operaciones binarias.

La Unidad de control gobierna y coordina todo el autómata. Siguiendo el programa, llama sucesivas veces las instrucciones contenidas en la memoria de programa, y las ejecuta. Para ello procesa las informaciones contenidas en

el PAE y se consideran los valores de los temporizadores y contadores internos, así como los estados de la señal de las marcas internas. Los resultados se registran en el PAA.

Los programas pueden almacenarse en estos módulos EPROM o EEPROM. Los programas contenidos en estos módulos pueden copiarse en la memoria del programa.

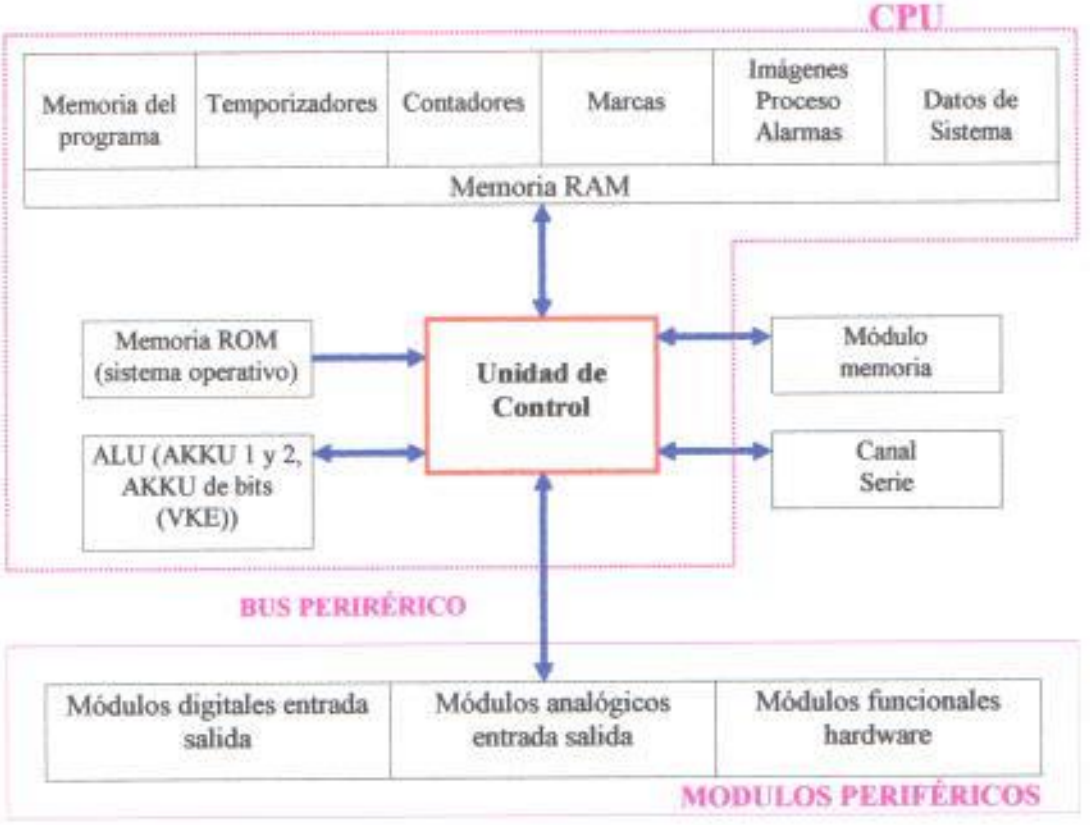
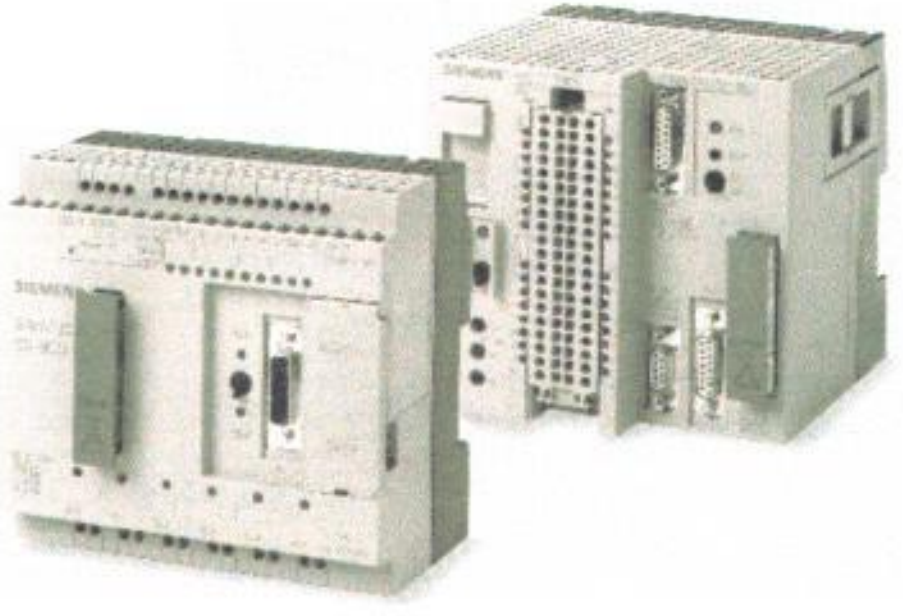


FIGURA 2 ***CONFIGURACIÓN INTERNA DE UN AUTÓMATA***

AUTOMATA estudio hecho sobre la marca SIEMENS



CAPITULO II

2. DIRECCIONAMIENTO DE LOS AUTOMATAS PROGRAMABLES:

2.1. AUTÓMATA PROGRAMABLE S5 100U- 95U -90 U

[2]El autómata S5-100 U es un sistema de mando programable de la familia SIMATIC S5. Ha sido desarrollado expresamente para realizar tareas de automatización a nivel de baja potencia. No es un **aparato compacto**, sino

[2] Centro de Formación del Departamento de Automatización de SIEMENS Curso de nivel 1 de equipos SIMATIC S5

que está compuesto de diferentes módulos, que se pueden agrupar en función de la tarea de automatización a resolver.

Unidad Central: contiene el procesador central (CPU) y bornes para la alimentación de 24Vcc.

Elementos del Bus: Los elementos del bus se enganchan a la derecha de la CPU sobre el riel normalizado. En cada elemento del bus se pueden enchufar dos módulos periféricos. También dichos elementos sirven para conectar los cables que transportan las señales que van y vienen de la periferia.

Módulos periféricos: Elementos que sirven para lectura y mando de señales de campo.

Para el autómata programable S5-100U se dispone de los siguientes tipos de módulos periféricos:

1. **Módulos de entradas y salidas digitales:** Estos módulos los utiliza cuando quiera resolver tareas de mando sencillas. Reconocen (módulos de entrada) o proporcionan (módulos de salida) 2 niveles de tensión: alto y bajo.

2. **Módulos de entrada y salida analógicos:** Los módulos analógicos se utilizan cuando se presentan magnitudes que varían continuamente, como por ejemplo: intensidad, tensión, temperatura o resistencia.

3. **Módulo simulador (Simulator):** Para generar las señales de entrada o indicar señales de salida, (digitales) sin necesidad de tener instalados elementos de campo.

4. **Fuente de alimentación:** Para alimentar el AG S5-100 U a través de una red alterna de 115 o 230V. Proporcionará la tensión de 24V de c.c. necesaria para alimentar al CPU.

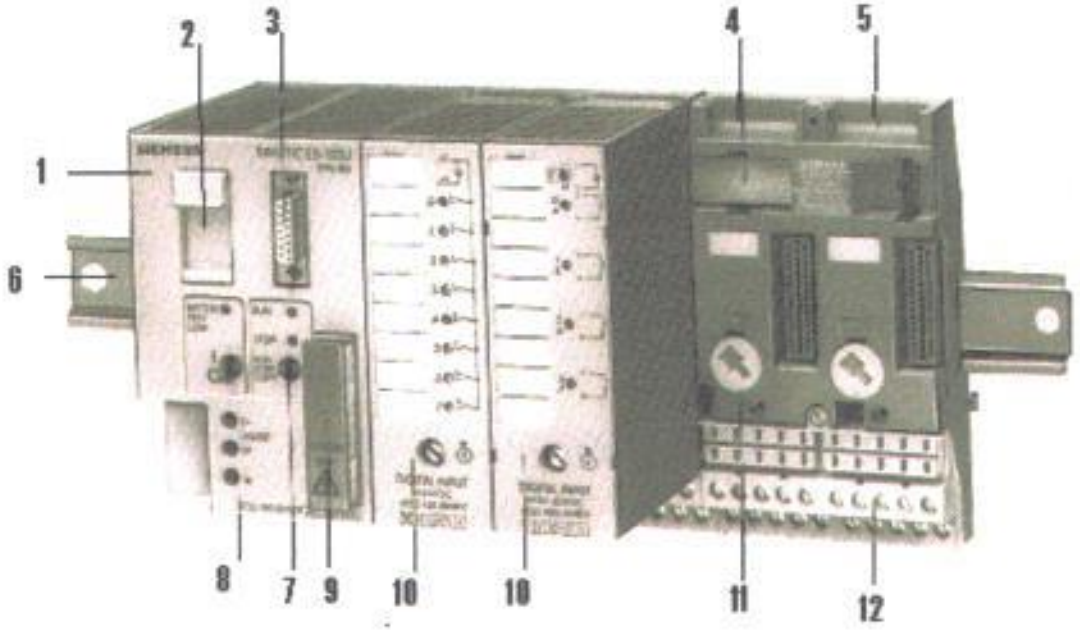
5. **Módulos temporizadores (Timer) :** Para el ajuste de tiempos con un destornillador (sin tener que modificar el programa).

6. **Módulo de contadores (Counters):** Para el ajuste manual de contadores (sin tener que modificar el programa).

7. **Módulos de comparadores (Comparator):** Para controlar si se rebasan o no se detectan intensidades / tensiones.

8. **Interfases:** Para configurar el AG S5-100 U en varias filas.

9. **Aparatos de programación:** Se pueden utilizar todos los aparatos de programación, es decir: PG 605, PG 615, PG 635, PG 685, PG 710, PG 730, PG 750 o PG 770.



Descripción[3]:

1. Unidad central CPU 100/102/103 siempre con fuente de alimentación integrada DC 24/9 V para módulos periféricos.
2. Batería tampón.
3. Conexión para unidad de programación.
4. Cable plano (bus interno).
5. Elemento de bus.

[3] Imagen que muestra la mayoría de los módulos que componen el autómata.

6. Espacio para fuente de alimentación (AC 115/230V, DC 24V).
7. Panel de operación con indicación de estado operativo y conmutador de operación.
8. Conexión para DC 24V.
9. Cartucho de memoria EPROM.
10. Módulo periférico
11. Elemento codificador giratorio.
12. Conexión de tornillo o aleatoriamente tipo pinza (no representado)

2.2. ESTRUCTURA DEL AUTÓMATA PROGRAMABLE S5-100

2.2.1. ORGANIZACIÓN DE LA SECUENCIA DE TRABAJO:

Se debe interpretar en un programa la tarea de mando de su máquina e introducirlo con su aparato de programación en el autómata programable. Este programa se compone de una secuencia de instrucciones que almacenan luego en la Memoria de programa interna de la unidad central (CPU). Además de esta Memoria de programa su autómata programable contiene también la Unidad de control y la Periferia, que son necesarias para que pueda cumplir su función.

Los elementos que integran la Unidad Central (CPU):

- Unidad de control,
- memoria,
- imagen de proceso,
- temporizadores internos,
- contadores internos, y
- marcas (memorias intermedias).

2.2.2. MEMORIA DE PROGRAMA:

En la memoria del programa se deberá cargar el programa, luego de lo cual debe funcionar el autómata programable S5-100U. Esto puede hacerlo de dos maneras diferentes:

Si introduce el programa con ayuda de un aparato de programación, ó, introduce un programa que se encuentra grabado en el módulo de memoria EEPROM ó EPROM, que solo tiene que enchufar en la CPU de su autómata programable.

La memoria donde se encuentra el programa de su autómata programable es una memoria de las denominadas RAM.

Una memoria RAM tiene las siguientes características:

- Su contenido puede variarse rápidamente (Por ejemplo: durante una puesta en marcha).
- Si falla la alimentación, la memoria pierde el programa en ella contenida si no está insertada la batería tampón.

Si se quiere garantizar totalmente el que no pueda perderse su programa, deberá almacenarlo entonces su programa en un módulo de memoria (EPROM ó EEPROM).

Un módulo de memoria tiene las siguientes características:

- Se puede modificar el contenido de la memoria, esto es, el programa.
- Si utiliza un módulo de memoria EPROM y quiere modificar su contenido, es necesario borrarlo previamente con luz ultravioleta (UV).

  Para volver a grabarlo es necesario un aparato de programación (no el PG 605).

Si se usa un módulo de memoria EEPROM, su contenido puede modificarse simplemente escribir encima. La programación de la modificación puede realizarse directamente en la unidad central (función COPY) ó con un aparato de programación.

2.2.3. LA UNIDAD DE CONTROL:

La unidad de control es el organizador del autómata programable. Esta se encarga de que se ejecuten adecuadamente las diferentes instrucciones de un programa. La unidad de control:

- Organiza la entrada de los datos que vienen de la periferia, situándolos o copiándolos en la imagen de proceso de las entradas (PAE),
- combina estos datos,
- efectúa cálculos,
- considera los valores de los temporizadores y contadores internos,
- considera los estados de señal memorizados en las marcas,
- deposita en la denominada **imagen de proceso de salidas** (PAA) durante el procedimiento del programa los resultados de las combinaciones, y
- organiza la salida de los resultados de la periferia.

2.2.4. PERIFERIA:

La periferia está integrada por:

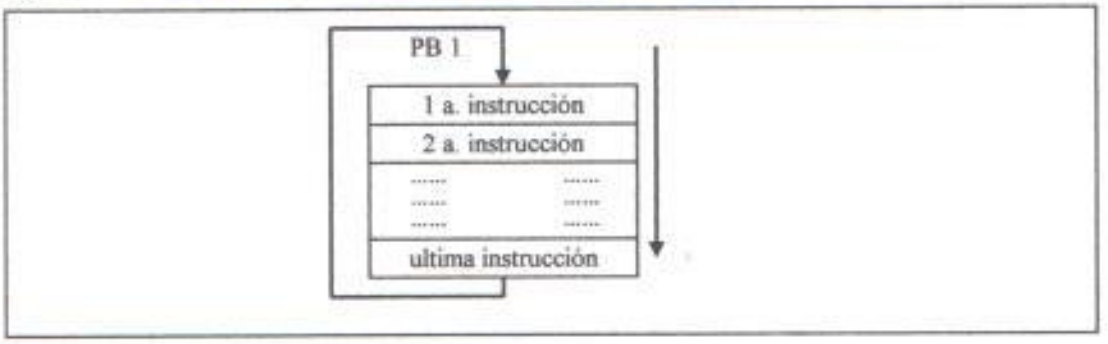
- Los módulos de entradas y de salidas digitales,
- los módulos de entradas y salidas analógicas,
- los módulos de temporizadores, contadores y comparadores.

El denominado **bus periférico** encausa el intercambio de datos entre la unidad central y la periferia. Este bus periférico se forma conectando los módulos de bus.

2.2.5. PROCESAMIENTO CÍCLICO DEL PROGRAMA:

El autómata programable S5-100U funciona cíclicamente, esto es, una vez finalizado un recorrido completo del programa, comienza a procesar nuevamente su primera instrucción.

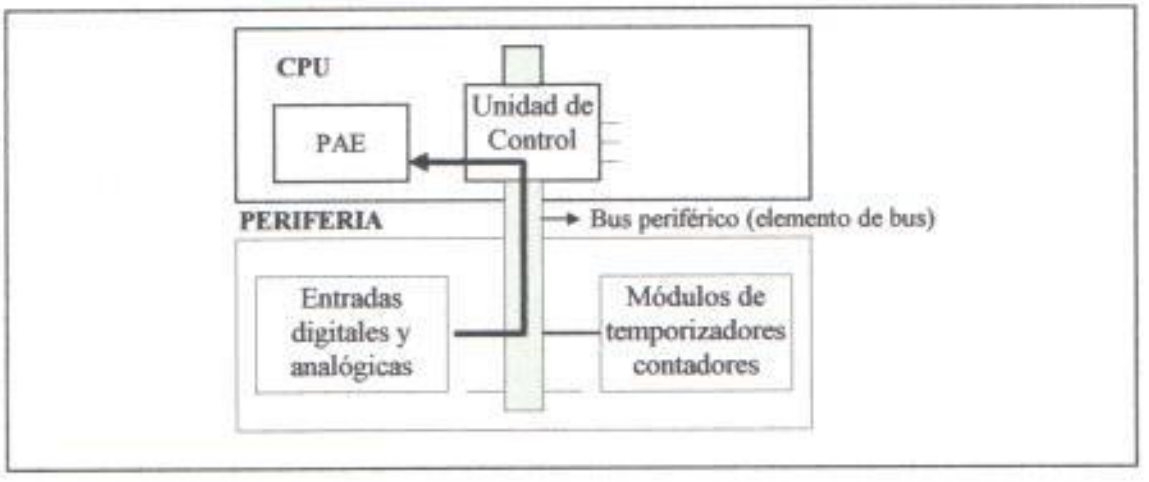
Figura 4 ***PROCESAMIENTO CÍCLICO DEL PROGRAMA***



El procesamiento cíclico del programa en el autómata programable discurre de la siguiente forma:

- Al comenzar cada ciclo, la unidad de control consulta los estados de señal de todos los módulos de entrada (digitales y analógicas), y forma la imagen de proceso de las entradas PAE. La PAE, pues, es una copia de las señales de entrada.

Figura 5 ***LA UNIDAD DE CONTROL RELLENA LA IMAGEN DE PROCESO DE LAS ENTRADAS (PAE)***



Ahora comienza el procesamiento del programa. La unidad de control toma de la memoria interna el programa (instrucción por instrucción), y las ejecuta. Para ello establece combinaciones, efectúa cálculos con datos de PAE y considera los estados de los temporizadores y de contadores, así como de las marcas.

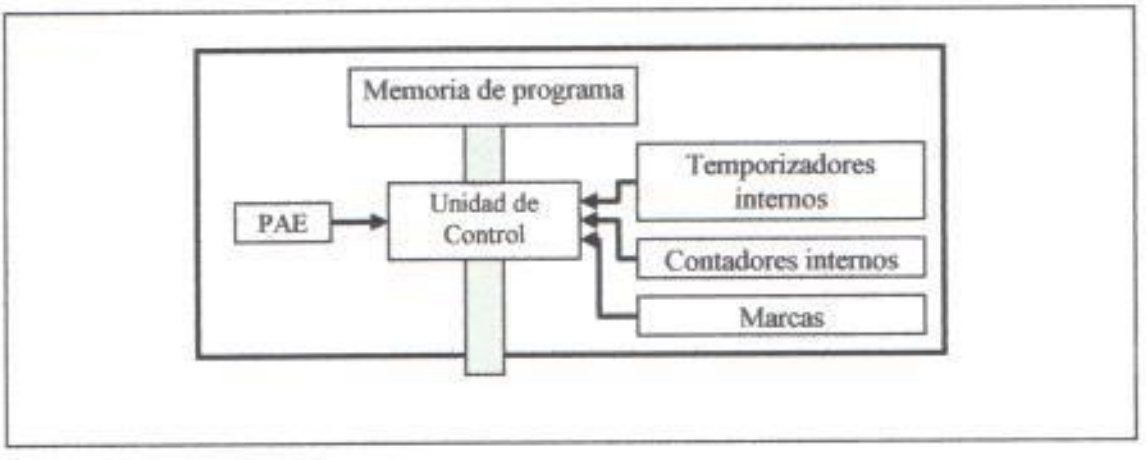
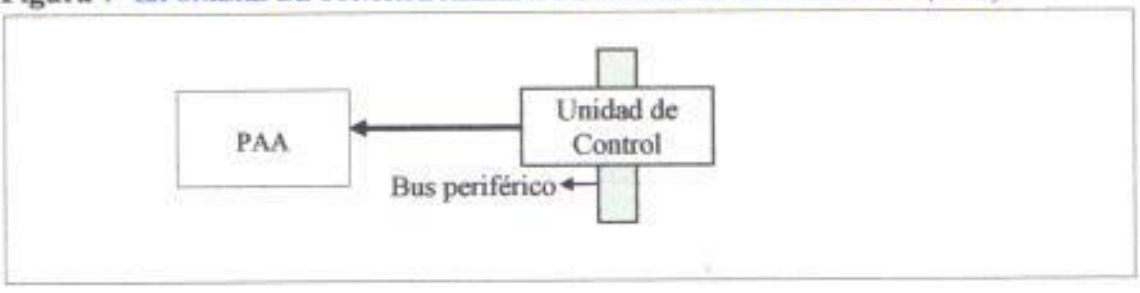


Figura 6 ***LA UNIDAD DE CONTROL EJERCIENDO LA FUNCIÓN DE ORGANIZADOR***

La unidad de control deposita el resultado del procesamiento del programa en la imagen de proceso de las salidas "PAA".

Figura 7 *LA UNIDAD DE CONTROL RELLENA DE LA IMAGEN DE LAS SALIDAS (PAA)*



La unidad de control solo transfiere a los módulos de salida, temporizadores, y contadores los estados de señal contenidos en la imagen de proceso de las salidas (PAA) cuando ha finalizado el recorrido del programa, esto es, al final de un ciclo. Ahora puede comenzar un nuevo ciclo.

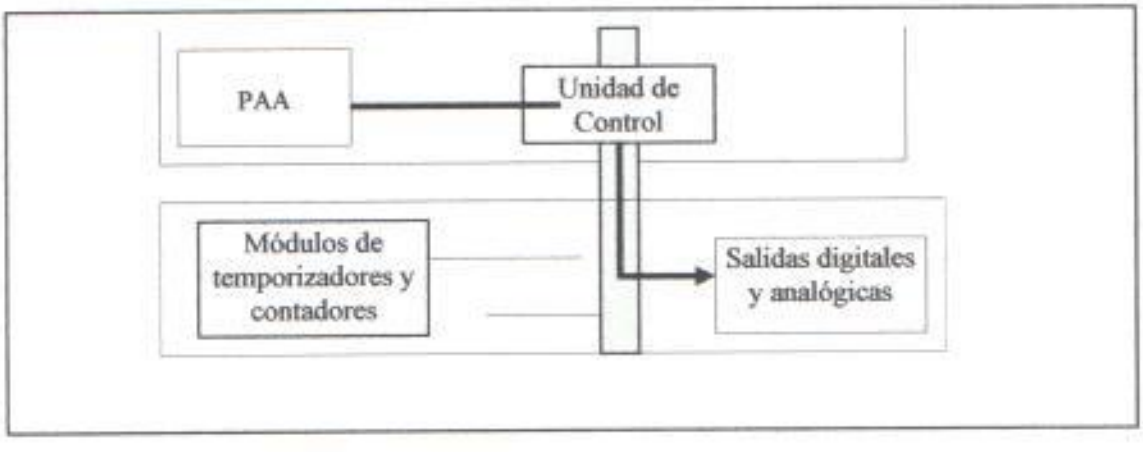


Figura 8 La unidad de control transfiere la PAA a los módulos periféricos.

2.3. DIRECCIONAMIENTO DE SEÑALES S51-100U

2.3.1. FIJACIONES DE LAS DIRECCIONES:

Cada dirección se compone de:

- Una abreviatura que define el tipo de módulo enchufado,
- el número del puesto de enchufe, y
- el número del canal.

El número del puesto de enchufe y el número del canal están separados por un punto.

2.3.2. TIPO DE MÓDULO:

Si ha enchufado un módulo de entrada, la dirección comenzará con la "**E**"; si ha enchufado un módulo de salida, la dirección comenzará con una "**A**".

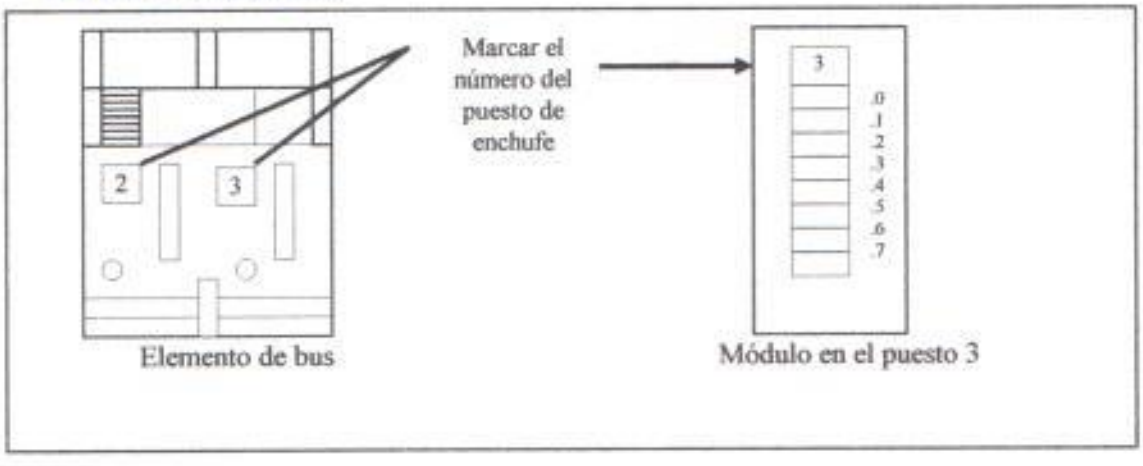
2.3.3. NÚMERO DEL PUESTO DE ENCHUFE:

Cada puesto de enchufe en un elemento del bus tiene asignado un número determinado. Se comienza a contar por 0 justo a la derecha de la unidad central y se prosigue desde allí hasta un máximo de 31. Esto significa que: puede utilizar hasta 16 elementos del bus y enchufar hasta 32 módulos. Para identificar el elemento del bus y el módulo, marque su número en la etiqueta para tal efecto.

2.3.4. NÚMERO DEL CANAL:

Si se ha enchufado un módulo digital de cuatro canales, dispone de los números del canal 0 a 3. Si se ha enchufado un módulo digital de ocho canales, podrá utilizar los números del canal 0 y 7.

Figura 9 *MARCADO DE LOS NÚMEROS DEL PUESTO DE ENCHUFE EN EL MÓDULO Y EN ELEMENTO DE BUS.*



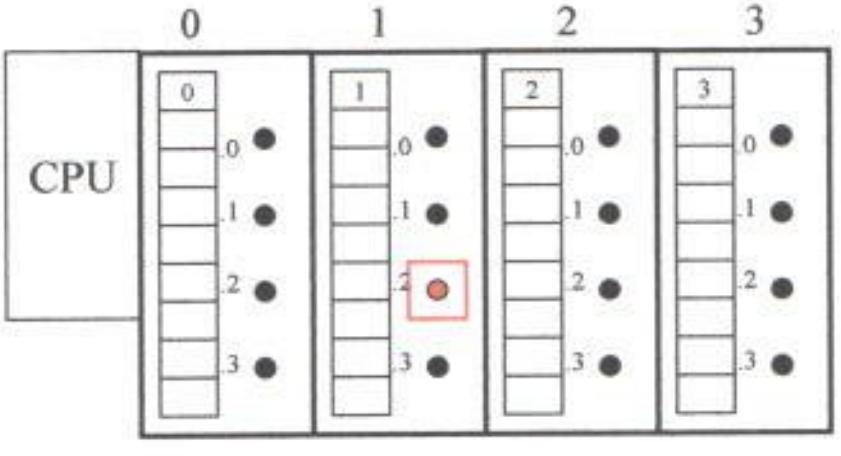
Algunas formas de direccionar al autómata es por ejemplo:

U E 1.2

donde la letra U es un operador que simboliza la operación lógica de "y", la letra E le indica que es una entrada la que se está direccionando. El número uno indica que se trata del puesto 1 y el 2 es que se trata del canal 2.

Es importante dar a entender que el autómata a pesar que se coloquen direcciones que no hayan sido asignadas aun, es decir, que fisicamente no existen, el PLS las censará como si fuera que existen, pero, les colocará el estado de cero. Lo veremos en el gráfico.

Figura 10 *UBICACIÓN DE LA SALIDA*



Si fueran de varias filas, siendo que puede tolerar según el autómata una o más filas, en el caso de S5-100u puede tolerar hasta 4 filas y una cantidad de 15 módulos por fila. Hay formas de ampliar aun más la capacidad pero si fuera ese el caso hay que agregarle nuevos elementos que se complementan al autómata para vincularse con este, con el fin de administrar los periféricos que ya no alcanza el PLC. Estos elementos pueden ser elementos tales como **las interfases** que son, pues, elementos capaces de prolongar el bus.

Si se amplía el PLC, dentro de una fila, hay que tener en cuenta en el momento de querer direccionar alguna entrada o salida en la programación, ya que en algunos casos no se mantiene las direcciones. Los enchufes se enumeran correlativamente, empezando por el ubicado en forma más próxima al CPU, al que se le asigna el número 0 (cero). En caso de agregar un bloque de 4 salidas a uno ya existente, éste último toma las últimas direcciones que le corresponden a ese módulo, razón por la cual, es que hay que agregarlo al final de un módulo, sería un error conectarlo del lado de la dirección menor.

No está demás que después de cada ampliación, se compruebe las direcciones si corresponden o no a lo que se espera.

A la hora de programar debemos de tener en cuenta la dirección ya sea de entrada asignada por la letra E o de salida direccionada por la letra A. Una vez que comencemos a explicar la programación de un PLC nos daremos cuenta de la importancia de estos conceptos a fin de aclarar en que dirección exacta queremos accesar, no solo de que tipo (entrada, salida) sino como decíamos anteriormente el número de enchufe, módulo, etc. Tanto las entradas como las salidas pueden ser manipuladas en un programa a fin de que el autómata haga lo que le indiquemos.

A la hora de direccionar, no solo podemos direccionar entradas y salidas de manera simple, poniéndole la dirección del módulo y el número de bit, sino que además podemos decirle al PLC que la información que le entra es de tipo byte o palabra, lo que equivale a decir que solo haría falta decirle cual es la dirección de inicio y el sabría cuanto debe de direccionar. Además también podemos direccionar contadores y marcas. Estas marcas se pueden considerar como si fueran unos relees internos al PLC que pueden manipularse internamente a fin de extraviar trabajo y costo en el armado de un sistema de relees externos para una actividad en la que se requiera gran cantidad de ellos.

El PLC trabaja por un sistema de imagen en el que puede testear el estado de las entradas y las salidas, eso lo hace como en el esquema que exhibimos a continuación:

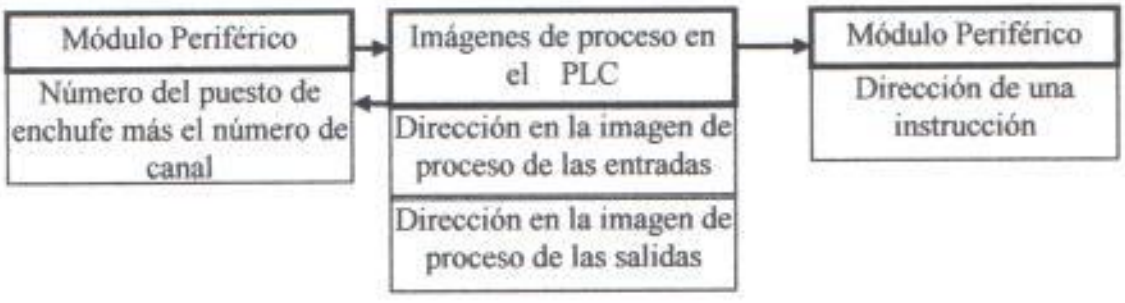


Figura 11 *TESTEO DE DIRECCIONAMIENTO*

2.3.5. MÓDULOS ANALÓGICOS:

Cuando contamos con módulos analógicos, hay que tener en cuenta que pueden enchufarse en puestos desde el 0 hasta el 7. Lo que implica que tenemos un rango de 65536 informaciones distintas esto es 16 bits = 2 Bytes = 1 palabra. Estos módulos permiten varias formas de direccionarse byte a byte o palabra a palabra usando operaciones de carga y transferencia. Es de recordar, que un canal digital, solo permite comunicar por canal, la información de 0 ó 1, se precisa solo un bit, ya sea de entrada o de salida. En el momento en que instalamos un módulo analógico, el PLC ya reconoce el

módulo sabiendo que tiene que direccionar más espacio de memoria. Es decir: por cada puesto se reservan 8 bytes, que es equivalente a cuatro palabras; se conmuta la zona de direcciones del puesto de enchufe y el margen de direcciones abarca del 64 (puesto 0, canal 0) hasta el byte 127 (puesto 7, canal 3). Gráficamente podemos verlo mejor:

Nro de enchufe	0	1	2	3	4	5	6	7	Nro de canal
AG	64+65	72.....	80.....	88	96	..104	...112	...120	_0
	66+67								_1
	68+69								_2
	70+71	79	87	...95	...103	...111	...119	...127	_3

Como un ejemplo un poco práctico como para poder entender mejor la matriz presentada arriba, si pidiéramos el byte 90+91, este se hallaría en el número de enchufe 3 y número de canal 1.

Para direccionar módulos analógicos se debe de tener en cuenta los siguientes aspectos:

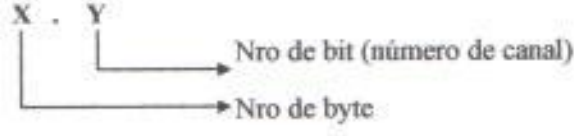
- Los módulos de entrada solo pueden funcionar con uno, dos o cuatro canales. Mientras que los de salida solo pueden funcionar con dos canales.
- El rango que permite el módulo analógico va desde 64 a 127.

- Estos módulos pueden enchufarse en cualquier posición comprendida entre el 0 y 7 como dirección de enchufe. Por cada enchufe se reservan 8 bytes, y por cada canal se necesitan 2 bytes.

2.3.6. MÓDULOS DIGITALES:

En estos módulos, la información que se puede transmitir es solo un nivel lógico de 1 o 0, que para ello es usado un nivel de voltaje de 24 voltios como límite superior de verdadero, que implica que la entrada está en 1 (conectada), y un nivel inferior de 0 (desconectado).

En el PLC pueden enchufarse en cualquier posición los módulos digitales que van enumerados desde el 0 hasta el 31. Cada uno de los canales de este tipo de módulo se representa mediante un bit. Esta numeración se realiza en la forma que describimos a continuación:



La dirección de byte; **X** es el puesto en que está ubicado o enchufado el módulo, respecto al CPU del PLC. El número de canal **Y** (dirección de bit)

resulta de tomar en cuenta donde se ha conectado los actuadores o los emisores en el bloque de bornes. Normalmente al comprar los módulos en estos vienen indicados el número de módulo y los números de bits que corresponde a cada entrada de datos.

2.3.7. MÓDULOS COMBINADOS DE ENTRADA Y SALIDA

Este tipo de módulos posibilita escribir datos en el módulo desde el programa de mando y leer desde el módulo, datos en el programa de mando. Las direcciones en el proceso de imagen son las mismas ya sea que se esté hablando de salida o entrada. Los datos transmitidos tienen generalmente diferentes significados.

Módulos de salida con diagnosis de perturbaciones: Los módulos de salida pueden señalizar perturbaciones del PLC, además de que este error se manifiesta en forma óptica mediante la exhibición de un led rojo. Estas señales de error al estar el programa corriendo en el PLC son testeadas en su estado, estas son los canales de entrada E x.0 y E x.1, en caso de existir este error la señal se pone en estado alto, es decir, 1 lógico.

Las perturbaciones que pueden ser chequeadas son aquellas como:

- Corto circuito en un canal de salida / fusible quemado o falta de alimentación para la carga que se indica con E x.0.

- Módulo averiado, que es cuando el transistor de salida está quemado, esto se manifiesta en el E x.1 (donde la "x" en cada caso significa la dirección de byte del módulo de salida).

  En los módulos de salida que no cuentan con la presencia de diagnosis de perturbaciones la PAE se pone a 0.

Módulo de entrada y salida digital 16E/16^{A} DC 24V: Este tipo de módulo solo se pueden enchufar en los puestos desde el cero, hasta el siete (0 al 7). Este módulo ocupa la misma zona en términos de dirección que un módulo analógico. Sin embargo solo se usan los dos primeros bytes de los reservados.

La dirección se compone de la dirección de byte n o n+1 y del número de canal **"Y"**. La dirección inicial **"n"** es la dirección inicial de un proceso de enchufe, o sea, del primer byte reservado. Los números **"n"** están marcados en la cara frontal de cada módulo indicando en que lugar se encuentran a partir de la posición 0 que es la contigua al PLC. Mientras que el número de canal es aquel de donde estén conectados los emisores y actuadores en el

conector con terminales tipo pinza. Es de recordar que los datos de entrada y salida también ocupan las mismas direcciones.

Por ejemplo: Si decimos la dirección 121.5, indica que el módulo está conectado en el puesto 7, y el número de canal es el 5. Poniendo esto en forma más gráfica tenemos:

Puesto de enchufe		0	1	2	3	4	5	6	7
PAE	Canal N0..n7	64.0... ...64.7	72.0... ...72.7	80.0... ...80.7	88.0... ...88.7	96.0... ...96.7	104.0... 104.7	112.0... ..112.7	120.0... ...120.7
PAA	Canal n+1.0.... n+1.7	65.0... ...65.7	73.0... ...73.7	81.0... ...81.7	89.0... ...89.7	97.0... ...97.7	105.0... 105.7	113.0... ..113.7	121.0... ...121.7

Módulos funcionales hardware: Los módulos funcionales tienen su direccionamiento propio. Algunos se direccionan como módulos ya sean digitales o analógicos.

2.3.8. ESTRUCTURA DE LAS IMÁGENES DE PROCESO:

La imagen de proceso es la forma en que las entradas PAE depositan la información dentro del PLC. Como datos técnicos podemos decir que:

- PAE y PAA ocupan cada una 128 bytes en la memoria RAM.
- PAE y PAA tienen la misma estructura y pueden dividirse en tres zonas que las representamos en forma gráfica:

Dirección de byte en PAE y PAA AG S5 - 95U	Areas de la periferia	Nro puesto de enchufe
0....31	Módulos s5 - 100U	0...31
32...55	Periferia integrada	
56...63	Zona de dirección no ocupada	
64..127	Módulos s5-100U	0...7

- La zona de direcciones comprendida entre los bits 0 y 31 está reservada para informaciones que van o vienen de módulos que se direccionen como los de tipo digital.
- La zona de direcciones no ocupada reservado para informaciones desde o hacia la periferia integrada depende del tipo de PLC.
- La zona de direcciones no ocupada sirve para almacenar resultados intermedios.
- La zona de direcciones comprendida entre los 64 y 127 está reservada para informaciones que van o vienen de módulos que se direccionan como los de tipo analógico.

Direc. de byte relativas	Area de Periferia	PAE hexad.	PAA hexad.
0 – 31	Entradas y salidas digitales de la periferia externa	6300-631F	6380-639F
32 - 33	Entradas y salidas digitales de la periferia integrada	6320-6321	63A0-63A1
34	Entradas de alarma de la periferia integrada	6322	63A2
35	Byte de diagnosis	6323	-
36 - 37	Contador integrado A	6324 -6325	-
38 - 39	Contador integrado B	6326-6327	-
40 - 55	Entradas y salidas analógicas de la periferia integrada	6328-6337	63A8-63A9
56 - 63	Zona de dirección no ocupada	6338-633F	63AA-63BF
64 – 127	Entradas y salidas analógicas de la periferia externa	6340-637F	63C0-63FF

CAPITULO III

3. INTRODUCCIÓN A LA PROGRAMACIÓN:

Antes de comenzar a describir parte de la programación que necesitaremos de ahora en adelante, describiremos brevemente ciertas partes de PLC.

3.1. EL INTERRUPTOR DE CONECTADO/DESCONECTADO:

Sirve para alimentar o no al autómata de corriente. En estado de desconectado el PLC no recibe corriente, por lo tanto al no estar alimentado no puede

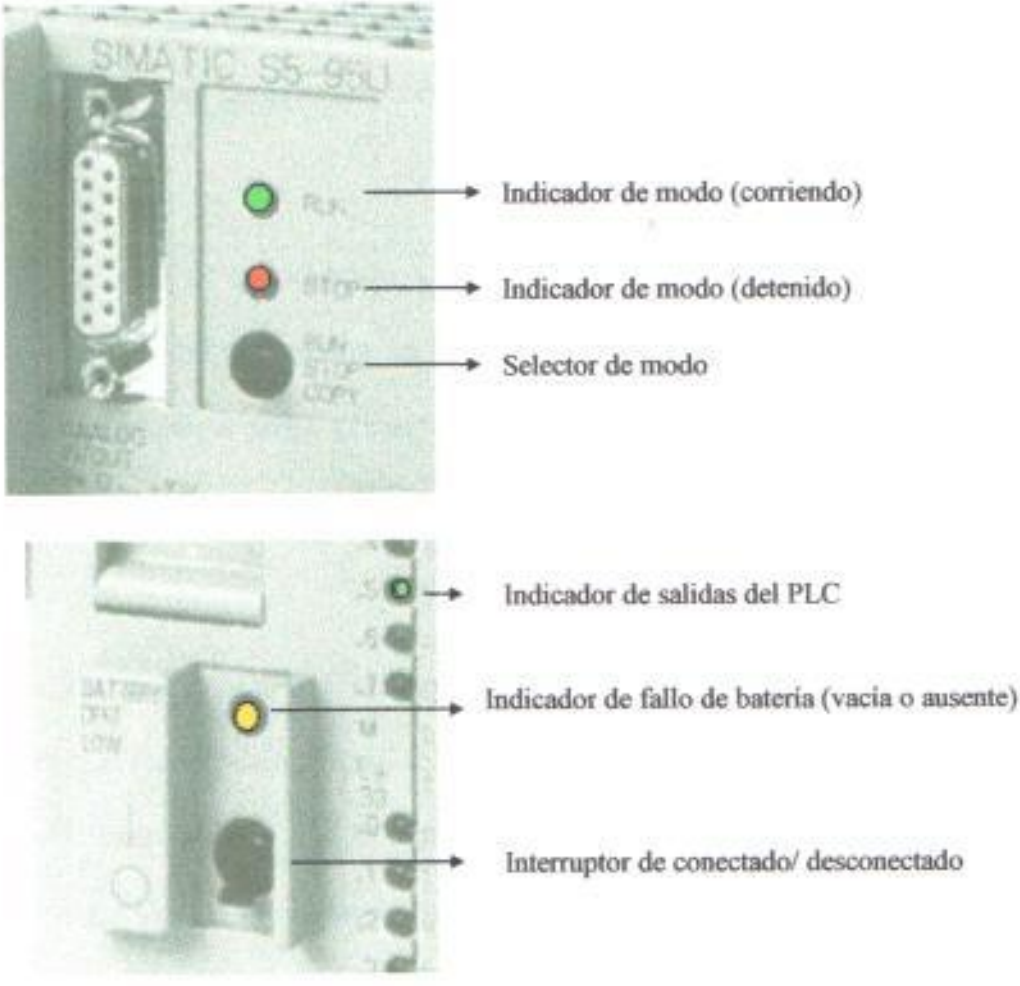
realizar ningún tipo de manipuleo de información. Esta alimentación le viene suministrada del regulador de tensión.

Al arrancar el PLC el sistema operativo procesa en el DB1 y acepta los parámetros. Además se ejecuta el software de arranque con su respectivo interprete de comandos.

3.2. SELECTOR DE MODO:

Permite seleccionar el modo en el que quiero trabajar, si pongo RUN significa que quiero hacer correr el programa. Este corre en forma secuencial realizando en forma cíclica cada una de las instrucciones, los temporizadores comienzan su secuencia de conteo, también se cargan los estados de señal de las entradas integradas y de los módulos de salida.

Figura 12 *INDICADORES DE MODO*



Si selecciono el modo STOP, estoy deteniendo el corrimiento del programa o lo que es lo mismo dejo al microprocesador solo con corriente, eso implica que podrá recibir datos, pero, no los estará identificando ni tampoco manipulando hasta que no se pase a un estado de RUN, los valores de los temporizadores, contadores, marcas e imágenes de proceso presentes al entrar

en el estado STOP. Este modo puede cambiar si se llegara a accionar el selector de modo. También por invención de un aparato de programación, si el AG está en RUN y por perturbaciones que lleven al AG al modo STOP.

Para pasar de un programa a otro, es necesario que borremos el programa previo, ya que la memoria del PLC queda cargado con el programa anterior, o si llegáramos a cargar el nuevo programa, igual quedarán en la memoria los datos antiguos.

Un programa en el PLC comienza llamando al primer módulo OB1, razón por la cual independientemente del nombre del programa, el primer módulo tiene el mismo nombre que el del programa anterior, eso nos dará si estamos trabajando con el ordenador, un mensaje de confirmación de sobre escritura, en caso de aceptar esta sobre escritura, el programa nuevo entrará en el PLC borrando el programa anterior, o lo que sucede en la mayoría de los casos en que se trabaja con el computador es que el PLC edita el programa residente y pregunta que si se desea modificar o no. Para evitar problemas posteriores o de confirmación de sobre escritura es mejor borrar los módulos residentes en la memoria del PLC. Al borrar eso se borra lo siguiente:

- La memoria de programa del PLC.
- Todos los datos (marcas, temporizadores y contadores)
- Todos los identificadores de error.

Para lograr un borrado previo a la programación del PLC se puede hacer una limpieza de la memoria mediante los siguientes pasos:

- Colocar en STOP el selector de modo.
- Extraer la batería del PLC.
- Poner el interruptor principal de energía en desconectado (posición 0).
- Poner el interruptor principal de energía en conectado (posición 1).
- Colocar la batería.

Veamos que ha pasado en este caso internamente en el PLC, al no estar energizando éste mantiene el programa residente en memoria RAM gracias a la alimentación que le proporciona la batería auxiliar. Al desconectar la batería, la memoria RAM no recibe los impulsos de refrescamiento que ella necesita para mantener la información interna, así que termina perdiendo la información. Una vez que conectamos de nuevo el equipo, (interruptor en posición uno), el PLC comienza a dar a la memoria nuevamente los pulsos que

ella necesita para mantener la información, pero esta última ya está vacía. Si colocamos de nuevo la batería es para que se mantenga la información dentro del PLC una vez que la energía es interrumpida.

3.3. PARA CARGAR EL PROGRAMA:

En el caso de algunos de estos PLC se permite la carga de los diferentes tipos de memoria que expresaremos más adelante en los anexos.

Cuando existe una memoria ROM, es decir, el cartucho, el programa que reside en él se almacena inmediatamente en la memoria de la memoria RAM en el área de programas.

Solo se cargarán aquellos módulos que sean válidos para la ejecución. Se puede modificar un módulo, pero de igual forma podemos borrar o sobre escribir módulos completos. No obstante, para esto no se borran los módulos en la memoria del programa, sino simplemente se los deja sin utilidad, se los invalida. Estos espacios de memoria no pueden ser más escritos nuevamente. Este hecho en muchos casos provoca un desbordamiento de memoria, razón por la que comienza a parpadear el led rojo de stop, en caso de contar con el programa para comunicar el ordenador con el PLC, saldrá el problema en

forma explícita en la pantalla del monitor. En algunos casos expone que no hay espacio en memoria, y en otros simplemente avisa un desbordamiento. Para poder remediar este defecto es necesario comprimir la memoria del PLC, esto se hace mediante el programa de interfaces de la computadora al PLC. Hay una segunda forma de cargar y esta es cargar el programa en la memoria del cartucho, ya sea EPROM o EEPROM y se la inserta, en ese momento podemos sacar la pila del PLC a fin de extraer el programa residente en la memoria del programa, o de contar con la computadora, nos vamos al menú de módulos, borrar y le especificamos que son los módulos del PLC, de esa forma nos aseguramos de que el programa residente en la memoria no interfiera en ella corrida del nuevo programa, ya que las variables que hayan quedado activas en la corrida del programa anterior a menos que se desactiven por software estas permanecerán encendidas. Una vez realizado estos pasos de limpieza de la zona de la memoria destinada a los programas, procedemos a la inserción del cartucho y posteriormente, encendemos el PLC y se cargará en la memoria de programa el residente en la EPROM.

3.4. SALVAGUARDA DEL PROGRAMA:

Un programa solo puede ser almacenado en forma independiente de si está o no la alimentación del PLC si se encuentra presente la bateria tampón. Durante la operación de salvado, el programa es depositado en un cartucho de memoria . Como indicáramos anteriormente solo se copian los módulos válidos tal como el DB1 que contiene valores prefijados integrados, tan pronto como hayan sido cambiados algunos de sus valores.

Para lograr la copia del programa en la EPROM, necesitamos poner el selector de modo en COPY, durante 3 minutos aproximadamente hasta que parpadee la luz roja, en ese momento podemos soltarlo y lo tendremos ya grabado en la memoria. En caso de encenderse la luz roja en forma intermitente, quiere decir que se ha cometido un error, ya sea: que no está presente el cartucho de memoria, ó, no hay programa residiendo en memoria RAM.

La batería en caso de encontrarse en buen estado puede mantener en la memoria durante el periodo de 1 año.

En caso de encontrarse con problemas en la alimentación del PLC, el contenido de la memoria solo se mantiene (remanencia) si hay insertada una batería tampón. Cuando se vuelve a conectar, vuelven a estar disponibles los siguientes datos:

1. El programa de mando y los módulos de datos.
2. Los estados de las marcas y contadores remanentes.
3. El contenido de la pila de interrupción.

3.5. VISUALIZACIÓN DEL ESTADO DE SEÑAL DEPENDIENTE DEL PROGRAMA STATUS.

En el programa contamos con diferentes menús que nos pueden estar mostrando el estado de las variables.

Estas formas pueden variar dependiendo de la forma en que programemos ya que hay 3 diferentes. Una de ella es por medio de instrucciones, las otras dos son por construcción gráfica de un circuito. Existe en el menú, una opción, en la que podemos ver el paso de la corriente en caso de tener en modo gráfico, en caso de tener en modo de instrucciones muestra el estado de VKE y de los AKKU1, AKKU2, además de otros parámetros de suma importancia a fin de ver el desarrollo del programa. Es de recalcar en esta ocasión la necesidad de que cuando se programa en instrucciones, hay que tener muy en cuenta el

estado del VKE ya que la mayoría de instrucciones de salto, al igual que de temporización lo necesitan en un nivel lógico verdadero. De no ser así, de tenerlo en cuenta en múltiples ocasiones el programa se saltará instrucciones que aparentemente no debería, pero, si encuentra en estado bajo al VKE no puede usarla, además de ello no todas las instrucciones a parte de necesitar un estado alto, la dejan en ese mismo estado, pues hay algunas que a pesar de ser dependientes, la modifican su nivel lógico. (Esto lo explicaremos con mayor detalle más adelante. Además de encontrarse en un anexo A una lista de las instrucciones e indica si dependen o no del VKE).

El programa además cuenta con un submenú en el que podemos decir el estado de cuales variables nos importan y nos indica en cada caso el estado que tienen durante el desarrollo del programa.

3.6. FORZADO DE VARIABLES "STEUERN VAR"

Esta es una función que permite modificar la imagen de proceso de los operandos binarios y digitales. Es posible modificar las siguientes variables E, A, M, T, Z y D.

Este comando forzado no corre en sincronismo con la corrida del programa. Una vez que el programa corre, el PLC pasa al estado de RUN, es este estado se ejecuta usando las variables de proceso modificada, eso no implica que durante las siguientes ejecuciones no puedan volver a modificarlas. Recordamos que el programa una vez que llega al final a menos que se le indique lo contrario regresa al principio, lee las entradas nuevas y condiciona las salidas de acuerdo a esas nuevas entradas.

Las variables como E, A y M se modifican en la imagen del proceso ya sea bit a bit, byte a byte o de palabra en palabra. Las otras variables como T y Z con formato KM y KH proceden de acuerdo a forzado de marcas que les sirve para trabajar con los flancos y modificar su estado.

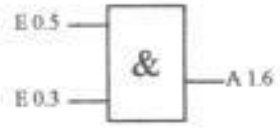
El estado visible de las variables puede ser interrumpido en el caso que en la entrada tengamos un formato u operando erróneo. Un error de formato es, por ejemplo, ponerle como base de tiempo de un contador un 5 en adelante, ya que no existe esa base, o sobrepasarse el límite máximo que puede tolerar.

3.7. BÚSQUEDA

Esta es una función de facilidad que se encuentra el programador en programas de gran extensión, ya que mediante ella podemos accesar directamente a una marca, instrucción, dirección u operandos del listado del programa, ella salta en cada una de las coincidentes. Estas instrucciones son válidas ya sea en la función de entrada, salida o status. Las marcas solo es posible en estados funcionales.

3.8. FORMAS DE REPRESENTACIÓN DEL LENGUAJE STEP5

Podemos poner un esquema que ayudará un poco a dejar clara la idea de las tres formas de representar el lenguaje. Cualquiera de estas formas es válida para programar al PLC.

En contactos (KOP)	En funciones (FUP)	Lista de instrucciones (AWL)
		U E 0.5 U E 0.3 = A 1.6 BE

Así como existen las diferentes formas de representar la misma cosa, los programas también pueden realizarse de diferentes maneras, eso es, existe una forma de programar que es la lineal en la cual el programa contiene las instrucciones una detrás de otra, y las va leyendo en forma secuencial. La otra forma de hacerlo es una programación estructurada, eso es: contamos con un módulo principal, y de este derivan otros módulos que son llamados dentro del principal. Cuando se trata de programas de gran extensión es conveniente alcanzar la costumbre de poner los problemas en partes divididas e irlo atacando en pequeños sub-problemas, eso ayuda que posteriormente a la programación podrá dársele un mantenimiento adecuado y será más fácil su interpretación.

- Las operaciones básicas que comprenden funciones ejecutables en módulos de organización, de programa, de paso y funcionales. Con excepción de la suma y resta entre números en coma flotante y las operaciones organizativas, pueden ser representada en cualquiera de las formas, es decir, AWL, FUP y KOP.
- Las operaciones complementarias comprenden funciones complejas como funciones de prueba de bit, de desplazamiento, sustitución y transformación. Solo pueden ser representarse por medio de listas

de instrucciones amado AWL, no puede representarse en forma de esquema eléctrico.

- Las operaciones de sistema acceden directamente al sistema operativo, estas son funciones que requieren conocimientos para poder manipularlas, ya que se trabaja directamente con la configuración del PLC. Tanto la entrada como la salida de operaciones de sistema es solo posible en la forma de representación AWL.

3.9. OPERACIONES BÁSICAS:

3.9.1. OPERACIONES COMBINACIONALES

Operación	Operando		Significado
O			**Combinación O** se realiza la combinación O con consulta del operando respecto al estado de señal "1".
ON			**Combinación O negado:** se realiza la combinación O con consulta del operando al estado de la señal "0".
O(			**Combinación O de expresiones entre paréntesis:** se realiza la operación O de lo precedente con la expresión que se encuentra entre los paréntesis.
U			**Combinación Y :** consulta del operando respecto al estado de la señal "1".
UN			**Combinación Y negado:** consulta del operando respecto al estado de la señal "0".
U(			**Combinación Y paréntesis:** consulta del operando respecto al estado de la señal "1" de la señal entre paréntesis

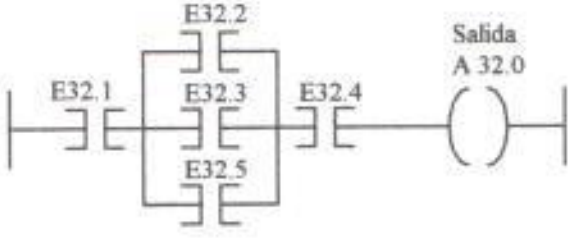
En cada una de estas operaciones combinacionales es de suma importancia tener en cuenta el estado del VKE. La forma de realizar las diferentes operaciones es entre el VKE de la operación con el precedente. El VKE es el resultado de las combinaciones. Pueden usarse con estas instrucciones diferentes operandos tales como: E (entrada), A (salida), M (marcas de memoria), T (temporizadores), Z (contadores).

Ponemos como ejemplo lo siguiente:

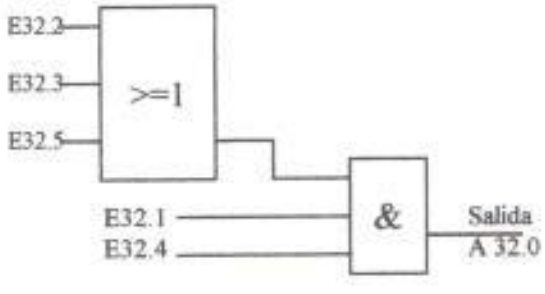
Supongamos que tenemos que introducir una clave en una puerta de 5 interruptores de dos estados en el que nos dicen que la combinación puede ser:

teclas: 1, 2, 4;ó, también puede ser la combinación 1, 3, 4; ó 1, 5, 4. Podemos representar el circuito que obedezca esta secuencia en las tres formas de programar.

1.) **KOP**



2.) **FUP**



3.) **AWL**

```
U E32.1
U(
O  E32.2
O  E32.3
O  E32.5
)
U  E32.4
=  A32.0   Salida
BE
```

Recordamos que el formato de entrada es E, el número que sigue es el lugar que ocupa a partir del CPU donde está enchufado y finalmente después del punto viene el bit que ocupa dentro de ese módulo.

3.9.2. OPERACIONES DE MEMORIA

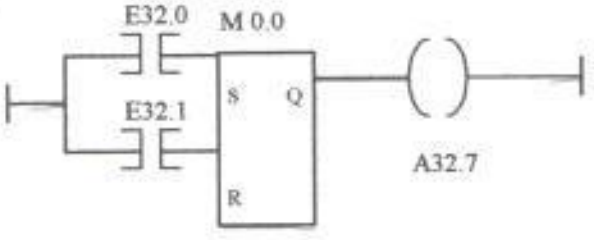
Estas dos instrucciones son de suma importancia durante una programación, ya que son dos funciones mediante las cuales podemos activar una entrada, una salida o una marca. No necesita de operandos anteriores que la activen más si es tener en cuenta el estado del VKE, pues en caso de que no esté en uno, no se ejecuta esta instrucción lo que nos obliga al poner cualquiera de estas instrucciones que el VKE se encuentre en estado de señal alto (uno). Es de tener en cuenta además que tiene en consideración al VKE lo inhibe.

La función set pone en 1 al operando que la acompaña, y en caso de que se la apague esta permanecerá encendida. Mientras que la función reset, hace la operación contraria, es decir, que apaga el operando que lo acompaña, poniéndole en estado 0.

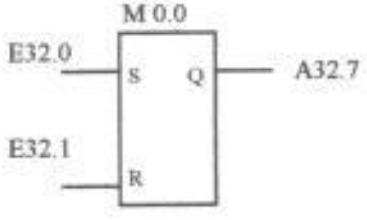
Operación	Operando		Significado
S	E,A,M	□	**Activar (poner a 1).** Durante la primera ejecución del programa el VKE = 1 se asigna el estado de señal 1 al operando afectado. Las modificaciones del VKE no varían ya este estado.
R	E,A,M	□	**Borrar (poner a cero).** Durante la ejecución del programa con VKE = 1 se asigna el estado de señal "0" al operando afectado.
=	E,A,M	□	**Asignar.** En cada ejecución del programa se asigna el VKE actual al operando afectado.
		↑	Parámetro: 0.0 ... 127.7 0.0 ... 127.7 0.0 ... 255.7

Como ejemplo de estos comandos podemos ver el siguiente programa:

1.) KOP



2.) FUP



3.) AWL

```
U   E   32.0
S   M    0.0
U   E   32.1
R   M    0.0
U   M    0.0
=   A   32.7
```

3.9.3. OPERACIONES DE CARGA Y TRANSFERENCIA.

Este tipo de operaciones permite asignar ciertos valores, ya sean constantes o variables de entrada. Permiten intercambiar información entre las diferentes zonas de operandos, también carga a temporizadores y contadores con valores constantes para su posterior uso. Todas estas cargas se hacen manipulando los registros AKKU1 y AKKU2. Estos son registros especializados del PLC que hacen la función de memoria intermedia. En la mayoría de los PLCs estos registros tienen una longitud de 16 bits. Tienen dos niveles a saber alto y bajo, cada uno de 8 bits que conforman entre ambos niveles un byte. El AKKU2 no puede ser accesado directamente por el programador más que por AKKU1, ya que el AKK1 antes de admitir una nueva carga transfiere su valor al AKKU2. Siempre en AKKU2 estará almacenado el último valor que tuviera el AKK1.

Los operandos de este tipo pueden cargar y transferir ya sea byte a byte o palabra a palabra. En caso de que la transmisión sea de 8 bits se realiza por la derecha, es decir, los 8 primeros bits de los registros comenzando a contar por la derecha, que es lo mismo que decir que corresponde al byte bajo. Los restantes bytes se ponen a cero.

El contenido de los dos registros pueden ser empleados para ejecutarse diferentes operaciones.

Tanto la función de carga como la de transferencia se pueden efectuar con independencia de las indicaciones, además, tampoco la ejecución de estos dos tipos de operandos afecta a las indicaciones.

Si se va a utilizar estos operandos es preferible que sea en lista de intruccioines (AWL) ya que el modo gráfico solo alcanza a realizar estas operaciones con tiempo y contaje, las demás operaciones de carga y transferencia no la admiten.

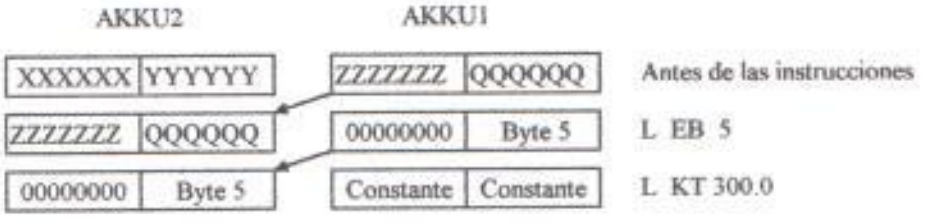
Cargar: La información que está residente en memoria, al efectuar una operación de carga como puede ser de los periféricos de entrada PAE, se copia en el AKKU 1, mientras que el contenido previo que yacía en AKKU 1 pasa al AKKU2.

Por ejemplo:

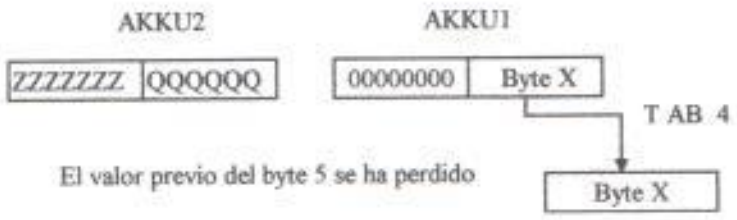
Supongamos que vamos a realizar dos cargas una de un periférico y otra de una constante, lo que nos indica que las instrucciones serían:

- L EB 5 (es una carga del byte número 5, supongamos que tiene 10111001).
- L KT 300.0 (es una carga de un temporizador con 3 seg.).

Poniendo en forma gráfica los contenidos de los AKKUs queda:



Transferir: Al transferir ocurre el proceso inverso, es decir, que de los contenidos de los AKKUs se copia en la memoria. Pero a diferencia del caso anterior, en esta instrucción no se modifica el contenido del AKKU 1. Al transferirse a la zona de salida digital, se actualiza automáticamente en el módulos de entrada el byte o palabra correspondiente a la transferencia.



Operación	Operando		Significado
L	□	□	**Cargar** Los operandos se copian en el AKKU 1 con independencia del VKE. Tampoco influencia en el VKE
T	□	□	**Transferir** El contenido de AKKU 1 se asignan a un operando con independencia del VKE. Tampoco influencia en el VKE.
Identificador			**Parámetros de PLC S5 - 95U**
EB			0…127
EW			0…126
AB			0…127
AW			0…126
MB			0…255
DR			0…254
DL			0…255
DW			0…255
T*			0…255
Z*			0…127
PB/PY**			0…127
PW*			0…127
KM*			0…126
KH*			0…FFFF
KF*			-32768…+32767
KY*			0…255 por cada byte.
KB*			0…255 por cada byte
KC*			2 caracteres alfanuméricos
KT*			0.0…999.3
KZ*			0…999.3
LC	□	□	**Cargar codificadamente:** En los AKKU 1 se cargan, codificados en BCD, temporizadores o ajustes de contadores binarios, con independencia del VKE.
Identificador: T y Z			**Parámetro:** 0…127

*no con transferir
** depende del PLC

Haciendo un ejemplo de estas instrucciones, tenemos los casos típicos que son:

Ejemplo 1: (**Carga de un Temporizador**)

La mayoría de operaciones con tiempo, llaman a los temporizadores internos del PLC, para hacer uso de ellos. Hay algunos PLC, que puede ser activado un temporizador, que lleva el calendario, es decir, que pueden usarse con programación del día y de la hora, además, cuentan con un factor de corrección que puede ser introducido en caso de variaciones ya sea de adelanto o atraso del reloj por influencia de la temperatura, aun que mínima, normalmente se garantiza en los 20 ° C de temperatura el buen funcionamiento del reloj. Este factor lo suma el PLC cada vez que ve que se ha cumplido el tiempo de base de corrección que le hemos puesto. Ejemplo: en caso de que se adelante, 1 seg. por cada hora , el factor de corrección será de 1 seg. y la base de tiempo será de una hora.

Cuando invocamos a un temporizador, tampoco este será exacto en la medición del tiempo, aun que el error puede ser disminuido cuando introducimos una mayor cantidad de bucles para que se cumpla. Como el PLC realiza en forma cíclica el programa, cuando ve una marca de

invocación de tiempo, la activa si no lo estuviera, y cuando vuelve a pasar por esa llamada de tiempo, consulta el estado del tiempo, él no sabe si le falta mucho o poco por terminar el conteo, sino que cuando cumple el tiempo, y pasa el programa por la marca y consulta le avisa que ya se cumplió el tiempo que esperaba. De allí que existe el error dependiendo de cuanto demoró en caer en la marca el programa desde que se cumplió el tiempo.

Al introducir mayor cantidad de ciclos para un mismo tiempo, el error será minimizado, ya que si ponemos 300 centésimas de seg. el programa deberá de pasar 300 veces por la marca, de tal suerte que puede equivocarse al detectar la marca a los 299 ó 301 centésimas, en cambio si le ponemos 3 seg. solo serán 3 pasadas del programa pudiendo detectar la marca a los 2 ó 4 seg. Como vemos en este caso el error perpetuado en los cálculos será dependiente de la mayor o menor cantidad de veces que se consulte la marca de pedido de tiempo. De allí que la base de tiempo que podemos poner a cada temporizador y contador está expresada en la siguiente tabla.

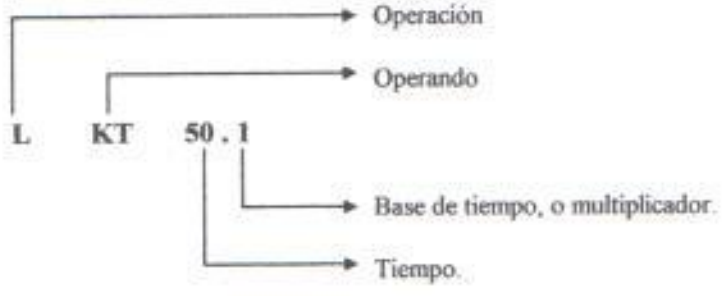
Base	0	1	2	3
Factor	0.01s	0.1s	1s	10s

Es de tener en cuenta que el factor de la base de tiempo es la multiplicación por lo introducido, es decir, si ponemos 300.o decimos 300 * 0.01 que es lo mismo que 3, o pones 003.3 que son en total 30 seg.

Al arrancar una operación de tiempo se toma como valor de temporización la palabra almacenada en el AKKU 1. De allí la necesidad de haber hecho previamente una carga de este registro.

Un temporizador puede ser cargado con cualquiera de los siguientes tipos de datos:

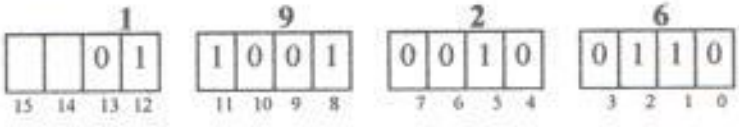
- KT .....que es un valor constante.
- DW.....palabra de datos.
- EW.....palabra de entrada.
- AW.....palabra de salida.
- MW.....palabra de marcas.



Ejemplo 2: (**Carga de un Temporizador como palabra de entrada, de salida, de marcas, o de datos**)

Consideremos para nuestro análisis la siguiente instrucción: L DW 3.

Para introducir la información en esta instrucción, es necesario que la palabra esté cargada en código BCD. Pongamos una demostración gráfica a fin de entender mejor:



En este caso tenemos que el bit 12 y 13, constituyen la base del tiempo o factor multiplicador, si tiene el número 1, según la tabla que presentábamos antes será 0.1s la base de tiempo. Mientras que el número que será cargado será 926, codificado en BCD. Esto indica que en el temporizador se cargarán 92.6 seg. solo que para ejecutar estos 92 seg. tendrá que hacer el programa 926 ciclos. Con esto como explicáramos antes aumenta la precisión del tiempo, pues cuanto menor sea la base de tiempo, mayor la precisión.

Supongamos que queremos depositar el número 375 mili segundos en la palabra de datos 2, la información la sacaremos del módulo de datos 5.

Los pasos que tenemos que seguir, son los siguientes:

A **DB** **5** (leemos la entrada de datos # 5).

L **KT** **375.0** (Hacemos una carga de 375 con multiplicador de 0.01 seg.).

T **DW** **2** (Transferimos el contenido de AKKU1 a la palabra de datos #2).

3.9.4. OPERACIONES DE TIEMPO

Operación	Operando		Significado
SI	□	□	**Arrancar como impulso una temporización:** La temporización se arranca con el flanco creciente del VKE. Con VKE "0" se pone a cero la temporización. Cualquier consulta durante la temporización indica estado de señal "1"
SV	□	□	**Arrancar una temporización como impulso prolongado** La temporización se arranca con el flanco creciente del VKE. Un VKE igual a 0 no afecta a la temporización. Cualquier consulta durante la temporización indica el estado de señal "1"
SE	□	□	**Arrancar como retardo a la conexión una temporización** La temporización se arranca con el flanco creciente del VKE. Con VKE igual a "0" se pone a "0" la temporización. Las consultas indican estado de señal "1" cuando ha transcurrido la temporización y en la entrada sigue aplicado el VKE.
SS	□	□	**Arrancar como retardo a la conexión una temporización** La temporización se arranca con el flanco creciente del VKE. El VKE igual a "0" no afecta a la temporización. Las consultas indican "1" cuando ha transcurrido la temporización. El estado de señal es "0" cuando la temporización ha sido borrada con la operación "R".
SA	□	□	**Arrancar como retardo a la desconexión una temporización** La temporización se arranca con el flanco decreciente del VKE. Con VKE igual a "1" se ajusta la temporización a su valor inicial. Las consultas indican estado de señal "1" mientras el VKE en la entrada sea "1" o corra la temporización.
R	□	□	**Reponer (borrar) una temporización** La temporización se repone al valor inicial mientras el VKE sea "1". Un VKE "0" no afecta a la temporización. Las consultas indican estado de señal "0" mientras la temporización se reponga o no haya sido todavía arrancada.
Identificador	**T**		**Parámetro** 0....127 (depende de cada PLC)

Estas son operaciones que se usan normalmente en la temporización ya sea para encender o apagar algo transcurrido ese tiempo.

Todas estas funciones, como podemos darnos cuenta en la tabla, en excepción de "Borrar una temporización", arrancan con el cambio de flanco del VKE de "0" a "1". A diferencia de lo que se supondría en el funcionamiento de estos temporizadores, el valor interno se desconoce, solo emite una señal de que ya se a cumplido el tiempo. Por ello se lo carga con una valor inicial, y este se va decrementando en forma automática en una unidad de acuerdo a la base de tiempo que le asignamos.

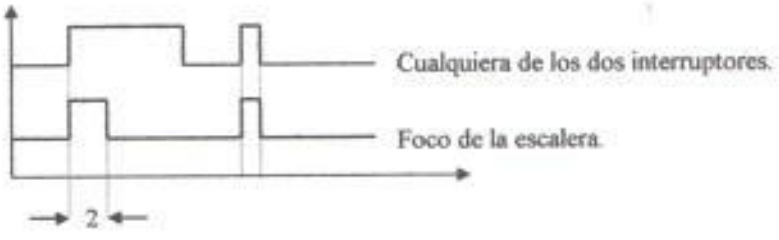
Si llegara a cambiar el flanco durante la corrida de tiempo del temporizador, este vuelve a ajustarse al valor inicial, y se arranca nuevamente.

Hagamos un ejemplo de cada uno de estos casos a fin de que quede más claro.

Ejemplo: **(Arrancar como impulso una temporización)**

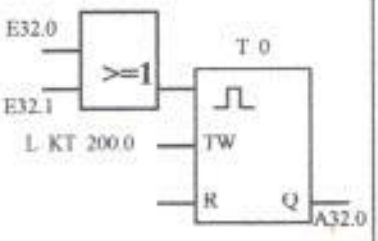
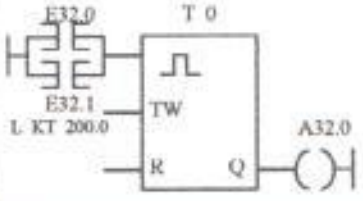
Supongamos que tenemos una luz en una escalera que queremos que se encienda tan rápido se conecte cualquiera de los dos interruptores ubicados en los extremos de la misma. Además queremos que permanezca encendido durante 2 minutos, y sea independiente de si se ha quedado activado más tiempo, pero en caso de que se

desactive al llegar al otro extremo y aun no se ha cumplido el tiempo, igual se desactive como indicamos en el diagrama de tiempo.



Asignemos a los interruptores la dirección de entrada E32.0 y E32.1 y a la salida A32.0.

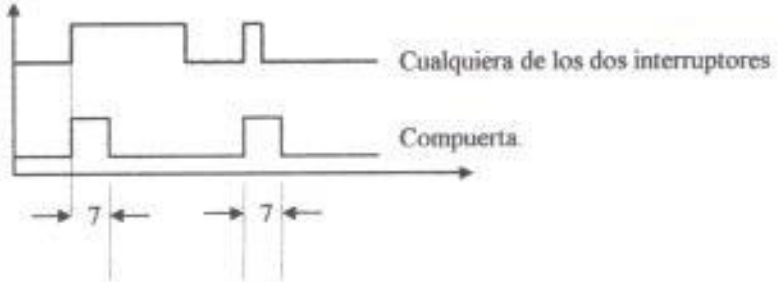
Recordamos que las "E" indican entradas, y que las "A" indican las salidas. Donde los números 32 indican el número de módulo que en este caso se encuentra en el PLC, el número después del punto, indica que número de bit corresponde de ese módulo.

AWL			FUP	KOP
O	E	32.0		
O	E	32.1		
L	KT	200.0		
SI	T0			
U	T0			
=	A	32.0		

Ejemplo: **(Arrancar una temporización como impulso prolongado)**

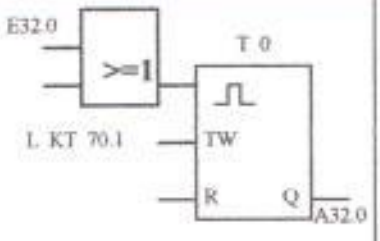
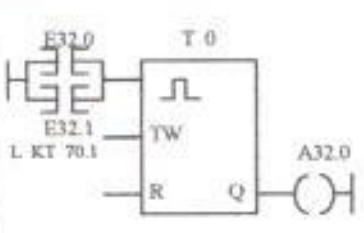
Tenemos una compuerta de agua controlada por un pistón neumático accionado por una electroválvula. El drenado de agua debe de ser de 7 seg. con independencia del tiempo de pulsación de los dos pulsadores que se encuentran en diferentes oficinas.

En este caso debemos tener en cuenta que el problema nos afirma que debe de abrirse la compuerta durante 7 seg. para ello tratemos de hacer lo más exacto posible usando la menor base de tiempo, con ello decimos 70.1 que es 70 x 0.1 seg. con ello haremos que aumenten los ciclos de programa.



Asignemos a los pulsadores la dirección de entrada E32.0 y E32.1 y a la electroválvula que es la salida A32.0 .

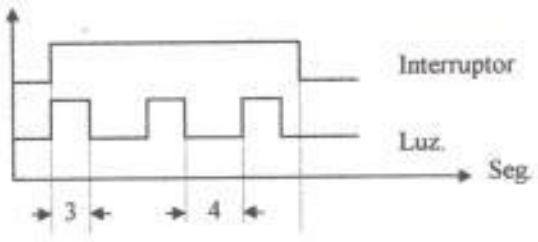
Recordamos que las "E" indican entradas, y que las "A" indican las salidas. Donde los números 32 indican el número de módulo que en este caso se encuentra en el PLC, el número después del punto, indica que número de bit corresponde de ese módulo.

AWL			FUP	KOP
O	E	32.0		
O	E	32.1		
L	KT	70.1		
SV	T1			
U	T1			
=	A	32.0		

Ejemplo: **(Arrancar como retardo a la conexión una temporización)**

Supongamos que queremos hacer un programa en el que parpadee una luz, mientras esté activado un interruptor. Tenemos que recordar que el estado del temporizador depende del VKE, por lo tanto antes de invocar a la instrucción

del temporizador tenemos que asegurarnos que el VKE se encuentra activo. Hagamos de que el parpadeo dure encendido 3 seg. y apagado dure 4 seg. por lo tanto tendrá un periodo de ciclo de 7 seg.



Asignemos al interruptor E32.0 y al foco que es la salida A32.0 .

Recordamos que las "E" indican entradas, y que las "A" indican las salidas. Donde los números 32 indican el número de módulo que en este caso se encuentra en el PLC, el número después del punto, indica que número de bit corresponde de ese módulo.

AWL

U	E	32.0	Si se cumple que está activada la entrada 32.0, y además
UN	M	0.0	no está activa la marca de memoria 0.0, entonces
L	KT	300.0	cargará 300x0.01 = 3 seg.
SE	T2		Mientras no se cumpla el tiempo, no haga nada, solo cuente.
U	T2		Si se ha cumplido el tiempo, entonces
S	M	0.0	Encienda la marca 0.0 poniéndole el valor de "1"
NOP	0		Comando de no operación.

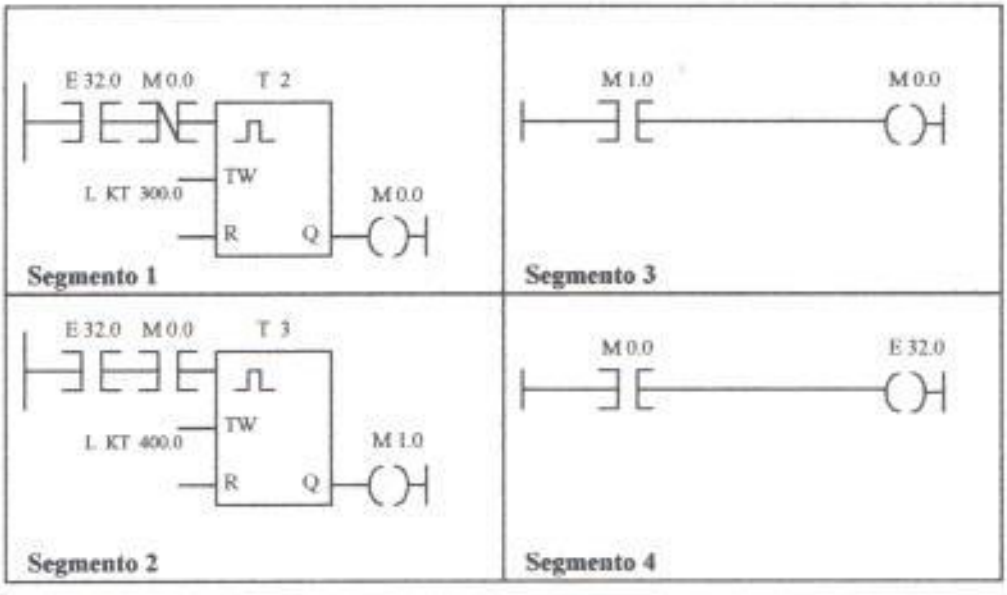
U	E	32.0	Si sigue activado el interruptor, y
U	M	0.0	la marca de memoria 0.0 está activada, entonces,
L	KT	400.0	cargar en AKKU 1 el valor de 4 seg. que usará luego T3.
SE	T3		Si no se cumplió aun los 4 seg. siga contando.
U	T3		Si se cumplió el tiempo, entonces
R	M 0.0		Apague la marca de memoria dándole el valor de "0".
NOP	0		No operación.
U	M	0.0	Si está activa la marca de memoria 0.0.
=	A	32.0	active la salida 32.0
BE			Fin de módulo.

Es de acostumbrarse al programar que las entradas estén solo al principio del programa y en el resto del programa no figure más que las marcas que hacen de espejo de estas entradas, eso ocurre lo mismo que con las salidas, mientras no termine el programa conviene manejarse solo con marcas. En el final del programa asígnese a las marcas las correspondientes salidas. De esta forma el programa se hace más fácil de manejar y adecuar a cualquier PLC, ya que en caso de implantar el programa en un PLC que tenga ocupadas sus entradas y salidas que requiere el programa, solo será cuestión de cambiar al principio y final ciertas instrucciones de asignación de marcas, mientras que en caso contrario, habría que revisar cada módulo del programa que no tuviera ninguna entrada o salida.

El PLC cuenta con lo que denominan el **Perro Guardián** que no es más que un temporizador de proceso, eso es: la corrida del programa, tiene un tiempo límite que en caso de sobrepasar este tiempo, el PLC pasa a un estado de error, encendiéndose la luz de STOP. Eso nos asegura de que no quedará en estado de inhibición.

En este caso haremos la traducción del programa en KOP, pero, no en todos los casos lo haremos ya que en la medida que los programas se hacen más largos demandan mayor espacio los diagramas y realizar en algunos casos artificios.

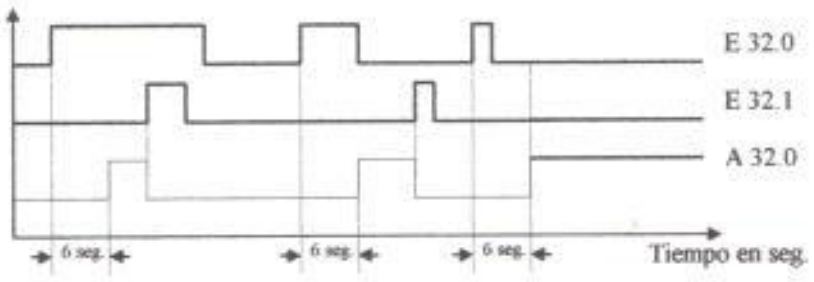
KOP



Ejemplo: (Arrancar como retardo a la conexión una temporización)

Activar una alarma que está conectada en la salida 32.0 que es activada, 6 segundos después de que el sensor de una puerta (E 32.0), es activado. La alarma debe de ser activada con independencia de si la puerta a sido cerrada o abierta varias veces antes de que se active. La alarma debe poder ser desactivada mediante la entrada E32.1.

Este hecho podemos reflejarlo mediante el siguiente diagrama de tiempo:



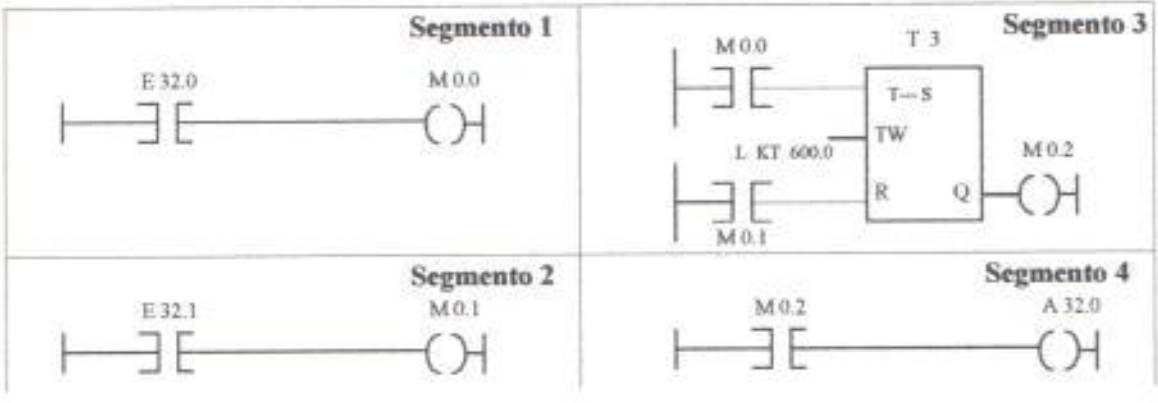
Ahora elaboramos el programa:

AWL

U	E	32.0	Asignamos a cada entrada una marca.
=	M	0.0	A la marca 0.0 le asignamos el sensor.
U	E	32.1	
=	M	0.1	A la marca 0.1 le asignamos la entrada 32.1 para desactivar.
NOP	0		No operación.
U	M	0.0	Si se activó la marca (se activó el sensor) entonces,
L	KT	600.0	cargue 6 segundos.
SS	T3		Si aun no a pasado el tiempo, siga contando y no haga nada.
U	M	0.1	Si se activó esta marca es que quiero desactivar la alarma,
R	T3		entonces reseteo el temporizador, poniéndolo en 0.
NOP	0		No operación
U	T3		Si está activo el temporizador, entonces,

=	M	0.2	activar la marca 0.2 correspondiente a la salida.
NOP	0		No operación.
U	M	0.2	Asignamos a la marca su salida correspondiente.
=	A	32.0	
BE			Fin de módulo.

KOP



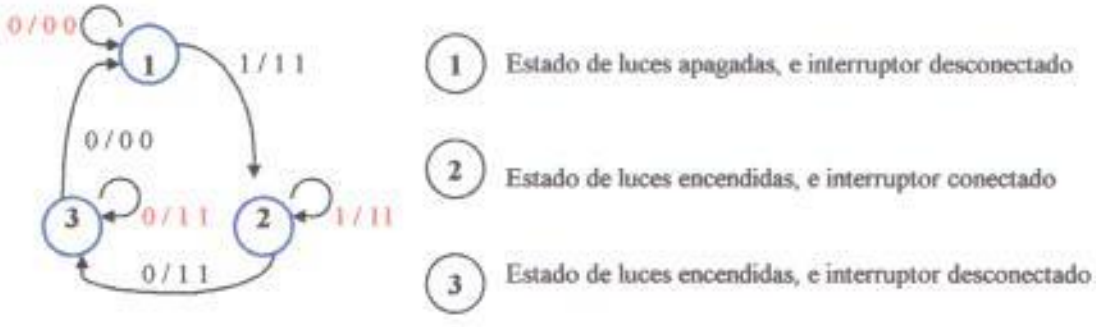
Ejemplo: (Arrancar como retardo a la desconexión una temporización)

Deseamos programar el PLC para que un sistema de alumbrado de una bodega, dé tiempo a abandonarla después de "n" segundos que ha sido apagado el interruptor de

iluminación de la misma. Donde "n" será un tiempo que lo establece cada persona que entra mediante la palabra residente a partir del módulo 14.

Mediante el diagrama de tiempo haremos nuestro sistema, pero agregaremos una herramienta más que en ocasiones es de vital importancia para visualizar mejor el problema.

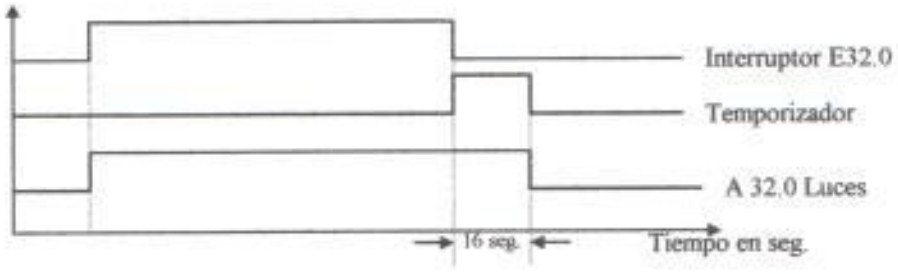
El formato del diagrama es: E 32.0 (entrada) / T3 (estado del temporizador) luces .



Explicamos brevemente este diagrama de estados primitivos: El sistema se encuentra en un estado de luces apagadas mientras el interruptor está apagado. Una vez que está activado el interruptor, el sistema pasa a un segundo estado de luces encendidas, y, permanecerá en él mientras que el interruptor permanezca activado. En el momento en que el interruptor se conmuta y se desactiva, el sistema pasa a un tercer estado en

el que corre un tiempo, y permanecerá en este estado de luces encendidas e interruptor desactivado, mientras el temporizador corre. Una vez cumplido este tiempo, las luces son apagadas y retorna a su estado anterior de luces apagadas e interruptor desactivado.

Mediante un diagrama de tiempo tendremos lo siguiente:

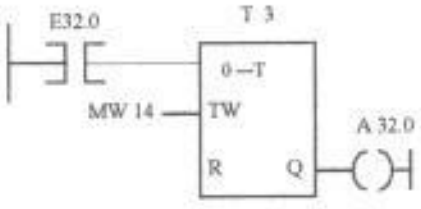


AWL

U	E	32.0	Si está activado el interruptor, entonces
L	MW	14	carga la palabra que se encuentra a partir del módulo 14.
SA	T3		Mientras no pase el VKE a cero, no se activa.
NOP	0		No operación (se usa para separar un módulo de otro).
U	T3		Si ya se inició la cuenta del tiempo y está dentro del tiempo,
=	A	32.0	mantener activa la salida 32.0 de las luces.
BE			Fin de módulo.

Tenemos que tener muy en cuenta que la operación de U T3 está activa mientras el VKE es igual a uno o mientras está corriendo el tiempo.

KOP



3.9.5. OPERACIONES DE CONTAJE:

En este caso se trata de otro elemento muy usado dentro de la programación de los PLC y de la programación en general, se trata de un contador. Este puede incrementar o decrementar un valor, además de poder cargar un valor previo. Los límites de conteo son de tres cifras, es decir, que tenemos un rango de 000 hasta 999.

Operación	Operando		Significado
S	□	□	**Activar (cargar) un contador:** El contador se activa con el flanco creciente del VKE.
R	□	□	**Borrar (reponer) un contador** El contador se pone a cero siempre que VKE es "1"
ZV	□	□	**Incrementar un contador (contaje hacia adelante)** Con flanco creciente se incrementa en 1 el valor del contador. Con VKE igual a "0" no se modifica el valor del contador
ZR	□	□	**Decrementar un contador (contaje hacia atrás)** Con flanco creciente del VKE reduce en 1 el valor del contador. Con el VKE igual a "0" no se modifica el valor del contador.
Identificador	**Z**		**Parámetro** 0....127 (depende de cada PLC)

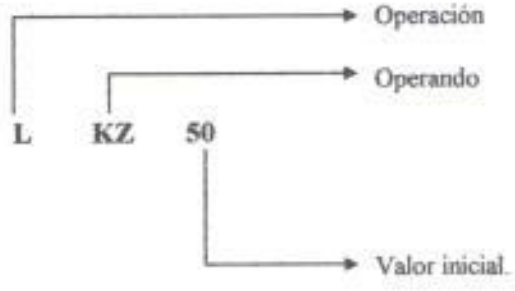
Para cargar un contador como explicáramos anteriormente al definir la función de carga "L", se toma el valor de lo que encuentra como contenido del registro AKKU 1. De allí la importancia de hacer previamente una carga voluntaria del valor. Cuando se invoca una de estas funciones se activan los contadores internos del hardware del PLC. Existe una gran similitud entre

cargar un contador y cargar un temporizador, ya que lo único que varía es la "T" por la "Z".

Un contador puede ser cargado con cualquiera de los siguientes tipos de datos:

- KT .....que es un valor constante.
- DW.....palabra de datos.
- EW.....palabra de entrada.
- AW.....palabra de salida.
- MW.....palabra de marcas.

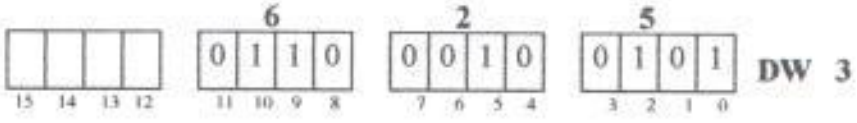
Con esta instrucción del L KZ 50, el contador se iniciará en un valor de 50, eso significa que de allí podemos manipular al 50 ya sea en forma creciente o decreciente dependiendo la instrucción que escogemos (ZV, ZR).



Podemos hacer una carga mediante las marcas de datos mediante la siguiente instrucción:

L DW 3

Eso viendo en forma gráfica, y suponiendo que el número contenido en esta marca es el número 625 tendremos los siguiente:

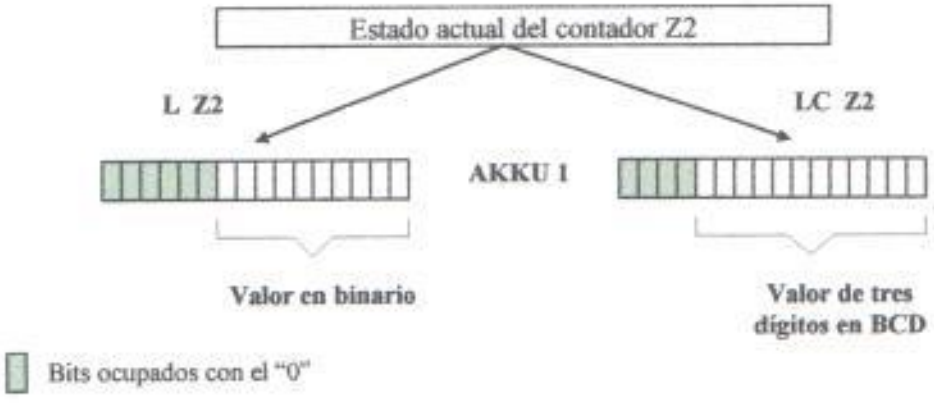


Como vemos los valores están codificados en código BCD solo usa hasta el bit 11 para representar el número de tres dígitos.

Para fines de la programación debemos de tener en cuenta que **al consultar el contador** este dará el valor de 1 a menos que contenga en su interior el valor de "0".

La **salida del estado actual del contador** puede ponerse en el registro del AKKU 1 utilizando una operación de carga, esta función de carga no impide el

funcionamiento normal del contador, es decir que igual sigue contando. Para poder sacarlo a través de un visualizador digital lo mejor es utilizar la operación "Cargar codificadamente". Esto también es válido para el caso del temporizador.



Hagamos unos ejemplos con los contadores:

Ejemplo: **Activar un contador "S" y decrementar "ZR"**

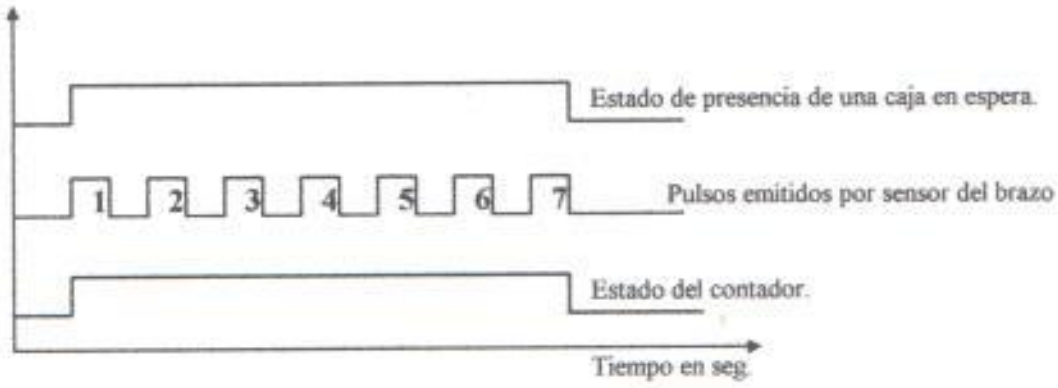
Programemos un PLC para que éste, cuando reciba la señal de que existe ya una caja en espera, cargue en su interior 7 unidades de piezas mecánicas a ser exportadas.

Contamos para hacer eso con un brazo mecánico que está activado mientras el contador está en "1". Para ello, contamos con un sensor que me emite un pulso cada vez que el brazo mecánico introduce una pieza en la caja. Este pulso lo recibimos a través de la entrada E32.1. Mientras que la señal de existencia de caja la recibimos por la entrada E32.0. La salida del contador que va a activar el brazo mecánico es la salida A32.0. Cuando se llega a completar una caja existe un mecanismo que al detectar que el contador ya llegó a completar las 7 piezas retira la caja.

Hacemos previamente un diagrama de tiempo que nos ayude a comprender mejor el enunciado del problema, que conviene que se le dé su importancia.

Como veremos en el diagrama de tiempo, tanto la señal que recibe el brazo mecánico, como, la señal que emite el contador y estado de presencia de la caja es exactamente la misma señal. Mientras que el sensor del brazo emite cada cierto tiempo, el depósito de una pieza mecánica dentro de la caja.

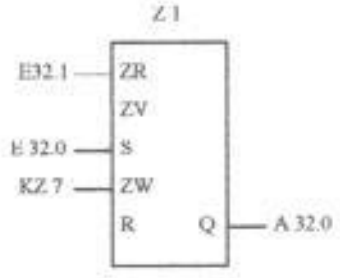
En este caso en la programación de este pequeño problema usamos la instrucción S, que es una instrucción de encendido, de activado, independientemente del estado de la señal, si el VKE se encuentra en "1" pasará la señal al estado "1" y permanecerá en él hasta que se la apague. Pero es de advertir en el listado del programa, que aquí la usamos, para activar al contador.



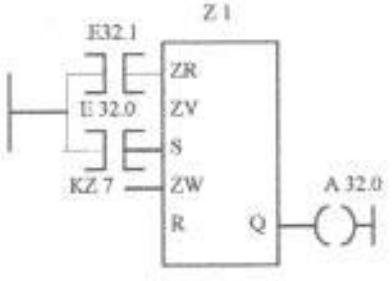
AWL

U	E	32.1	Si está activa la entrada 32.1 del sensor del brazo,
ZR	Z1		decremente en uno el valor del contador.
U	E	32.0	Si, está activo el sensor detector de la presencia de la caja,
L	KZ	7	entonces, cargue el contador con el valor 7.
S	Z1		Active el contador.
U	Z1		Si está activo el contador, entonces
=	A	32.0	active la salida 32.0 que activa al motor del brazo para trabajar.
BE			Fin de módulo.

FUP



KOP

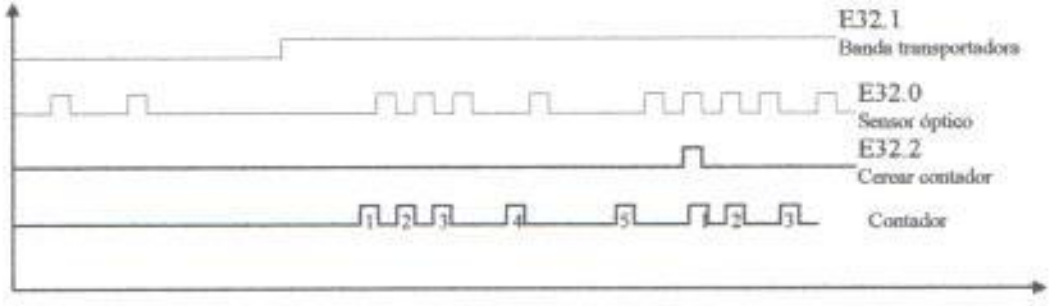


Ejemplo: **Borrar un contador "R" e incrementar "ZV".**

Programar el PLC para que sea capaz de detectar cuantas piezas pasan delante de un sensor óptico que estará conectado en la entrada E32.0. Esta cuenta la realizará siempre que el operario active una banda transportadora. Para saber si está en marcha la banda transportadora contamos con un detector de funcionamiento del motor, este detector lo tenemos conectado en la entrada E32.1. En caso de que la banda

transportadora esté detenida, no queremos que cuente nada, a pesar de que el sensor detecte presencia. Contamos con un pulsador que cada vez que se active pondrá en cero el contador, éste está conectado en la entrada 32.2. Además queremos tener solo una salida que se active cada vez que el contador tenga un estado de cero.

Hagamos esto de acuerdo al siguiente diagrama de tiempo.

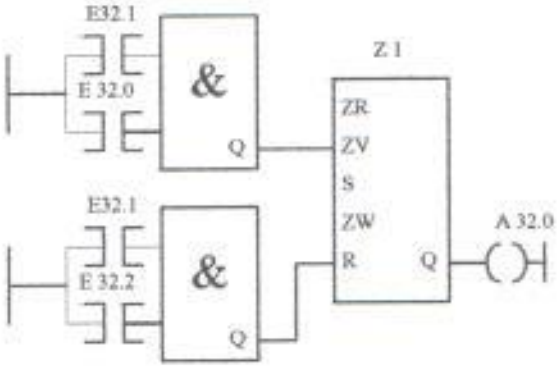


AWL

U	E	32.1	Imponemos la condición más fuerte. Si está activo
U(			el motor de la banda transportadora,
U	E	32.0	y además está activo el sensor óptico, entonces
ZV	Z1		incrementará el contador en uno.
U	E	32.2	Si activo el cerear contador, entonces
R	Z1		borro el contador.
NOP	0		Separo este bloque con el otro con una no operación.
U	Z1		Si el contador tiene algún valor diferente de cero,

=	A	32.0	activo la salida 32.0.
)			Cierro, la condición fuerte.
BE			Instrucción de final de módulo.

KOP



3.9.6. OPERACIONES DE COMPARACIÓN:

Estas son operaciones con que cuenta el PLC para que puedan mediante ellas comparar valores de variables, ver si ya llegaron al límite, etc. Esta función se ejerce mediante la comparación de los dos acumuladores AKKU1 y AKKU2, de allí que hay que cargarlos previamente a la comparación con los valores que se irán a comparar. Debemos tener muy en claro que esta función actúa en forma directa sobre el VKE, ya sea activándolo o desactivándolo. En

caso de que se cumpla la condición, se activará (VKE = 1), en caso de que no cumpla se desactivará (VKE = 0).

Operación	Operando		Significado
!=F	□	□	**Comparación respecto a la igualdad:** Los contenidos de los AKKUs se interpretan como configuración binaria y se comparan respecto a igualdad.
><F	□	□	**Comparación respecto a la desigualdad** Los contenidos de los AKKUs se interpretan como configuración binaria y se comparan respecto a la desigualdad.
>F	□	□	**Comparar respecto a la superioridad** Los contenidos de los AKKUs se interpretan como números en coma fija. Se investiga si el operando en AKKU2 es mayor que el de AKKU1.
>=F	□	□	**Comparación respecto a la superioridad o igualdad.** Los contenidos de los AKKUs se interpretan como números en coma fija. Se investiga si el operando en AKKU2 es mayor o igual que el de AKKU1.
<F	□	□	**Comparación respecto a la inferioridad.** Los contenidos de los AKKUs se interpretan como números en coma fija. Se investiga si el operando en AKKU2 es mayor que el de AKKU1.
<=F	□	□	**Comparación respecto a la inferioridad o igualdad.** Los contenidos de los AKKUs se interpretan como números en coma fija. Se investiga si el operando en AKKU2 es mayor o igual que el de AKKU1.

La comparación se efectúa siempre en forma independiente de si el VKE está activo o no, el resultado de esa comparación es binario, es decir, 0 ó 1. Debemos tener en cuenta al comparar dos variables que ellas tienen el mismo formato, pues en caso de comparar diferentes formatos, el resultado de la comparación ya será de por si solo falso, así sea que el valor sea el mismo, pero al estar en diferentes formatos el PLC botará un resultado erróneo.

Hagamos algunos ejemplos que nos ayuden a comprender mejor estos operandos de comparación.

Ejemplo: **Comparación respecto a la igualdad.**

Queremos programar el PLC para que cuente las personas que ingresan en las graderías de un estadio. Solo contamos con dos entradas principales, una al norte y otra al sur. Queremos además que nos avise cuando hay igual cantidad de personas en una graderia y en otra, o en cual hay más. Contamos con un sensor de paso óptico en cada entrada, que están conectados de la siguiente forma: E32.0 gradería norte, E32.1 gradería sur. Las salidas indicadoras son A32.0 de igualdad, A32.1 si norte es mayor que sur y A32.2 si sur mayor que norte. Podemos extraviar en este programa algunas instrucciones de comparación innecesarias, ya que en caso de no cumplir la igualdad, tampoco la desigualdad en un sentido, eso implica que necesariamente se cumple en el otro sentido. De allí que podemos usar solo dos comparaciones, a saber: una de igualdad y una de desigualdad.

Como los contadores trabajan en el mismo formato, no tenemos que tener en cuenta se si estará o no la respuesta del programa errónea.

Nota: (en la programación en este caso no tendremos en cuenta el valor tope de los contadores, asumiremos un sistema automático de reseteo).

En estos casos como los programas no son muy largos, y su interpretación no es tan complicada, no lo estructuramos, ni respetamos la convención directa de usar solo las entradas al principio y las salidas solo al final. Cuando los programas ya sean de mayor envergadura como lo haremos al final, nos preocuparemos en hacerlo en forma estructurada y en bloques pequeños que ataquen en forma separada los diversos problemas, que engloban el ejercicio.

AWL

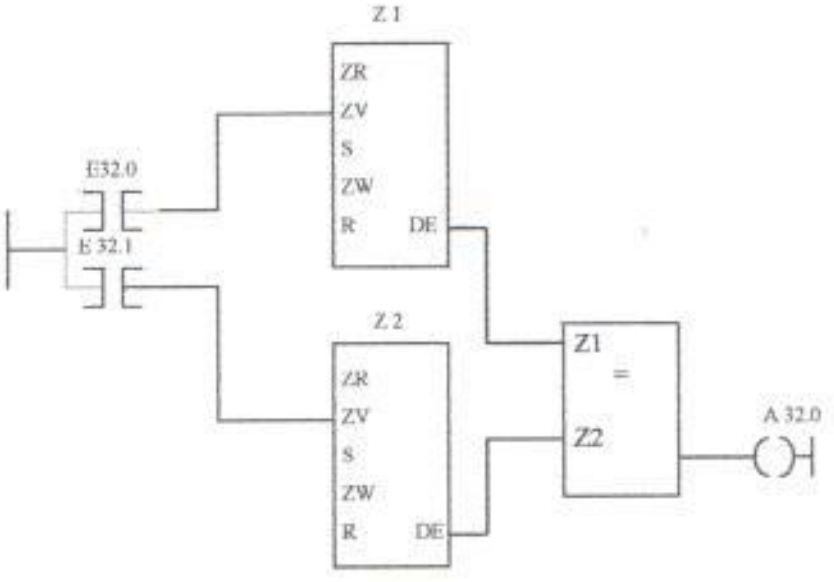
U	E	32.0	Si está activo el sensor de la entrada norte
ZV	Z1		incremente el contador de norte.
U	E	32.1	Si está activo el sensor de la entrada sur
ZV	Z2		incrementar el contador de sur.
NOP	0		Separamos módulo de entrada con comparación.
L	Z1		Cargamos en AKKU1 el contador norte.
L	Z2		Desplaza al c. norte al AKKU2 y se carga en AKKU1

NOP	0		el contenido de c. sur. Y separamos el bloque de carga.
!=F			Si son iguales los AKKUs entonces,
=	A	32.0	activo la salida A32.0 de igualdad.
UN	A	32.0	Si no está activa la salida igualdad, hay uno mayor.
>F			Comparamos si c. norte es mayor que c. sur.
=	A	32.1	Si cumple la condición, activamos la salida A32.1.
UN	A	32.0	Si no está activa la igualdad, y
UN	A	32.1	tampoco está activa c. norte mayor que c. sur,
=	A	32..2	activamos la salida de c. sur mayor que norte.
BE			Fin de módulo.

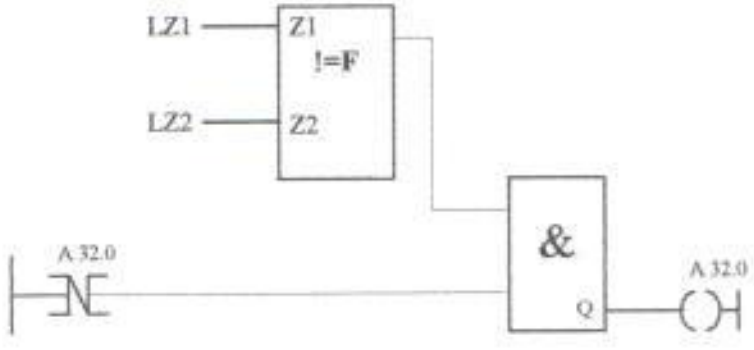
KOP/FUP

Cuando se programa en KOP, se hace mediante segmento, donde cada uno de los segmentos puede llevar como máximo una bobina de salida. Llamamos bobina a las salidas ( ).

Segmento1:



Segmento 2:



Segmento 3:

3.9.7. OPERACIONES ARITMÉTICAS:

En éste tipo de funciones podemos encontrar diversas utilidades, ya que nos facilita en muchas ocasiones la suma o la resta que a nivel de software se complicaba un poco sustituir estas operaciones, para ello se ha incluido en el PLC estas operaciones. La restricción es que actúan directamente sobre el contenido de los registros AKKUs, razón por la cual necesariamente deben de cargarse los registros antes de realizar estos comandos. Además el contenido es tratado como números de coma fija y es de tener en este caso la precaución de que ambos operandos tengan un mismo formato. El resultado de la operación será depositado en el AKKU1.

Recordamos que el contenido de AKKU2, no es más que el contenido previo del AKKU1, que antes de cargarse desplaza su valor al AKKU2, y posteriormente se carga.

A diferencia de otras operaciones dentro del PLC, esta no requiere que se tenga en cuenta el estado del VKE, lo que indica que con VKE "1" ó "0" la instrucción igual se ejecuta. De igual forma que la realización de ellas no modifican al contenido del VKE. Pero las indicaciones si se activan dependiendo del resultado.

Operación	Operando		Significado
+F	□	□	**Sumar:** Se suman los contenidos de los AKKUs. El resultado de la operación se pone en el AKKU1.
-F	□	□	**Restar:** El contenido de AKKU1 se resta del contenido del AKKU2 y el resultado se transfiere en el AKKU1

Observación: Existe dentro de la gamma de PLCs algunos que permiten además operaciones de multiplicación y división, para ello se dispone de módulos funcionales integrados.

Ejemplo: Operaciones Aritméticas.

Queremos controlar el flujo de personas que entra en un cine y en base a ello poder tomar decisiones que limite la cantidad de entradas, contamos con dos entradas, donde, cada una de ellas cuenta con un contador del PLC a saber: Z0 y Z1, la cantidad de personas que se desea de acuerdo al día será puesta dentro de una entrada tipo palabra en el puesto 3. Queremos que en caso de que sean iguales nos active la salida A32.0. Además deseamos visualizar la suma en la salida tipo palabra número12.

AWL

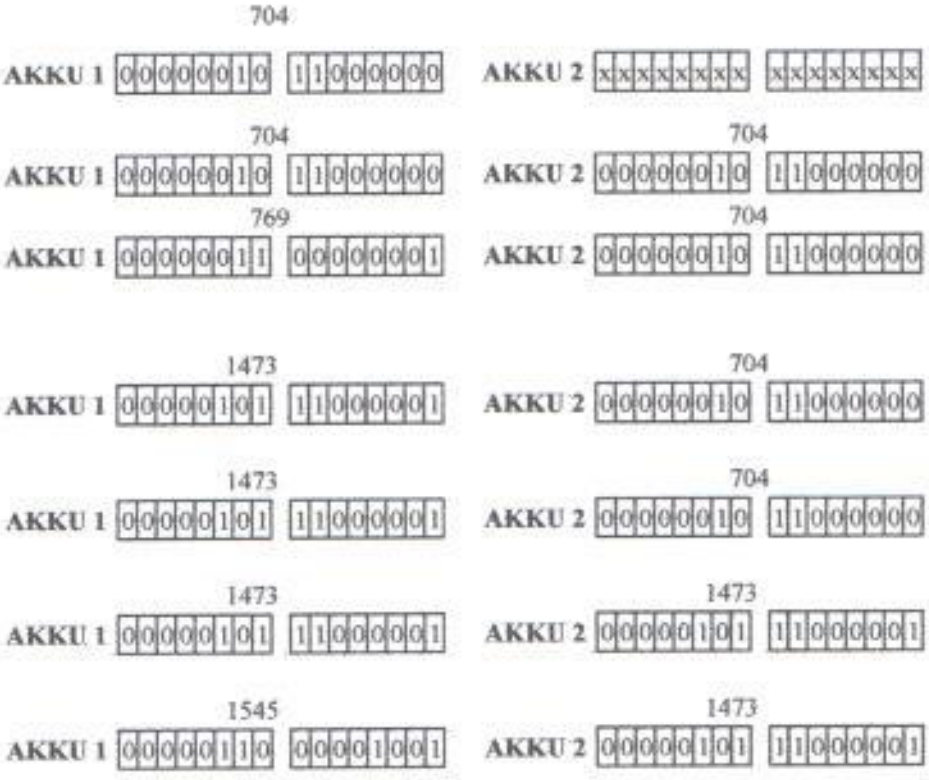
L	Z0	Se carga en el AKKU1 el contenido del contador 0.
L	Z1	Se desplaza el AKKU1 al AKKU2 y se carga el contador 1 en
NOP	0	el AKKU1.

+F			Llamamos a la operación de suma. El resultado está en AKKU1
T	AW	12	Transferimos el contenido de AKKU1 a la salida tipo palabra desde la posición 12.
NOP			
L	DW	3	Desplazamos el contenido de la resta, pasándolo al AKKU2 y cargo el contenido de la palabra de la posición 3 en AKKU1.
!=F			Comparación de la igualdad.
=	A	32.0	
BE			Fin de módulo.

Viendo a los AKKUs durante el programa lo tendríamos de la siguiente forma:

Suponemos la suma de 704 y 769 y comparo con 1545 que son las personas que pueden entrar hoy.

Iremos graficando a cada instante las variaciones de los AKKUs, desde la carga del primero, desplazamiento, guardado y carga respectivamente.


704
AKKU 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0
AKKU 2 x x x x x x x x x x x x x x x x
704
AKKU 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0
704
AKKU 2 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0
769
AKKU 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1
704
AKKU 2 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0
1473
AKKU 1 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1
704
AKKU 2 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0
1473
AKKU 1 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1
704
AKKU 2 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0
1473
AKKU 1 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1
1473
AKKU 2 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1
1545
AKKU 1 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1
1473
AKKU 2 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1

3.9.8. OPERACIONES DE LLAMADA DE MÓDULO:

Estas funciones son usadas en la programación estructurada, son usadas para llamar a las diferentes subrutinas dentro de un programa principal, existen de tipo condicional e incondicional. El caso de la condicional, solo dará el brinco, si el VKE está activo lo que indica que es una instrucción dependiente del VKE. Normalmente se le antepone una condición, que en caso de que cumpla, entrará en la subrutina y en caso contrario pasará a la siguiente instrucción. En caso de las instrucciones que son de salto incondicional, significa que no tiene en cuenta el VKE. Los módulos que se ejecutan siempre, es conveniente que sea normalmente una función en la que se le transfiera parámetros (apartado que trataremos más adelante). Es de buscar que en un programa principal, solo está con llamadas condicionales e incondicionales, y que el resto sean módulos que realicen funciones especificas, pero que sean re utilizables.

En el caso de este tipo de PLC sobre el cual basamos nuestro estudio, cuenta con dos llamadas de salto, luego existe otras instrucciones que sirven para llamar a módulos de datos, etc. En el caso de las llamadas a los módulos de datos, esta es una instrucción dependiente del VKE, lo que indica que si queremos que se ejecute esa instrucción será necesario que se asegure de que

VKE se encuentre en "1". Pero no interrumpe la ejecución normal del programa.

La instrucción de creación o de borrado de módulo de datos, tiene independencia del VKE lo que significa que al igual que la llamada incondicional, siempre se ejecuta.

Operación	Operando		Significado
SPA	□	□	**Salto absoluto (incondicional):** La ejecución del programa continúa en otro módulo, con independencia del VKE. Esto no afecta al VKE.
SPB	□	□	**Salto condicional:** Con el VKE = " 1 " se salta a otro módulo. De no ser "1" el VKE se seguirá ejecutando el mismo módulo. En este caso el VKE se pone a "1".
Identificador			**Parámetro**
OB			0...255
PB			0...255
FB			0...255
SB			0...255

Ejemplo: **Llamada absoluta (incondicional) y incondicional.**

Queremos hacer un programa estructurado, para que encienda luces, dependiendo de la condición las rojas o las verdes. Asignaremos las salidas A32.0, A32.1 los que son de color rojo, y A32.2, A32.3 a las que son de color

verde. Queremos que siempre se enciendan las rojas, pero, si la entrada E32.0 está activa, además se encienda las verdes.

Para estructurar el programa lo haremos de acuerdo a las normas, así que el módulo principal debe de ser OB1, los dos restantes podemos hacerlos PB o FB.

Advertencia: Cuando programamos en forma estructurada, antes de invocar a un módulo, es preciso que este ya exista y esté elaborado con sus parámetros en caso de que los llevare, o si no en el momento de invocarlo el ensamblador emite una señal de error. Además al ingresar en la creación de un módulo, este pedirá un nombre mnemotécnico a fin de facilitar el entendimiento del que realiza esa función aun que el PLC solo conoce el nombre de PB o FB, etc. pero no obstante al invocar un módulo, el PLC presenta el nombre para que se asegure el usuario que es el que necesita, además en caso de contar con parámetros, me los exhibe a continuación del nombre, y me pide su correspondencia en el programa principal. Los nombres de los módulos son limitados de 8 caracteres normalmente.

PB 0:

Nombre: ACT_ROJO

```
S     A     32.0
S     A     32.1
BE
```

PB1:

Nombre: ACT_VERD

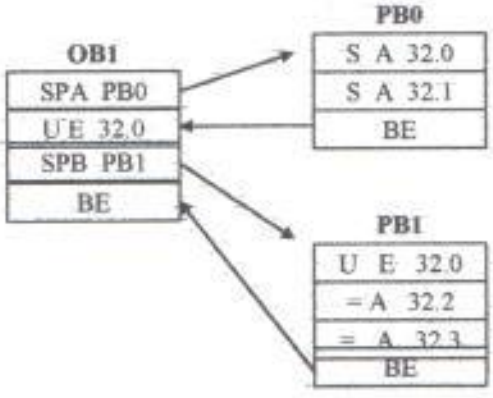
```
U     E     32.0
=     A     32.2
=     A     32.3
BE
```

OB1: (PRINICIPAL)

```
SPA       PB0
Nombre: ACT_ROJO
U         E     32.0
SPB       PB1
Nombre: ACT_VERD
BE
```

Si describimos paso a paso lo que hace el programa es lo siguiente:

Primero el PLC busca al OB1 y lo ejecuta, así que encuentra primero una llamada y le da el control al módulo que fue llamado, hasta que ese módulo se lo retorna. Recordamos que tiene un sistema de **perro guardián** en el que si en un tiempo límite el módulo no devuelve al OB1 el control, el PLC pasa a un estado de error y se detiene en la corrida del programa. Viendo en forma gráfica funciona así:



Ejemplo: **Llamada de un módulo de datos "A DB".**

Estos módulos se los llama siempre en forma incondicional, en consecuencia todos los demás módulos actúan directamente sobre estas variables del módulo de datos. Al igual que el caso anterior, estos módulos deben ser creados antes que el programa principal que los usa.

En el módulo de programa 1, se necesita de la información que ha sido programada en el DW2 del DB8. Además el resultado del cálculo ha sido guardado en DW4 del DB 15.

Haciendo el programa en AWL tenemos:

AWL

A	DB	8	Llamamos al módulo de datos 8.
L	DW	2	Cargamos en el AKKU1 la información de palabra 2.
NOP	0		No operación.
A	DB	15	Llamamos al módulo de datos 15.
T	DW	4	Transferimos el contenido de AKKU1 a la salida de palabra 4.

Ejemplo: **Creación y borrado de un módulo de datos.**

Esta instrucción no llama a ningún DB, sino que ella lo genera. Si se desea utilizar datos, se lo llama usando la instrucción A DB.

Antes de E DB es necesario indicar al AKKU1 cuantas palabras de datos comprende el módulo, ya que en caso de encontrar la longitud de cero, el módulo de datos será borrado, eliminándose de la lista de direcciones. A partir de ese instante es como si no existiera. El módulo no desaparece realmente, sino que se lo mantiene como "no válido" hasta que se comprima la memoria del PLC.

La instrucción de creación queda sin efecto en caso de que el módulo que pretendemos crear ya existe. Estos módulos pueden contener hasta 256 palabras de datos (DW0....255).

AWL

```
L    KF   +127
E    DB   5
L    KF   +0
E    DB   5
```

Con la **primera instrucción y la segunda instrucción** lo que hacemos es cargar el AKKU1 con el valor de cuantas palabras de datos comprende el

módulo que son 127. En ese momento se crea en una zona de la RAM del PLC el módulo de datos 5 con una longitud de 128 palabras, registrándose en la lista de direcciones de módulos. La **segunda instrucción** esta no tendrá validez si el contenido o el valor del AKKU1 es cero. En el caso de la **tercera instrucción** cargamos en el AKKU1 el valor de 0 a fin de definir la longitud de cero en la longitud del módulo de datos. Este es un requisito según indicábamos antes para borrar un módulo de datos. Queda claro que si en el AKKU1 está ahora el valor de cero, significa que en el AKKU2 está el contenido de la dimensión anterior del módulo que fue creado. Con esta instrucción se declara no válido el módulo de datos 5 que está en la RAM dentro del PLC, y se suprime de la lista de módulos.

3.9.9. OPERACIONES DE FINAL DE MÓDULO:

Estas instrucciones como lo indica su nombre, son usadas una vez que se ha terminado de elaborar un módulo.

Cuando uno programa en el ensamblador, este normalmente al terminar un módulo y aceptarlo para que sea introducido dentro del PLC, el ensamblador

se asegura de que la última instrucción que existe en el módulo sea BE de lo contrario el mismo lo pone. Después de esta instrucción que existe en el módulo sea BE de lo contrario el mismo lo pone. Después de esta instrucción no puede haber otra, razón por la cual en ocasiones al agregar instrucciones dentro de un programa ya existente, y las ponemos después de BE, el programa de ensamblado, detecta que hay un BE ya existente antes de la última instrucción entonces o bota un error de ensamblaje o las versiones más nuevas, lo borran y lo ponen al final, como última instrucción.

Operación	Operando	Significado
BE		**Terminar módulo (fin de módulo)** Con independencia del VKE se finaliza el módulo actual. El programa se sigue ejecutando en el módulo desde donde se llama. El VKE puede arrastrarse, pero no modificarse. BE es siempre la última instrucción de un módulo.
BEA		**Terminar módulo de forma absoluta (incondicional)** Con independencia del VKE se finaliza el módulo actual. El programa se sigue ejecutando en el módulo desde donde se llama. El VKE puede arrastrarse, pero no modificarse.
BEB		**Terminar módulo de forma condicional:** Con el VKE "1" se finaliza el módulo actual. El programa se sigue ejecutando en el módulo desde donde se llama. Al cambiar el módulo no varía el VKE, sigue siendo "1". Con el VKE "0" no se ejecuta la operación. El VKE se pone a "1" y el programa sigue ejecutándose linealmente.

A diferencia del BE, el BEA esa una instrucción que no va necesariamente al final como última instrucción, sino que puede ir por el medio. Provoca el retorno dentro de un módulo. También puede evitarse su ejecución en los FBs con una instrucción de salto, es decir que después de una instrucción condicional de salto, podemos poner esta instrucción de tal forma que si no cumple la condición, forzamos el final del módulo. Esta instrucción no es dependiente del VKE para su ejecución, por lo que, nos aseguramos que si no salta el recorrido secuencial del programa en la condición anterior, se ejecutará esta instrucción de final. En el caso de la instrucción BEB, que también es de finalización de módulo, este fuerza el final dentro de un módulo, pero con dependencia del VKE, lo que significa que si VKE = 1, la instrucción se ejecuta, caso contrario queda sin efecto.

Ejemplo: **Operaciones de retorno.**

Deseamos que dentro de un módulo FB16 sea introducido lo siguiente:

Si la entrada E 32.0 está activa, active la salida A 32.0 y chequee si las demás entradas están activadas, si no se cumple esta condición queremos que finalice la ejecución del módulo.

Si la entrada E 32.1 está activa, queremos que active la salida E32.1 y finalice el módulo. Si no se cumple, que prosiga con el chequeo de las demás entradas E 32.3 y 32.4, en caso de que estén estas dos activas, active la salida A32.3.

AWL

FB 16

Nombre: final

```
      :U    E     32.0
      :=    A     32.0
      :SPB  =aqui
      :BEA
aqui  :U    E     32.1
      :=    A     32.1
      :BEB
      :U    E     32.3
      :U    E     32.4
      :=    A     32.3
      :BE
```

3.9.10. OPERACIONES ADICIONALES:

Estas son operaciones que solo pueden programarse en forma de lista de comandos, lo que es lo mismo decir, en lenguaje AWL.

La instrucción de STP, es una instrucción que fuerza al PLC a entrar en una estado de detención de la corrida del programa, esto es de suma importancia en aquellos módulos que son de detección de fallos dentro del funcionamiento normal de aquello que es controlado por el PLC. Por ejemplo, en caso de que sea un proceso productivo y se detecta que falta la materia prima para una determinada sección del proceso, es necesario detener el equipo a fin de evitar posibles averías por falta de material. Esta instrucción se maneja con las llamadas interrupciones algunas veces, ya que con un interrupción puedo ejecutar una subrutina de interrupción en la que detengo en forma adecuada el proceso.

Las operaciones de NOP, son instrucciones que permiten reservar o sobre escribir posiciones de memoria.

Las que son de estructuración de imagen, tenemos que saber que dentro de un módulo, es posible dividirlo en segmentos partes de un programa utilizando

las operaciones de estructuración BLD. Las operaciones nulas y de estructuración de imágenes solo tienen significado si se las representa dentro del programa STEP5 que es el ensamblador del lenguaje de los PLC de la Siemens.

Operación	Operando		Significado
STP			**Stop al finalizar la ejecución del programa (en el OB1)** Se acaba de termianar la ejecución del programa; se saca la PAA. A continuación el PLC pasa en estado de STOP.
NOP 0			**Operación nula** En la memoria RAM se ponen a "0" 16 bits.
NOP 1			**Operación nula** En la memoria RAM se ponen a "0" 16 bits.
BLD	□	□	Operaciones de estructuración de imagen para el PLC
Identificador →		← **Parámetro**	130, 131, 132, 133, 255

3.10. OPERACIONES COMPLEMENTARIAS:

Este tipo de operaciones tiene como restricciones que, solo puede ser ejecutada dentro de módulos funcionales a diferencia de las operaciones básicas que pueden ser programadas dentro de cualquier tipo de módulos. Además de esto, solo pueden programarse este tipo de operaciones con lenguaje AWL que es la lista de instrucciones.

3.10.1. OPERACIÓN DE CARGA.

En esta también permite la carga de datos en el acumulador AKKU.función el concepto no cambia respecto al que tenía como operación básica, es decir,

Operación	Operando		Significado
L	□	□	**Cargar** Con independencia del VKE se carga en el AKKU1 una palabra procedente de los datos de sistema
Identificador BS	**Parámetro** 0....255		

Ejemplo: **Operaciones de carga** (Ejemplo extraído directamente de Simatic de Escoles Profesionales Salesianas de Serriá).

Para parametrizar la comunicación vía SIMEC L! usando los datos de sistema, en el AKKU 1 usando los datos de sistema, en el AKKU 1se registrarán los números del aparato de programación PG y de esclavos contenidos en el SD57.

AWL

....

....

```
L     BS    57          Carga el AKKU1 con los números de PG y esclavo.
....  ....  ....
```

3.10.2. OPERACIÓN DE LIBERACIÓN

La siguiente instrucción es una instrucción de liberación, que se usa para poder ejecutar incluso sin cambio de flanco las operaciones que describimos a continuación:

- Arrancar un temporizador.
- Activar (cargar) un contador.
- Incrementar y decrementar un contador.

Operación	Operando		Significado
FR	□	□	**Liberación de un temporizador / contador** Con independencia del VKE se carga en el AKKU1 una palabra procedente de los datos de sistema
Identificador			**Parámetro**
	T		0....127
	Z		0....127

Ejemplo: **Operación de retorno** (Ejemplo extraído directamente de Simatic de Escoles Profesionales Salesianas de Serriá).

Un temporizador T2 se arranca mediante E32.0 como impulso prolongado (ancho del impulso 50 seg.). Esta temporización activa A 32.4 mientras dura el impulso. Se desea rearrancar nuevamente la temporización siempre que vuelva a activarse A32.5.

AWL

```
U    E    32.0
L    KT   500.1
SV   T2
U    T2
=    A    32.4
...  ...  ....
...  ...  ....
U    A    32.5
FR   T2
BE
```

3.10.3. OPERACIONES DE PRUEBA DE BITS (SOLO EN EL PLC S5-95).

Este tipo de operación es muy útil cuando trabajamos a nivel de bit, ya que ellas nos permitirán testear el estado de un solo bit determinado, podemos

activarlo o desactivarlo. Es una función puntual. Estas operaciones de prueba de bit deben de estar siempre al comienzo de una operación combinacional (lógica). Hay veces que en la programación de una determinada tarea se hace mucho más práctico trabajar por bits, es decir, manejar palabras, bytes, bits ya que existen muchas funciones que nos permiten manejarlo con mayor sencillez que si trabajáramos como contactores o simplemente como entradas y salidas puntuales con independencia, ya que a la hora de almacenar cualquier información en los AKKUs, debemos tener en cuenta que ellos los almacenan como si fueran 16 bit, lo que pasa, que llena de ceros lo que no es transferido.

Operación	Operando		Significado
P	□	□	**Probar si está en 1 (uno) un bit** Con independencia del VKE se consulta un bit individual. Según su estado de señal se influencia el VKE.(tabla siguiente)
PN	□	□	**Probar si está en 0 (cero) un bit** Con independencia del VKE se consulta un bit individual. Según su estado de señal se influencia el VKE.(tabla siguiente)
SU	□	□	**Activar incondicionalmente un bit.** Con independencia del VKE se pone a "1" el bit deseado. No se influencia e VKE.
RU	□	□	**Borrar incondicionalmente un bit.** Con independencia del VKE se pone a "0" el bit deseado. No se influencia el VKE.

Estado de P y PN sobre el VKE.

Operación	P		PN	
Estado de señal del bit en los operandos indicados	0	1	0	1
Resultado de combinación (VKE)	0	1	1	0

Ejemplo: **Operaciones de prueba.**

Realizar un programa en el que contamos con un sensor de piezas que nos sirve de contador, queremos que cada 8 piezas que vengan se ejecute el módulo FB0 o FB1, es decir, las 8 primeras piezas se les hará el proceso que indique el módulo funcional FB0, mientras que de la 9 a la 16 se le ejecutará el módulo funcional FB1, de allí se repite para la ocho que siguen el módulo el FB0, y de esa forma cada ocho piezas los módulos FB0 y FB1 se irán intercambiando. Una vez que lleguemos a las 400 piezas queremos que se detenga el proceso previó a activarse la salida 33.0

AWL

Asignamos al sensor de piezas la entrada E32.0

El contador de cada 8 le asignamos Z0.

Al contador de 400 piezas le asignamos Z1.

Lógica:

Haremos que cada vez que se detecte una pieza se incrementen ambos contadores, el Z0 cada vez que alcance el valor de 8 lo podemos resetear, pero ponemos una bandera para saber a cual de los dos módulos le pertenece el mando. Para saber si ha llegado al valor esperado lo haremos mediante el test de los bits que deben de activarse a fin de que dé esa cantidad.

La bandera, cada vez que se resetee el contador se preguntará el estado de la bandera, si está desactiva (se llama a FB0) la activamos, mientras que se está activada (se efectúa FB1) se desactiva, de esa manera se ira intercambiando los módulos.

Usaremos la entrada 32.1 para iniciar el proceso, poniendo el contador de 400 a cero de nuevo. Recordamos que el número estará en BCD en el AKKU.

```
:U    E    32.0        ***** Módulo de conteo *****
:ZV   Z    0
:ZV   Z    1
:                      ***** Módulo de reseteo *****
:U    E    32.1
```

```
:R    Z    1
:R    Z    0
:R    M    0.0
:R    M    0.1
:
:LC   Z    1     **** Módulo de terminación de proceso ****
:T    DW   12
:P    D    12.10 Cuando llega a 0100/0000/0000 que es 800 en BCD se detiene.
:BEB
:                **** Módulo de intercambio de FB0 a FB1****
:LC   Z    0
:T    DW   8
:P    D    8.3   Cuando llega a 0000/0000/1000 que es 8 en BCD cambio.
:S    M    0.1   Como esta instrucción depende del VKE, hasta que cumpla la
:                condición de arriba de P D 8.3 no se ejecutará S M 0.1
:U    M    0.1   Si se activó es que llegó a 8 y hay que poner cero a Z0
:R    Z    0
:
:U    M    0.0   Si la memoria 0.0 está activa y además se cumple que
:U    M    0.1   ya alcanzamos el valor de 8, entonces,
:R    M    0.0   desactivar la memoria para que vaya a FB0 y
:R    M    0.1   poner en cero de nuevo la marca 0.1.
:
:UN   M    0.0   Si la memoria 0.0 no está activa y además se cumple que
```

```
:U    M    0.1    ya alcanzamos el valor de 8, entonces,
:S    M    0.0    activar la memoria para que vaya a FB1 y
:R    M    0.1    poner en cero de nuevo la marca 0.1.
:
:BE
```

Ejemplo: Operaciones de prueba.

Ahora miraremos de usar la instrucción de RU o SU mediante un programa.

Compliquemos el programa anterior, hagamos el mismo programa anterior pero le agregamos el caso de que, hagamos de cuenta de que si no ha llegado una pieza 5 seg. después de la última detección es que se ha producido un error y queremos que se detenga el proceso y el PLC pase a un estado de error y se detenga la ejecución. Y si se demora 2.5 seg. de la última detección hacer que el proceso comience de nuevo el proceso, sin modificar el programa anterior.

Para resolver este problema tendremos que usar la función SU para forzar a 400 la cuenta del contador Z1 y en caso de que se complete los 5 seg. tendremos que usar la función de STP. Además tendremos que hacer que constantemente se resetee los temporizadores al detectarse una pieza.

AWL

```
:U    E    32.0
:R    T0
:R    T1
:
:UN   E    32.0
:L    KT   250.0     Cargo 2.5 seg. de tiempo.
:SE   T    0
:L    KT   500.0     Cargo 5 seg. de tiempo.
:SE   T    1
:
:U    T    0        Si completo el tiempo de 2.5 seg.
:R    Z    1        Primero pongo en cero todo el contador y
:SU   Z    1.8      fuerzo a "1" los bits 8, 7 y 4 que son los necesarios
:SU   Z    1.7      para que el contador tenga el número 400, ya que el
:SU   Z    1.4      contador tiene el número en binario
:
:U    T    1        Si cumple los 5 seg. sin que se haya reseteado el
temporizador
:STP                entonces detener el programa y el PLC entra en esta de error.
:
:                   ////////// PROGRAMA ANTERIOR //////////
:U    E    32.0     ***** Módulo de conteo *****
```

```
:ZV   Z    0
:ZV   Z    1
:U    E    32.1      ***** Módulo de reseteo *****
:R    Z    1
:R    Z    0
:R    M    0.0
:R    M    0.1
:
:LC   Z    1         **** Módulo de terminación de proceso ****
:T    DW   12
:P    D    12.10     Cuando llega a 0100/0000/0000 que es 400 en BCD se
detiene.
:BEB
:                    **** Módulo de intercambio de FB0 a FB1****
:LC   Z    0
:T    DW   8
:P    D    8.3       Cuando llega a 0000/0000/1000 que es 8 en BCD cambio.
:S    M    0.1       Como esta instrucción depende del VKE, hasta que cumpla la
:                    condición de arriba de P D 8.3 no se ejecutará S M 0.1
:U    M    0.1       Si se activó es que llegó a 8 y hay que poner cero a Z0
:R    Z    0
:U    M    0.0       Si la memoria 0.0 está activa y además se cumple que
:U    M    0.1       ya alcanzamos el valor de 8, entonces,
:R    M    0.0       desactivar la memoria para que vaya a FB0 y
```

:R	M	0.1	poner en cero de nuevo la marca 0.1.
:UN	M	0.0	Si la memoria 0.0 no está activa y además se cumple que
:U	M	0.1	ya alcanzamos el valor de 8, entonces,
:S	M	0.0	activar la memoria para que vaya a FB1 y
:R	M	0.1	poner en cero de nuevo la marca 0.1.

:BE

3.10.4. OPERACIONES AND, OR Y XOR ENTRE PALABRAS:

Estas operaciones son prácticas cuando queremos separar un bit de una palabra, o queremos cambiar el contenido de una entrada dependiendo de si está una u otra, etc. Ellas permiten combinar bit a bit los contenidos de los AKKUs. Estos son independientes del estado del VKE, es decir, sin importar si está en "1" o en "0" la función se ejecutará siempre. Ahora, tampoco afecta al VKE, pero activan las indicaciones en función del resultado de la operación aritmética.

Para poder ejecutar estas funciones, tenemos que cargar previamente los AKKUs, ya que estas funciones actúan directamente sobre los dos AKKUs. Recordamos que al efectuar la primera carga, estamos cargando el AKKU1, luego, al efectuar una segunda carga, desplazaremos el contenido de AKKU1

al AKKU2, y de allí, lo que esté en los dos archivos recibirán la acción directa de la función.

Operación	Operando	Significado
UW		Combinación "Y" bit a bit.
OW		Combinación "O" bit a bit.
AOW		Combinación "O exclusiva" bit a bit.

Ejemplo: **Operaciones AND, OR y XOR.**

En los bit 0 de la palabra de entrada 12, tenemos la información de una u otra frecuencia, así que necesitamos separar de la demás información, dejando en cero los demás bits menos el correspondiente al 0 (cero). Luego con la misma información de entrada queremos que cada bit se ponga en 1 menos el primer bit (el cero) que queremos que se mantenga en su valor de entrada, ya sea cero o uno. Queremos para un fin determinado poner en 0 todos los bits de la palabra menos el correspondiente al bit 5 que queremos que invierta el valor inicial. Veamos en cada caso que pasa con los contenidos de los AKKUs, recordando que antes de efectuar cualquier operación nos fijemos que esté realmente cargado en los AKKUs los operandos que

queremos. Los resultados de las diferentes operaciones queremos que aparezca en la salida 0,1 y 2 respectivamente.

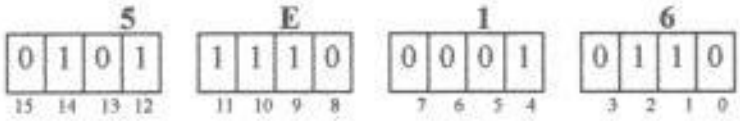
AWL

```
:L     EW    12
:L     KH    0001
:UW
:T     AW    0
:
:L     EW    12
:L     KH    FFFE
:OW
:T     AW    1
:
:L     EW    12
:L     KH    FFDF
:XOW
:T     AW    2
:BE
```

Supongamos que la palabra de entrada sea: 0101/1110/0001/0110

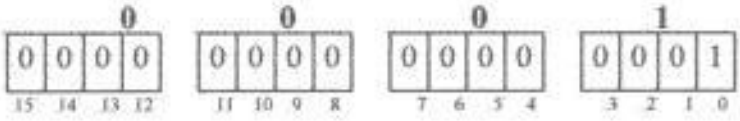
Primero:

AKKU1

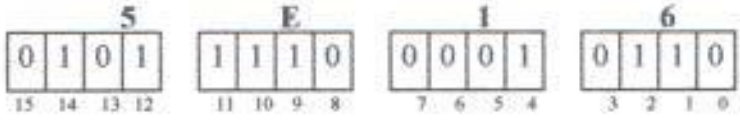


Segundo:

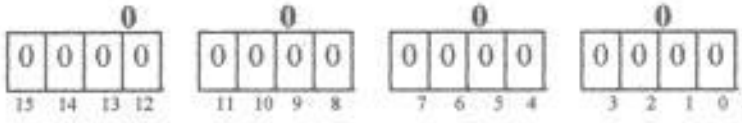
AKKU1



AKKU2

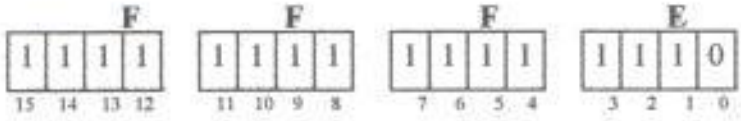


Resultado

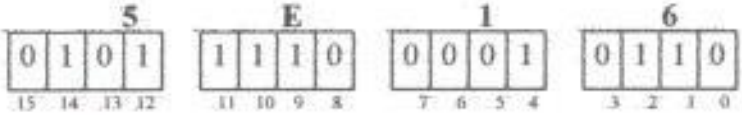


Tercero: tras haber cargado otra vez el AKKU1 con la información de entrada, y efectuar la carga del segundo operando tendremos:

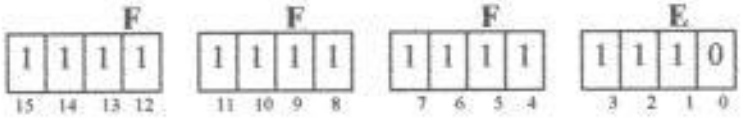
AKKU1



AKKU2

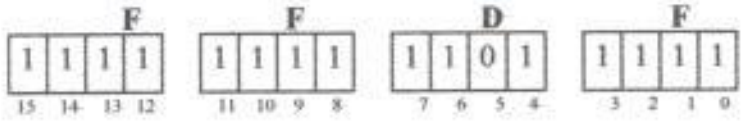


Resultado

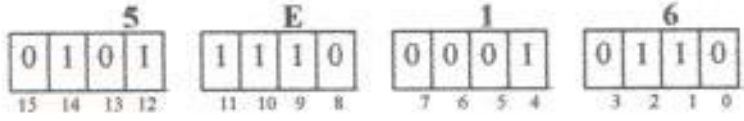


Cuarto: tras haber cargado otra vez el AKKU1 con la información de entrada, y efectuar la carga del segundo operando tendremos:

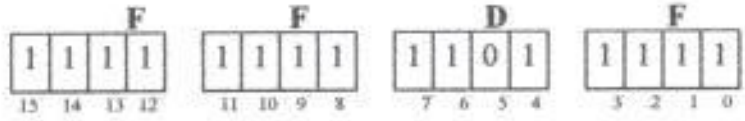
AKKU1



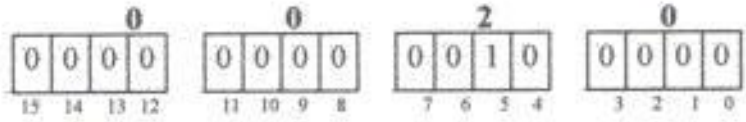
AKKU2



Resultado Parcial



Resultado final



3.10.5. OPERACIONES DE DESPLAZAMIENTO:

Encontramos que mediante las funciones que expondremos a continuación podemos manipular también a los registros, ya que disponemos de las funciones de desplazamiento, mediante ellas podemos desplazar los bits de los AKKUs . Es un desplazamiento binario. Actúa sobre el AKKU1, no modifica el contenido del AKKU2. Recordemos que el desplazamiento a demás de las múltiples aplicaciones que puede tener como mostraremos en los ejemplos, puede además ser usado para las importantes funciones de multiplicación y división. El desplazamiento en este caso hace las veces de multiplicar o dividir por potencias de dos.

La ejecución de las operaciones de desplazamiento no dependen de condiciones, tampoco influencia al estado de VKE. Sin embargo, las operaciones de desplazamiento afectan a las indicaciones. Esto permite consultar con funciones de salto el estado del bit últimamente desplazado.

El parámetro de la instrucción indica cuantas posiciones de bit se desplaza hacia la izquierda SLW, o hacia la derecha SRW el contenido de AKKU1. Las posiciones que quedan libres al desplazar, se llenan con ceros.

Uno de los inconvenientes que encontramos a esta función para ciertas aplicaciones es que perdemos el bit que ha sido expulsado. Una vez ejecutada la operación, el bit 2^0 o el bit 2^{15}, (SLW) influencian el bit ANZ1 que puede entonces evaluarse.

Una operación de desplazamiento con el parámetro "0" (cero) se trata como una operación nula NOP.

Operación	Operando	Significado
SLW	□	**Desplazar hacia la izquierda** Se desplaza hacia la izquierda la configuración en el AKKU1.
SRW	□	**Desplazar hacia la derecha** Se desplaza hacia la derecha la configuración en el AKKU1
	Parámetros	**0....15**

Ejemplo: **Operaciones de Desplazamiento.**

Queremos que el contenido de la palabra, solo quede el bit 5 vigente con su valor en la primera posición correspondiente al bit 0.

Para poder efectuar esta operación primero tendremos que lograr que los demás bits por lo menos de la izquierda sean cero, ya que los de la derecha se irán perdiendo en la medida que son desplazados.

AWL

:L	KH	0020	Haremos que se ejecute una instrucción AND con el
:L	EW	12	dato de entrada, a fin de independizar el bit quinto.
:UW			
:SRW	5		Al desplazar 5 posiciones a la derecha se ha quedado
:BE			en la posición cero.

Ejemplo: **Operaciones de Desplazamiento.**

Queremos hacer un programa el cual enciende un juego de luces conectado a la salida tipo palabra del PLC. Pero cada vez que presionamos la entrada E32.0 se desplazará la luz que inicialmente estaba en el centro hacia la derecha y cada vez que se presione la entrada E32.1 se desplazará hacia la izquierda. El juego se iniciará una vez presionada la tecla E32.3. Queremos que en caso de que la luz llegue a tocar el bit 16, se active una alarma conectada en la salida A32.0, en caso de tocar el menor de los bits el 0, entonces activar una alarma conectada en la entrada a la salida A32.1, y detener el proceso.

AWL

:U	E	32.3	Esperamos la condición de inicio, para cargar AKKU1 con la

```
:L    KH   0100     luz encendida en el centro, y activamos la memoria interna a
:S    M    0.0      fin de que no se intente desplazar el contenido de AKKU1 sin
:T    AW   12       antes haber presionado E32.3.
:
:U    M    0.0      Si está activo la memoria M 0.0 y además,
:U    E    32.0     está activa la entrada E 32.0, entonces
:SRW  1             desplacemos, una posición a la derecha y
:T    AW   12       transferimos el cambio.
:
:U    M    0.0      Si está activo la memoria M 0.0 y además
:U    E    32.1     está activa la entrada E 32.1, entonces
:SLW  1             desplacemos, una posición a la izquierda y
:T    AW   12       transferimos el cambio.
:
:P    D    12.0     Probamos el bit 0 de la palabra 12, y si está activa
:=    A    32.0     encendemos la alarma A32.0
:BEB                y detenemos el proceso.
:
:P    D    12.16    Probamos el bit 16 de la palabra 12, y si está activo
:=    A    32.1     encendemos la alarma A32.1
:BEB                y detenemos el proceso.
:
:BE
```

3.10.6. OPERACIONES DE TRANSFORMACIÓN.

Estas son operaciones que conocemos bajo el nombre de operaciones de complemento a1 y a2, en un caso consiste en cambiar cada bit por su complementario, en el otro consiste en cambiar, los bit que se encuentren a la izquierda del primer bit 1 que se encuentra desde la derecha, ó, lo que es lo mismo cambiar los bits y sumarle 1 al menos significativo.

Estas dos instrucciones también actúan directamente sobre el contenido del AKKU1. De igual forma podemos deducir que son funciones que modifican el VKE.

Operación	Operando	Significado
KEW		**Complemento a 1.** Se invierte bit a bit el contenido del AKKU 1.
KZW		**Complemento a 2.** Se invierte bit a bit el contenido del AKKU1. A continuación se le suma la palabra 0001_H.

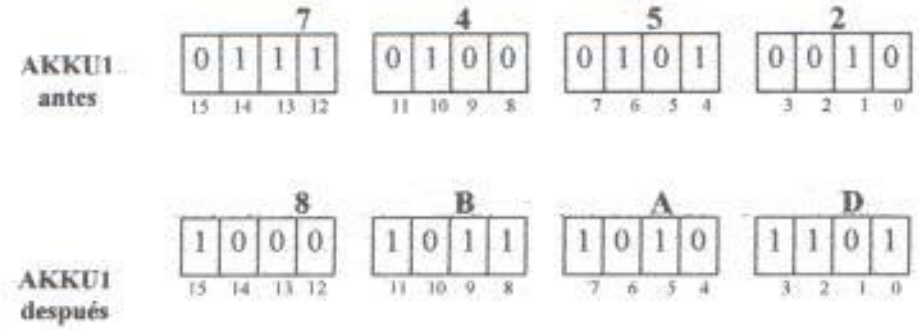
Ejemplo: **Operaciones de Transformación.**

Para ejemplificar, más que buscar una aplicación práctica a este tipo de instrucciones, lo que haremos es, usarlas en una simple lista de comandos, pero si veremos que ocurre con el contenido del AKKU1.

Capturar la información de la palabra de datos 12, y aplicarle a ella, ambas instrucciones. Poner las respuestas en las palabras 14 y 16 respectivamente.

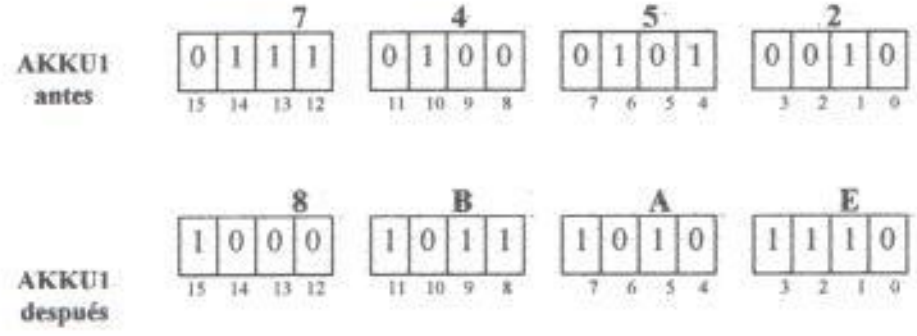
AWL

```
:L    DW   12        Primero cargamos el contenido de la palabra 12 en el
AKKU1.
:KEW                 Le aplicamos la operación de complemento a 1.
:T    AW   14        Finalmente la transferimos a la salida 14.
```




```
:
:L    DW   12        Tenemos que volver a cargar el AKKU1, ya que con la
```

```
:KZW                    instrucción anterior se modificó su valor.
:T    AW   16           Transferimos el resultado a la salida 16.
:BE
```



3.10.7. DECREMENTO E INCREMENTO:

Las operaciones de incremento y decremento son funciones que actúan también sobre el AKKU1. No depende ninguna de las anteriores del VKE y no afecta al VKE ni a las indicaciones.

Ambas instrucciones cuentan con un parámetro, que le indica a la función en qué punto el AKKU1, será modificado. Las operaciones se refieren a los valores decimales; mientras que el resultado se deposita en el AKKU1 en forma binaria. Estas modificaciones se refieren a byte bajo en el acumulador.

CAPITULO IV

4. DEMOSTRACION PRACTICA

4.1. EL BRAZO MECANICO

El brazo mecánico es una demostración muy limitada del potencial de un autómata, ya que si bien es cierto solo se ocupan algunas de las salidas y entradas disponibles, desperdiciando potencial.

4.1.1. CARACTERÍSTICAS TÉCNICAS DEL BRAZO:

Este es un robot inteligente ya que cuenta con un microprocesador, 214 de la SIEMENS Alemana, que lo controla y además tiene una gran versatilidad en la programación del mismo para realizar su manejo ya que puede hacerse mediante diagramas de contactos o lenguaje de programación.

Este brazo puede moverse en forma articulada en cinco ejes a saber: base giratoria, hombro, codo, muñeca, cerrado de pinzas.

Características Generales	
Aplicación	Trabajos que no requieran precisión, investigación.
Construcción	Estructura plásticas (polímeros) y laminas de lata
Precisión del posicionado	+/- 1 cm.
Capacidad de carga	200 a 500gr
Apertura de las pinzas	9.5 cm.
Velocidad de movimiento	Depende del eje
Peso del robot	12 Kg.
Longitud extendida del brazo	1 m.
Radio de alcance	76 cm.
Grados de libertad	5

Margen de giro de los motores (grados desde la horizontal)	
Cuerpo	200°
Hombro	92°
Codo	100°
Inclinación de la muñeca	100°
Rotación de la muñeca	360° (manual)

Alimentación eléctrica y control	
Motores	5 motores DC
Control de velocidad	Solo se puede hacer con pulsos intermitentes, o, disminución de voltaje
Interfaces	RS- 232

La transmisión de los motores a los ejes se hace a través de engranajes como podrá apreciarse en las fotografías, estos son colocados en los motores mediante el cálculo del radio de los engranajes contando desde el eje. Estos engranajes son sujetados fuertemente a las articulaciones mediante unos tornillos. Es muy dificultoso el intentar adaptar el motor en cada uno de los ejes directamente ya que en la medida que tenga más grados de libertad el brazo, mayor será el peso de los motores que tendrá que soportar la estructura, disminuyendo de esa forma la capacidad de carga que pueda soportar el brazo sin soportar daños.

4.1.2. ARMAJE DEL BRAZO Y DIMENSIONES

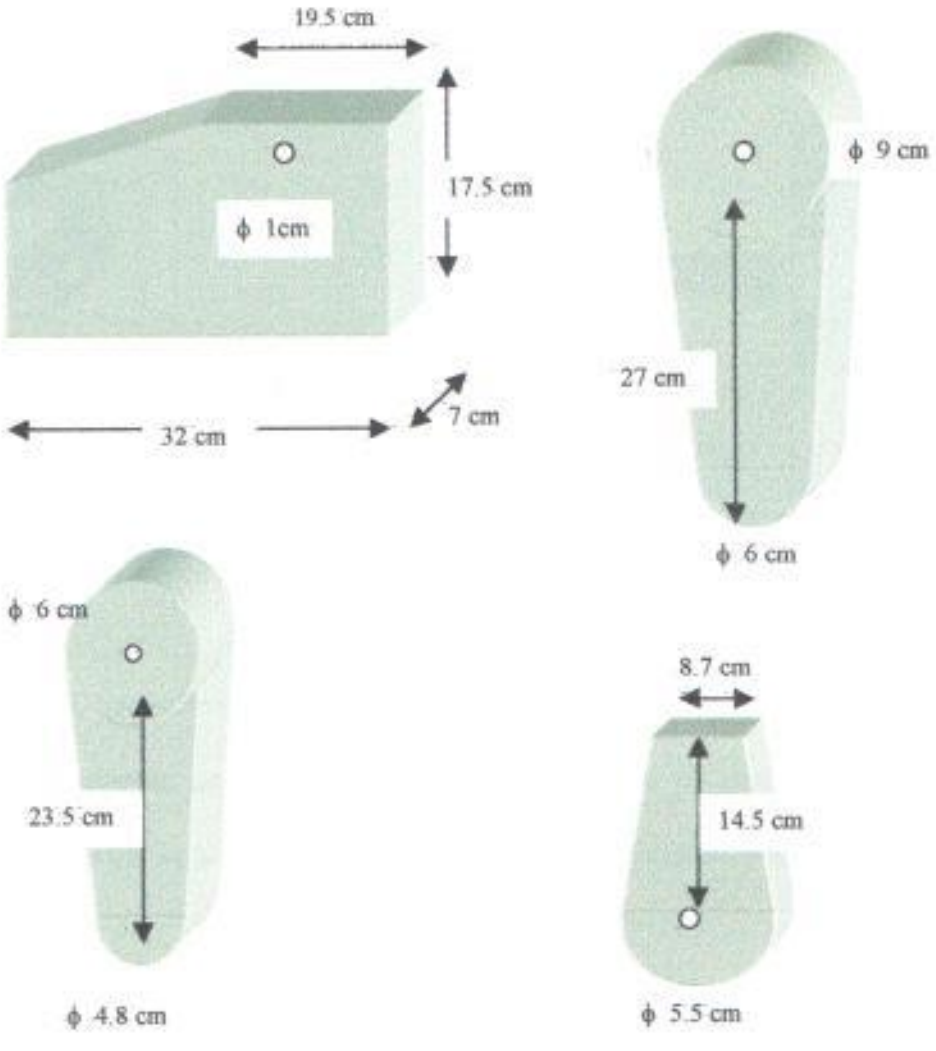


Figura 13 *MEDIDAS DEL BRAZO*

La base se une a su homónimo mediante un puente fuertemente empernado, para el efecto se han procurado unos agujeros con perforaciones concéntricas a fin de poder ocultar el perno, de la siguiente forma:

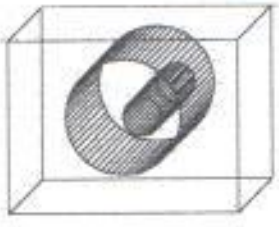


Figura 14 *VISTA FRONTAL DE COLOCACIÓN DE UN TORNILLO EMPOTRADO*

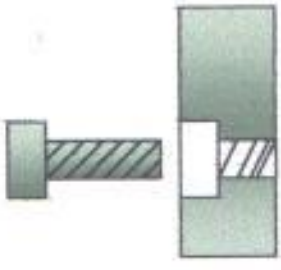


Figura 15 *VISTA LATERAL DE LA COLOCACIÓN DE UN TORNILLO EMPOTRADO*

Las articulaciones se efectúan mediante un eje de acero inoxidable haciéndole una ranura en los extremos a fin de que en ella pueda ser colocado un seguro de cierre elástico, como indicaremos a continuación:

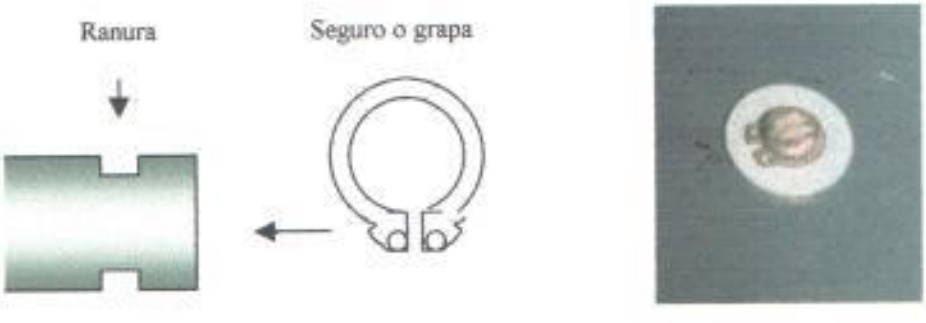


FIGURA 16 FORMA DE COLOCACIÓN DEL SEGURO EN EL EJE

Para poder mantener la estructura firme cuando los motores ejerzan esfuerzos de torsión le ponemos cubiertas las caras con laminas de lata:

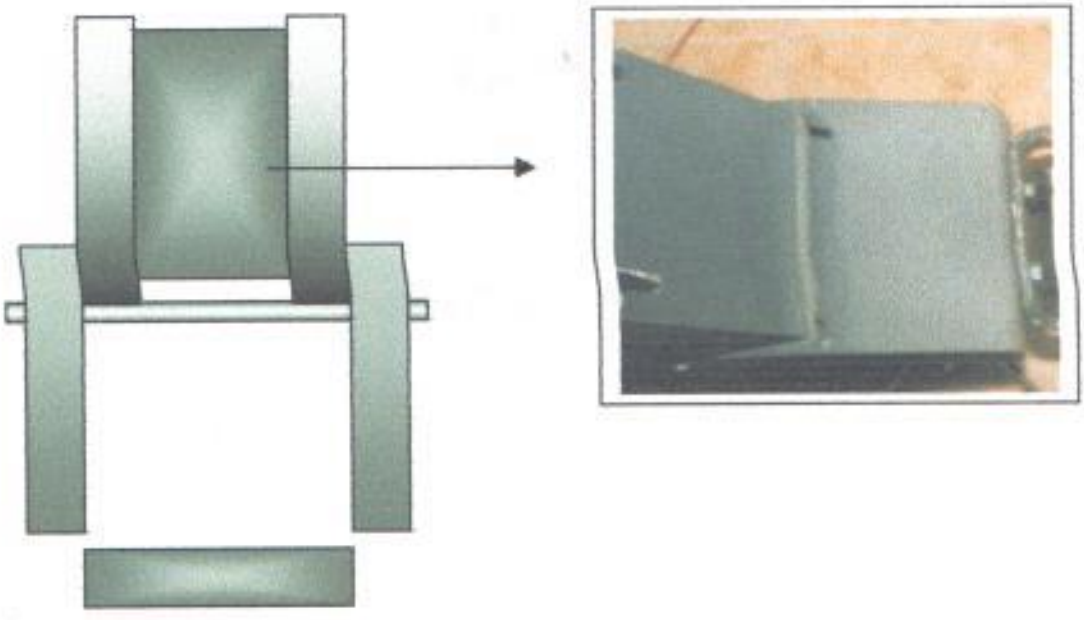


FIGURA 17 *VISTA DE ESTRUCTURA DEL BRAZO*

El brazo tiene un alcance de 100cm desde la posición de descanso hasta su estiramiento máximo, que puede ser logrado tanto en forma horizontal como vertical.

FIGURA 18 VISTA LATERAL DE LA POSICIÓN DE MÍNIMO ALCANCE

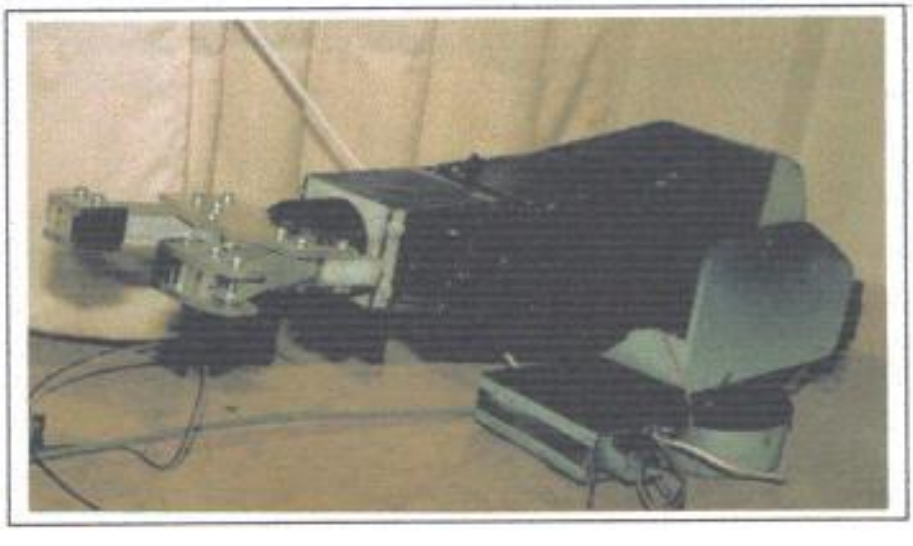


FIGURA 19 VISTA LATERAL DE LA POSICIÓN DE MÁXIMO ALCANCE

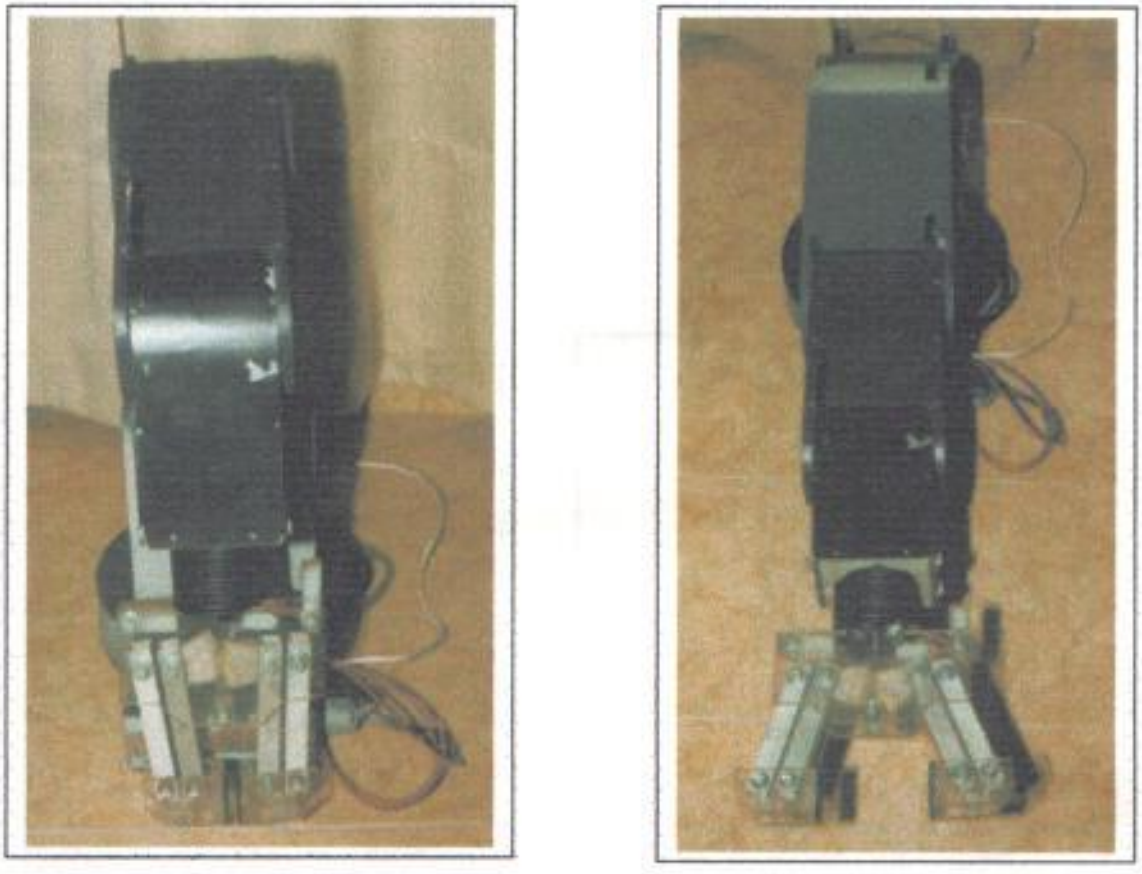


FIGURA 20 VISTA SUPERIOR DE LA POSICIÓN DE MÍNIMO ALCANCE Y VISTA DE LA MÁXIMA POSICIÓN

La transmisión de la tracción de los motores sobre las partes de mayor fuerza sobre el brazo se hacen desde la base de cada tramo. En ella se encuentran un motor con un engranaje reductor, que a su vez trabaja sobre un engranaje que esta firme a la parte superior del eje que queremos mover. El fin de ello es hacer que el peso del brazo no haga retroceder a los motores una vez que cese la corriente en ellos.

FIGURA 21 VISTA DE LOS MOTORES Y PIÑONES

FIGURA 22 VISTA DE LA TRANSMISIÓN

En caso de la pinza es un tema aparte, ya que fue hecha con un material acrílico y de aluminio de mayor consistencia, resistencia y dureza que el resto del brazo. La mano esta compuesta de partes bien marcadas, y a su vez preparada para poder rotar sin que ello afecte al resto de la movilidad del

brazo. La base sobre la que montamos la mano es una base de la siguiente forma:

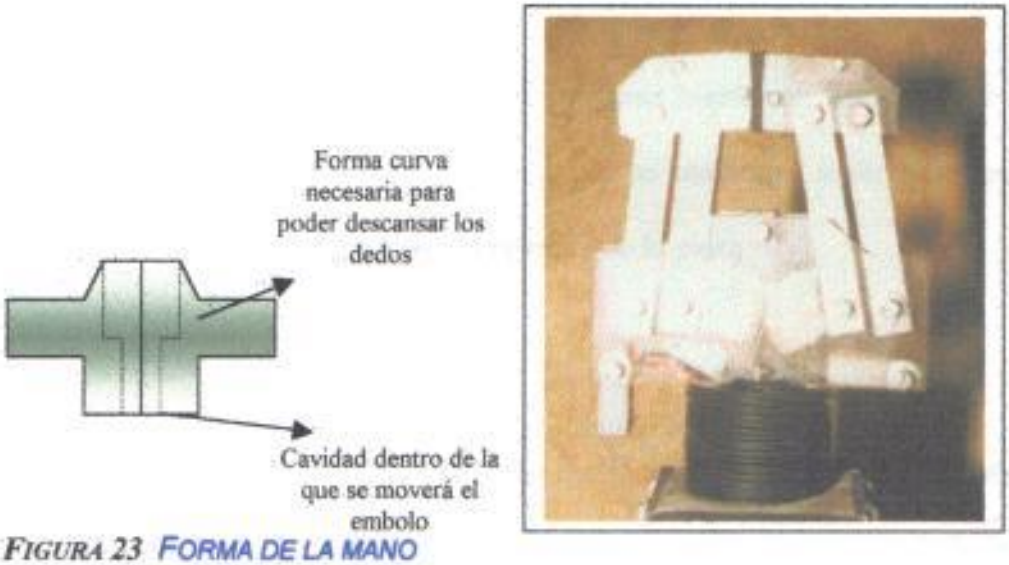
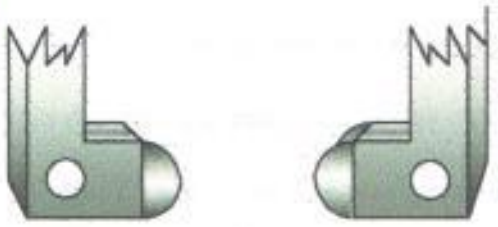


FIGURA 23 FORMA DE LA MANO

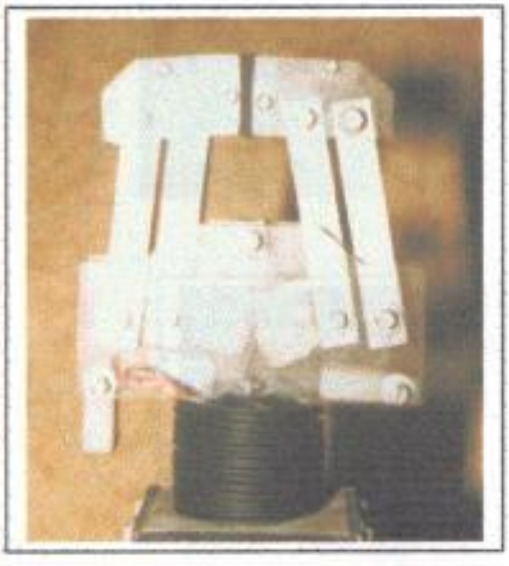
Dentro de esta base actúa un embolo que será el que al ser comprimido o dilatado logrará abrir y cerrar la mano. Para lograr eso, los dedos internos tenian que llevar la forma de una "L", como indicamos a continuación teniendo en cuenta de que haya el suficiente torque a fin de poder ejercer la fuerza necesaria sobre los dedos:

FIGURA 24 FORMA DE LOS DEDOS



Para que el agarre pueda ser perfectamente horizontal, diseñamos el sistema de dos dedos paralelos que hacen que se mantenga siempre las mordazas horizontales, adecuándose a la apertura de la pinza.

FIGURA 25 MUESTRA LA POSICIÓN DE LA PINZA CUANDO ESTÁ CERRADA Y ABIERTA



Para poder lograr la apertura de la pinza y cerrado de ella y que los dedos se muevan al mismo tiempo en sentido contrario he colocado en los extremos de la "L" de cada dedo un engranaje dentado que están entre ellos comunicados, a la vez hay una pieza en forma de "Z" que es la que une al pistón con los engranajes, este pistón es movido por medio de un motor que rota su eje los 360 grados. Al girar actúa sobre un corredizo unido al eje de hierro que hace que el movimiento en el pistón solo sea lineal, como lo indicamos ahora en el gráfico:

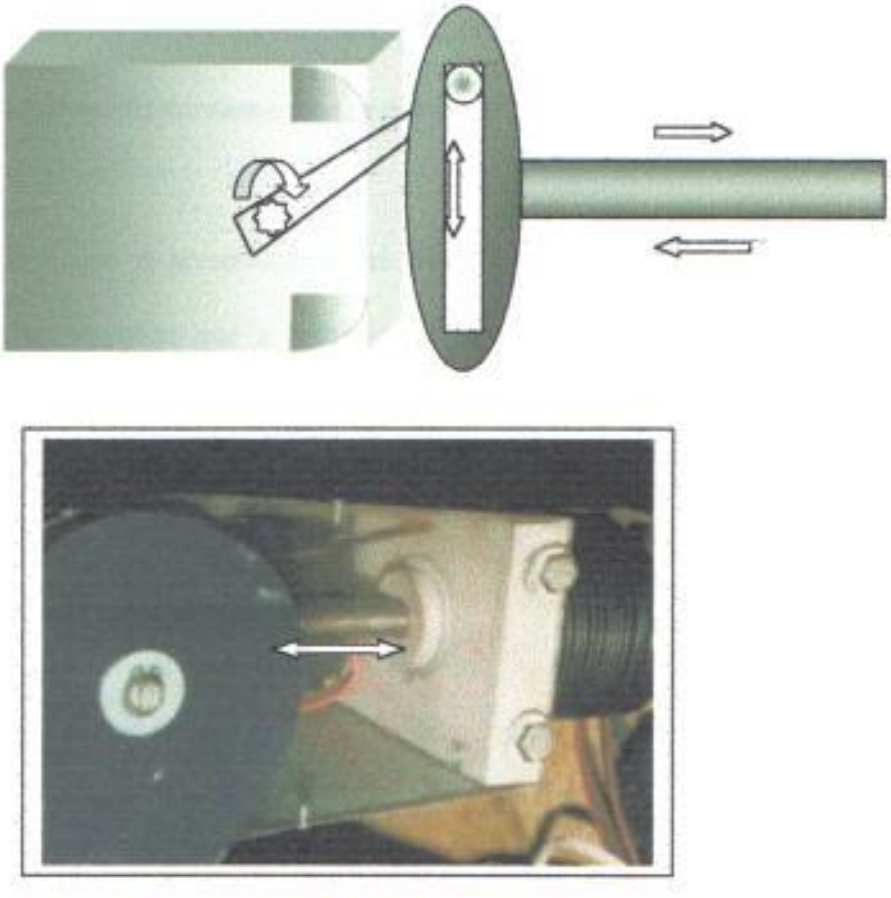


FIGURA 26 MECANISMO PARA ACCIONAR EL PISTÓN DE LA MANO

Toda la estructura del brazo está montada sobre una base giratoria, sobre la cual actuará un motor que está montado dentro una base metálica.

Para poder manejar los motores en el interior hemos adaptado un circuito que lleva relays (conocido como contactores bobinados) los que pueden dar la suficiente corriente a los motores, estos contactores trabajan a 24V en el bobinado primario y este conmuta un switch al cual está conectado una fuente de poder de 12V y hasta 5A .

El circuito le hemos hecho pistas lo suficientemente grandes como para poder soportar mucha corriente y no se queme.

En el anexo B presentaremos el diagrama esquemático del motor con más detalle. Mientras que a continuación presentamos el diseño de la placa, y luego el diagrama esquemático exportando la imagen.

FIGURA 27 PLACA DEL CIRCUITO

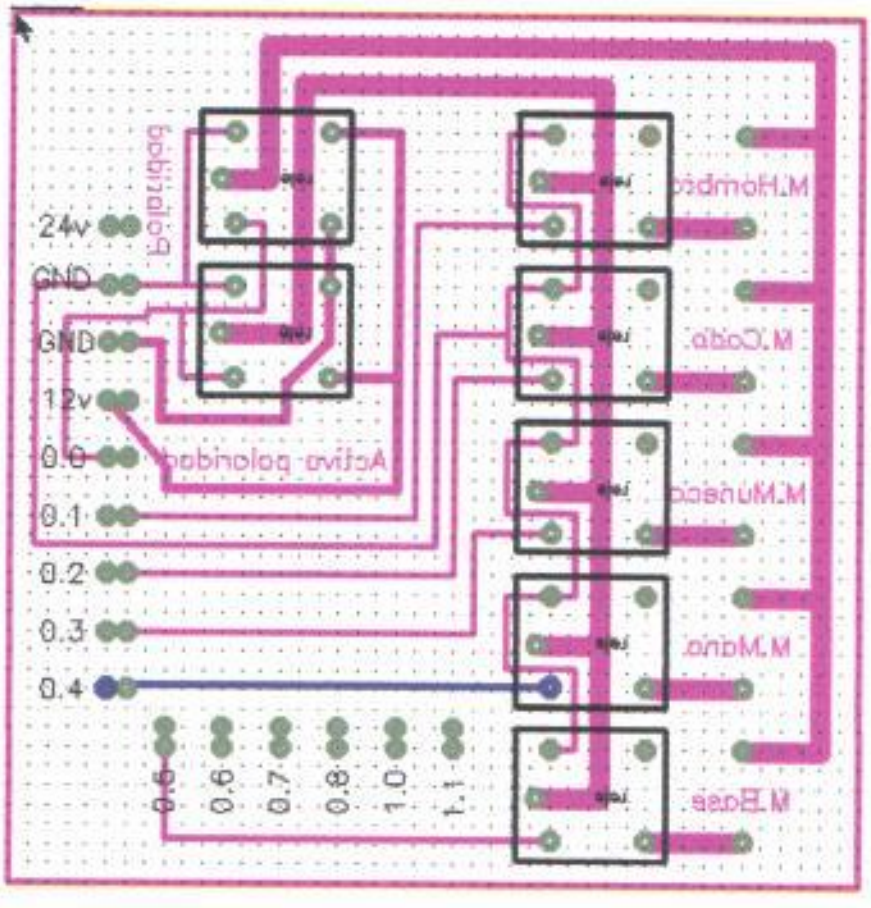
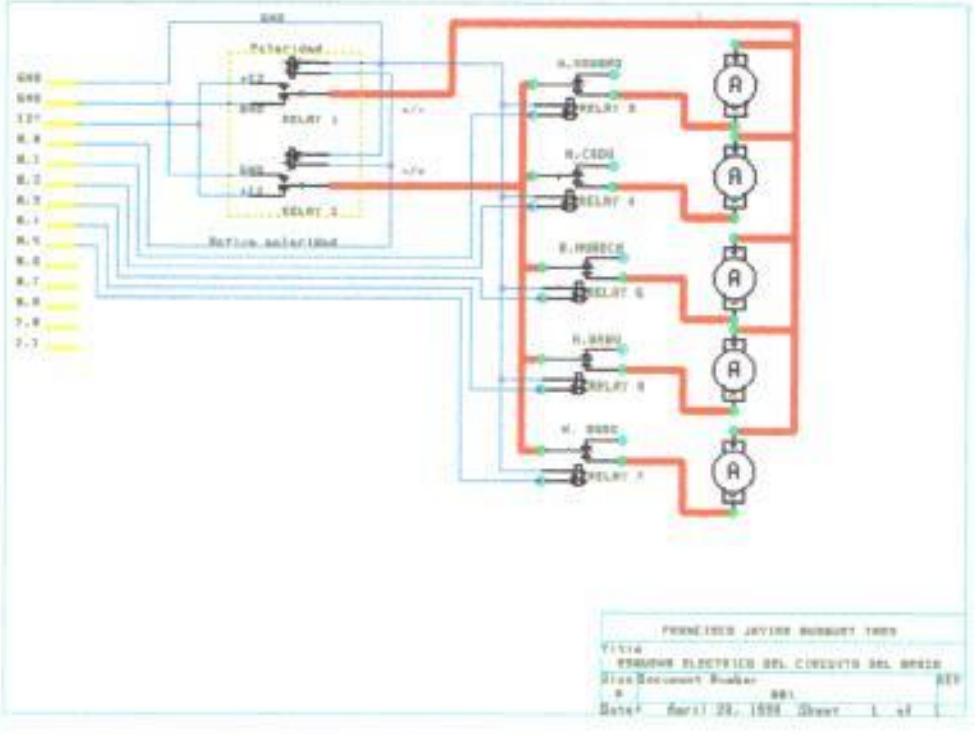


FIGURA 29 DIAGRAMA ESQUEMÁTICO DEL CIRCUITO (IMAGEN EXPORTADA DEL ORCAD)



Cave destacar que en esta placa se manejan dos tierras de diferentes fuentes, como ser una proveniente de la fuente de alimentación del autómata, que es de 24V a 1A, que es la encargada de conmutar a todos los relays, mientras que todos los voltajes de salida que ellos conmutan son de 12V a 5A, a pesar de que hay momentos que los motores exigen más corriente que la podemos suministrar

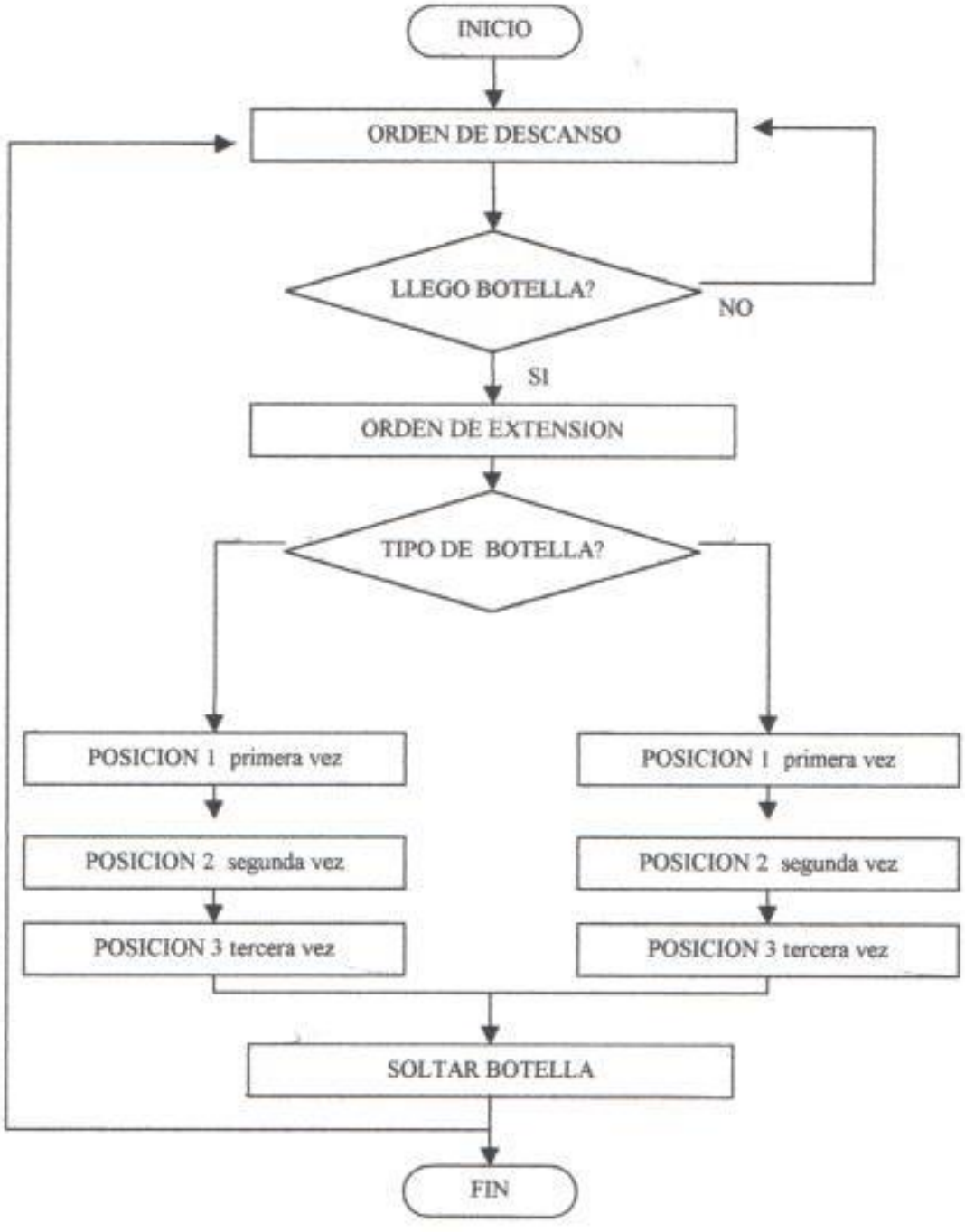
4.2. PROGRAMACION DEL BRAZO

Para la programación del brazo usaremos el lenguaje de programación del microprocesador CPU-214 de la Siemens, a su vez para una mayor comprensión usaremos unos diagramas de contactos, facilidad que nos entrega el software de programación . En la programación usaremos los siguientes elementos:

- Entradas I
- Salidas Q
- Temporizadores T
- Banderas internas M
- Banderas especiales SM
- Saltos JMP

La realización de la programación la haremos partiendo en pequeños módulos de funciones especificas de esa manera pueden ser invocados en diferentes momentos de la corrida del programa. Hay que tener en cuenta que ningún modulo ni principal, ni secundario no se puede usar dos veces la misma bobina de salida, pues el programa considera eso como un error.

4.2.1. DIAGRAMA GENERAL DE PROGRAMACIÓN



4.2.2. EXPLICACION GENERAL DEL PROGRAMA

El programa esta particionado de la siguiente manera:

- En la primer a línea de programación usamos una instrucción que incluye este CPU que es SM0.1 esta marca interna solo esta activa en el primer ciclo de corrida del programa, razón por la cual se usa para inicializar variables, temporizadores y demás variables. Como una vez cumplido el ciclo de colocación de una botella necesitamos las variables inicializadas usamos otra variable que es M0.6 que hace que ingrese el programa dentro de este bloque de iniciación de variables. Como el número de instrucciones por cada segmento es limitado, he tenido que hacer un artificio usando la memoria M10.0 a fin que si se están iniciando las variables del segmento 1, las del segmento 2 lo hacen en ese mismo instante.

- Creé en los siguientes segmentos de programa un espejo de las señales de entrada a fin de poderlas dejar encendidas a lo largo del programa, si se necesita, aun que ya no lo este fisicamente activa. Esto me dará una gran ventaja en la programación.

- De allí está la primera invocación a una subrutina que la SUB0 que es la de descanso. Esta subrutina lo que hace es dar corriente a los motores hasta que cada uno de ellos encuentre su final de carrera. La ventaja de esta iniciación de posición del brazo, es que, no le importa el tiempo, ni la posición que se encuentra para ubicarlo en descanso. Como el caso de la base es conflictivo ya que el la base cuento con tres microrruptores a saber: dos de final de carrera y uno de posición de medio. Aquí estaba el dilema ya que en caso de que el brazo este pasado el medio hacia la derecha abría que dar al motos de la base corriente directa hasta encontrar el medio y detenerse allí, pero si estaba en la mitad hacia la izquierda la corriente era inversa hasta encontrar el medio ¿cómo saber en que mitad está?. Para solucionar ese inconveniente sea la posición que sea de la base, el motor recibirá corriente directa hasta encontrar el tope derecho y desde allí, recibirá corriente inversa hasta encontrar la posición media. Como para girar la base hacia un sentido y luego hacia el otro se necesita invocar dos veces a un misma bobina de salida, tuve que anidar una subrutina dentro de esta que es la SUB7, la cual tiene por función retroceder la base hasta su extremo y de allí devolver el control a SUB0 para que la posicione en el medio. Como el corrimiento del programa es secuencial, es decir, si tiene un ciclo anidado él de igual forma ejecuta cualquier instrucción de programa que no este en ese ciclo si cumple sus condiciones. Por ello, ponemos un salto de protección JMP conforme no se ejecutarán esas instrucciones hasta que no se cumpla el ciclo.

Es de tener en cuenta que después de que termine cada una de las subrutinas se inician las variables que se usaron el ella, eso es con el objeto de que al ir al programa principal las variables puedan ser usadas en su estado original.

Todas las variables usadas en el programa, son variables globales, es decir, todas las rutinas las ven al mismo instante que el programa principal.

- Una vez colocado el brazo en su posición de descanso espera por la presencia de una botella. En el instante que llega una botella es invocada una rutina de despliegue del brazo hasta la botella SUB1 ella lo que hace es abrir la mano, y extender todo el brazo a su máximo alcance hasta posicionarse al nivel de la botella. De allí resetea variables y retorna el control al programa principal.

  Es de destacar que cada subrutina sea cual sea, tiene unas banderas de entrada y otras de salida, las de entrada son para cumplir con la corrida de esa parte del programa, y las de salida son para avisar que ese módulo ya está realizado.

- Una vez que el brazo esta desplegado sobre la botella llamamos a una rutina de coger botella y levantar el hombro con botella en mano SUB2. ¿ por qué no pudo estar esta instrucción en la rutina anterior de despliegue? Por que en

caso de estar allí habría que invocar por dos veces sucesivas la bobina de mano y la de hombro cosa que no está permitida por el programa.

En caso de haber dos llamados a una misma bobina el programa al ejecutarse simplemente ignora ambas instrucciones que las involucra.

- Cuando la botella se encuentra en el aire, se lee el tipo de botella que es a fin de saber a que lado hay que girar el brazo, hacia la derecha que son tipo B o la izquierda que son tipo A; y se invoca a la rutina de giro discriminado SUB3 .Esta subrutina se ejecuta solo si una de las cajas esta con algún lugar vacante, de lo contrario solo se ejecutara la rutina SUB9 que es la encargada de extraer las botellas del cargador y dejarlas a un lado a fin de esperar alguna de la otra clase.

- El brazo ahora lo tenemos sobre alguna de las cajas, ahora hay que posicionarlo en el lugar que le corresponde en la caja a esa botella, para ese efecto existen tres módulos SUB4, SUB6 y SUB8 que son posición 1,2 y 3 respectivamente. Estas solo se encargan de colocar al brazo en la posición de cada agujero de la caja retornando el control al programa principal.

- Para saber en cada momento la posición que corresponde al tipo A y B, ya que las botellas no llegan en orden preestablecido , hemos puesto en cada

uno de los módulos de posicionamiento de botellas, un grupo de memorias no inicilizadas al comienzo del programa principal, unas para el tipo A que solo cuando esta trabajando para el modo A puede ser cambiada y avisar que esa posición de botella ya esta ocupada; y otras que son para el tipo B que solo pueden variar si esta trabajando en modo B para avisar también que esa posición ya esta ocupada.

- Tenemos a la botella puesta sobre el hueco que le corresponde en la caja, ahora llamamos al módulo SUB5 que es el encargado de abrir la mano y dejar caer la botella, y para no dificultar el regreso del brazo lo levanta levemente a fin de pasar por sobre de la botella.

- Una vez cumplido este ciclo no hay más que hacer, de allí que la ruina 5 activa la memoria M0.6 que es la que resetea todas las variables y contadores y se inicia otra vez la posición de descanso.

- Como hay ocasiones en que se producen errores de alguna índole ya sea falseo de cualquier cosa, tenemos una rutina de protección que es la rutina SUB10 esta está hecha para que en caso que durante el proceso de ubicación de la botella llegara a encender algún tope de engranaje mediante los finales de carrera que contamos automáticamente el brazo pasa a un estado de parada.

4.2.3. DEACLARACION DE VARIABLES DELPROGRAMA

Nombre	Comentario
M0.0	INICIA LA RUTINA 0
M0.1	INICIA LA RUTINA 1
M0.2	INICIA LA RUTINA 2
M0.3	INICIA LA RUTINA 3
M0.4	INICIA LA RUTINA 4
M0.5	INICIA LA RUTINA 5
M0.6	INICIALIZA LAS VARIABLES
M0.7	INICIA LA RUTINA 7
M1.0	INICIA LA RUTINA 4 TIPO A
M1.1	INICIA LA RUTINA 6 TIPO A
M1.2	INICIA LA RUTINA 8 TIPO A
M1.3	INICIA RUTINA DE RECHAZO SUB9
M4.1	ES EL REFLEJO DE LA ENTRADA I0.1
M4.2	ES EL REFLEJO DE LA ENTRADA I0.3
M4.3	ES EL REFLEJO DE LA ENTRADA I0.0
M5.0	ES EL REFLEJO DE LA MEMORIA Q0.0
M5.1	ES EL REFLEJO DE LA MEMORIA Q0.1
M5.2	ES EL REFLEJO DE LA MEMORIA Q0.2
M5.3	ES EL REFLEJO DE LA MEMORIA Q0.3
M5.4	ES EL REFLEJO DE LA MEMORIA Q0.4
M5.5	ES EL REFLEJO DE LA MEMORIA Q0.5
M6.0	INICIA RUTINA 4 TIPO B
M6.I	INICIA RUTINA 6 TIPO B
M6.2	INICIA RUTINA 8 TIPO B
M6.3	INICIA RUTINA DE RECHAZO SUB9
M7.7	AUXILIAR
M10.0	MEMORIA DE ARTIFICIO PARA CONTINUAR SEGMENTO 1 EN 2
SM0.0	MARCA ESPECIAL: SIEMPRE ACTIVA
SM0.1	MARCA ESPECIAL ACTIVA EN EL 1er CICLO DE EJECUCION
I0.0	ENTRADA (TIPO DE BOTELLA)
I0.1	ENTRADA (PRESENCIA DE BOTELLA)
I0.2	ENTRADA (POSICION MDEIA DE LA BASE)

I0.3	ENTRADA (LIMITES DE LA BASE)
I0.4	ENTRADA (LIMITES DEL HOMBRO)
I0.5	ENTRADA (LIMITES DEL CODO)
I0.6	ENTRADA (LIMITE DE MUÑECA)
I0.7	ENTRADA (LIMITE DE MANO CERRADA)
Q0.0	SALIDA (POLARIDAD)
Q0.1	SALIDA (MOTOR DE LA BASE)
Q0.2	SALIDA (MOTOR DE HOMBRO)
Q0.3	SALIDA (MOTOR DE CODO)
Q0.4	SALIDA (MOTOR DE MUÑECA)
Q0.5	SALIDA (MOTOR DE MANO)
T33-T36	TEMPORIZADORES CON BASE DE TIEMPO 10ms
T37-T52	TEMPORIZADORES CON BASE DE TIEMPO 100ms
SUB_0	POSICION DE DESCANSO DEL BRAZO
SUB_1	DESPLIEGUE DEL BRAZO CON BOTELLA PRESENTE
SUB_2	COGE LA BOTELLA Y LA LEVANTA
SUB_3	GIRA AL BRAZO HACIA POSICION A o B
SUB_4	POSICIONA A LA BOTELLA EN LUGAR 1 DE LA CAJA
SUB_5	ABRE LA MANO Y LEVANTA EL HOMBRO
SUB_6	POSICIONA A LA BOTELLA EN LUGAR 2 DE LA CAJA
SUB_7	GIRA LA BASE HASTA SU LIMITE DERECHO
SUB_8	POSICIONA A LA BOTELLA EN LUGAR 3 DE LA CAJA
SUB_9	BAJA LAS BOTELLAS AL LADO, CAJAS LLENAS
SUB_10	CONTROL DE ERRORES
LBL	MARCA DE SALTO

4.2.4. PROGRAMA DEL BRAZO EN AWL

```
//ESTE PROGRAMA MANEJA UN BRAZO MECANICO CON MOTORES DC DE
12V SU FUNCION ES COGER
//BOTELLAS Y CLASIFICARLAS PONIENDOLAS EN UNA CAJA EN SUS
DISTINTAS POSICIONES HASTA OCUPARLAS TODAS

NETWORK 1 //PROGRAMA PRINCIPAL

LD   SM0.1
O    M0.6
S    M0.0, 1
R    M0.1, 1
R    M0.2, 1
R    M0.3, 1
R    M0.4, 1
R    M0.5, 1
R    M0.6, 1
R    M0.7, 1
R    M5.0, 1
```

```
R   M5.1, 1
R   M5.2, 1
R   M5.3, 1
R   M5.4, 1
R   M5.5, 1
R   M4.1, 1
R   M4.2, 1
R   M7.7, 1
R   T33, 1
R   T34, 1
R   T35, 1
R   T36, 1
R   T37, 1
R   T38, 1
R   T39, 1
R   T40, 1
R   T41, 1
R   T42, 1
R   T43, 1
R   T44, 1
R   T45, 1
R   T46, 1
```

```
S    M10.0, 1

NETWORK 2
LD   M10.0
R    T47, 1
R    T48, 1
R    T49, 1
R    T50, 1
R    M10.0, 1
R    T51, 1
R    T52, 1

NETWORK 3 //EL MICRORRUPTOR DE PRESENCIA DE BOTELLA USA MEMORIA PARA MANIPULAR MEJOR
LD   I0.1
=    M4.1

NETWORK 4 //EL FINAL DE CARRERA DE LA BASE USA UNA MEMORIA COMO REFLEJO PARA MANIPULARLA MEJOR
LD   I0.3
=    M4.2
```

```
NETWORK 5 //EL FINAL DE CARRERA DE LA BASE USA UNA MEMORIA
COMO REFLEJO PARA MANIPULARLA MEJOR
LD   I0.0
S    M4.3, 1

NETWORK 6 //CAMBIO DE CAJAS
LD   I1.3
R    M6.3, 1
R    M1.3, 1
S    M6.0, 1
S    M1.0, 1
R    M1.1, 1
R    M1.2, 1
R    M6.1, 1
R    M6.2, 1

NETWORK 7 //LLAMA A RUTINA DE DESCANSO
LD   M0.0
CALL 0

NETWORK 8 //LLAMA A RUTINA DE DESPLIEGUE
LD   M0.1
```

```
A    M4.1
CALL  1

NETWORK 9 //TOMA LA BOTELLA Y LA LEVANTA
LD   M0.2
AN   M0.1
AN   M0.3
AN   M0.4
CALL  2

NETWORK 10        //RECHAZA LA BOTELLA DEL TIPO A
LD   M0.3
AN   M4.3
A    M1.3
CALL  9

NETWORK 11        //RECHAZA LAS BOTELLAS DEL TIPO B
LD   M0.3
A    M4.3
A    M6.3
CALL  9
```

```
NETWORK 12        //GIRO DEL BRAZO HACIA A  O  B
LD    M0.3
LDN   M6.3
ON    M1.3
ALD
AN    M0.1
AN    M0.2
AN    M0.4
CALL  3

NETWORK 13        //POSICIONA LA BOTELLA EN LUGAR 1 TIPO A
LDN   M4.3
A     M0.4
A     M1.0
AN    M1.1
AN    M1.2
AN    M0.1
AN    M0.2
AN    M0.3
AN    M0.5
CALL  4
```

```
NETWORK 14      //POSICIONA LA BOTELLA EN LUGAR 1 TIPO B
LD   M4.3
A    M0.4
A    M6.0
AN   M6.1
AN   M6.2
AN   M0.1
AN   M0.2
AN   M0.3
AN   M0.5
CALL  4

NETWORK 15      //POSICIONA LA BOTELLA EN LUGAR 2 TIPO A
LDN  M4.3
A    M0.4
A    M1.1
AN   M1.2
AN   M0.1
AN   M0.2
AN   M0.3
AN   M0.5
CALL  6
```

```
NETWORK 16      //POSICIONA LA BOTELLA EN LUGAR 2 TIPO B
LD   M4.3
A    M0.4
A    M6.1
AN   M6.2
AN   M0.1
AN   M0.2
AN   M0.3
AN   M0.5
CALL 6

NETWORK 17      //POSICIONA LA BOTELLA EN LUGAR 3 TIPO A
LDN  M4.3
A    M0.4
A    M1.2
AN   M1.1
AN   M0.1
AN   M0.2
AN   M0.3
AN   M0.5
CALL 8
```

```
NETWORK 18      //POSICIONA LA BOTELLA EN LUGAR 3 TIPO B
LD   M4.3
A    M0.4
A    M6.2
AN   M6.1
AN   M0.1
AN   M0.2
AN   M0.3
AN   M0.5
CALL 8

NETWORK 19      //ABRE LA MANO PARA QUE CAIGA BOTELLA Y
LEVANTA LIGERAMENTE EL HOMBRO
LD   M0.5
AN   M0.1
AN   M0.2
AN   M0.3
AN   M0.4
AN   M0.6
CALL 5
```

```
NETWORK 20      //A CADA MEMORIA LE CORRESPONDE UNA BOBINA
DE SALIDA
LD   SM0.0
LPS
A    M5.0
=    Q0.0
LRD
A    M5.1
=    Q0.1
LRD
A    M5.2
=    Q0.2
LRD
A    M5.3
=    Q0.3
LRD
A    M5.4
=    Q0.4
LPP
A    M5.5
=    Q0.5
```

```
NETWORK 21

MEND

NETWORK 22      //RUTINA DE DESCANSO

SBR  0

NETWORK 23      //CIERRA LA MANO HASTA SU LIMITE

LD   M0.0

AN   I0.7

=    M5.5

NETWORK 24      //GIRA LA BASE HASTA SU EXTREMO DERECHO

LDN  M0.7

A    M0.0

CALL 7

NETWORK 25      //SI LA BASE NO LLEGO AL EXTREMO DERECHO NO SE
EJECUTE NADA DE LO SIGUIENTE

LDN  M0.7

JMP  0
```

```
NETWORK 26      //ACTIVO POLARIDAD INVERSA
LDN   M5.5
A     I0.7
O     M5.0
=     M5.0

NETWORK 27      //RETRAIGO HOMBRO, BASE AL MEDIO, CODO,
MUÑECA
LD    M0.0
A     M5.0
AN    M5.5
LPS
AN    I0.4
=     M5.2
LRD
AN    M5.2
AN    I0.2
=     M5.1
LRD
AN    M5.1
AN    M5.2
AN    I0.5
```

= M5.3

LPP

AN M5.1

AN M5.2

AN M5.3

AN I0.6

= M5.4

NETWORK 28 //RESETEO MEMORIAS USADAS AQUI Y SOLO DEJO ACTIVA LA MEMORIA DE SALIDA DE SUBRUTINA

LD I0.4

A I0.5

A I0.6

A I0.7

R M0.0, 1

S M0.1, 1

R M5.0, 1

NETWORK 29 //META DEFINIDA DEL SALTO

LBL 0

```
NETWORK 30      //RETORNO AL PROGRAMA QUE LO LLAMO
RET

NETWORK 31      //*******  DESPLIEGA EL BRAZO HASTA LA BOTELLA
**********
SBR  1

NETWORK 32      //HABRE LA MANO BOTELLA PRESENTE
LD   M0.1
LPS
AN   T33
=    M5.0
=    M5.5
LPP
TON  T33, +415

NETWORK 33      //DESPLIEGUA MUÑECA BOTELLA PRESENTE
LD   M0.1
AN   M5.5
LPS
AN   T34
=    M5.4
```

```
LPP
TON   T34, +190

NETWORK 34        //DESPLIEGE DEL CODO BOTELLA PRESENTE
LD    M0.1
AN    M5.5
AN    M5.4
LPS
AN    T35
=     M5.3
LPP
TON   T35, +315

NETWORK 35        //DESPLIEGUE DEL HOMBRO BOTELLA PRESENTE
LD    M0.1
AN    M5.5
AN    M5.4
AN    M5.3
LPS
AN    T36
=     M5.2
LPP
```

```
TON   T36, +327

NETWORK 36      //RESETEO DE LO USADO Y SETEO DE LA MEMORIA
DE SALIDA
LD    M0.1
A     T36
R     M0.1, 1
S     M0.2, 1

NETWORK 37      //RETORNO AL PROGRAMA QUE LLAMO
RET

NETWORK 38      //TOMA LA BOTELLA Y LA LEVANTA
SBR   2

NETWORK 39      //CONTROL DE ERRORES
LD    M0.2
CALL  10

NETWORK 40      //CIERRA LA MANO PARA ATRAPAR BOTELLA
LD    M0.2
LPS
```

```
AN   T37
=    M5.5
LPP
TON  T37, +32

NETWORK 41      //ACTIVA LA POLARIDAD INVERSA
LD   M0.2
AN   M5.5
AN   T38
=    M5.0

NETWORK 42      //LEVANTA EL HOMBRO CON BOTELLA EN MANO
LD   M0.2
AN   M5.5
AN   M5.1
AN   M5.3
LPS
AN   T38
=    M5.2
LPP
TON  T38, +15
```

```
NETWORK 43      //SETEO DE MEMORIA DE SALIDA Y RESETEO DE LAS
DEMAS
LD   M0.2
A    T38
S    M0.3, 1
R    M0.2, 1
R    M5.2, 1
R    M5.0, 1

NETWORK 44      //RETORNO
RET

NETWORK 45      //********* POSICION 1 DE GIRO TIPO A ***********
SBR  3

NETWORK 46      //CONTROL DE ERRORES
LD   M0.3
CALL 10

NETWORK 47      //ACTIVA POLARIDAD INVERSA
LD   M0.3
AN   M1.3
```

```
AN   M4.3
AN   T40
=    M5.0
```

NETWORK 48 //ACTIVA GIRO HACIA LA IZQUIERDA O DERECHA DEPENDIENDO EL TIPO DE BOTELLA

```
LD   M0.3
LPS
AN   T40
LD   M5.0
ON   M6.3
ALD
=    M5.1
LPP
TON  T40, +25
```

NETWORK 49 //RESETEA M(0.3) Y SETEA M(0.4) Y ASEGURA QUE LA POLARIDAD INVERSA NO QUEDE ACTIVA

```
LD   T40
S    M0.4, 1
R    M0.3, 1
R    M5.0, 1
```

```
R     M5.1, 1

NETWORK 50        //RETORNO AL PROGRAMA QUE LLAMO
RET

NETWORK 51        //*********  BAJA HOMBRO A LA POSICION 1 DE LA
CAJA **********
SBR   4

NETWORK 52        // CONTROL DE  ERRORES
LD    M0.4
CALL  10

NETWORK 53        //BAJA EL HOMBRO CON BOTELLA EN MANO
LD    M0.4
LPS
AN    T41
=     M5.2
LPP
TON   T41, +8
```

```
NETWORK 54      //SETEO DE MEMORIA DE SALIDA Y RESETEO DE LAS
DEMAS
LD   M0.4
AN   M4.3
A    T41
R    M0.4, 1
S    M0.5, 1
R    M5.2, 1
R    M5.0, 1
S    M1.1, 1
R    M1.0, 1

NETWORK 55      //SETEO DE MEMORIA DE SALIDA Y RESETEO DE LAS
DEMAS
LD   M0.4
A    M4.3
A    T41
R    M0.4, 1
S    M0.5, 1
R    M5.2, 1
R    M5.0, 1
S    M6.1, 1
```

```
R    M6.0, 1

NETWORK 56      //RETORNO
RET

NETWORK 57      //************** SUELTA LA BOTELLA DESDE LA MANO
**************
SBR  5

NETWORK 58      //CONTROL DE ERRORES
LD   M0.5
CALL  10

NETWORK 59      //ACTIVA POLARIDAD INVERSA
LD   M0.5
=    M5.0

NETWORK 60      //ABRE LA MANO Y SUELTA BOTELLA
LD   M0.5
A    M5.0
LPS
AN   T42
```

= M5.5

LPP

TON T42, +32

NETWORK 61 //LEVANTA EL HOMBRO UN POCO A FIN DE NO MOLESTAR EN NADA AL REGRESO DEL BRAZO A SU ORIGEN

LD M0.5

A M5.0

AN M5.5

LPS

AN T43

= M5.2

LPP

TON T43, +4

NETWORK 62 //SETEO DE MEMORIA DE SALIDA Y RESETEO DE LAS DEMAS

LD T43

R M0.5, 1

S M0.6, 1

R M5.5, 1

R M5.0, 1

```
R     M4.3, 1

NETWORK 63        //RETORNO
RET

NETWORK 64        //POSICION 2 DE BOTELLAS
SBR   6

NETWORK 65        //CONTROL DE ERRORES
LD    M0.4
CALL  10

NETWORK 66        //LEVANTA MUÑECA
LD    M0.4
LPS
AN    T44
=     M5.4
LPP
TON   T44, +6

NETWORK 67        //ACTIVA POLARIDAD INVERSA
LD    M0.4
```

```
A    T44
=    M5.0

NETWORK 68        //LEVANTA MAS EL HOMBRO
LD   M0.4
A    M5.0
LPS
AN   T45
=    M5.2
LPP
TON  T45, +10

NETWORK 69        //BAJA EL CODO HACIA CAJA
LD   M0.4
A    M5.0
AN   M5.2
LPS
AN   T46
=    M5.3
LPP
TON  T46, +20
```

```
NETWORK 70        //RESETEO DE LO USADO
LD   M0.4
AN   M4.3
A    T46
R    M0.4, 1
R    M5.0, 1
R    M5.2, 1
R    M5.3, 1
R    M5.4, 1
R    M1.1, 1
S    M1.2, 1
S    M0.5, 1

NETWORK 71        //RESETEO DE LO USADO
LD   M0.4
A    M4.3
A    T46
R    M0.4, 1
R    M5.0, 1
R    M5.2, 1
R    M5.3, 1
R    M5.4, 1
```

```
R    M6.1, 1
S    M6.2, 1
S    M0.5, 1

NETWORK 72
RET

NETWORK 73    //*********** MANDA LA BASE HASTA SU LIMITE Y
RETORNA EL CONTROL A SUB0 ***********
SBR  7

NETWORK 74    //A MENOS QUE ESTE ESTACIONADO EN EL MEDIO LA
BASE LA MOVERA HASTA SU EXTREMO DERECHO
LD   M4.2
A    I0.3
LD   I0.2
AN   M5.1
OLD
S    M0.7, 1
R    M5.1, 1
```

```
NETWORK 75      //MOVER BASE A SU EXTREMO DERECHO
LDN   M0.7
AN    M4.2
AN    I0.3
=     M5.1

NETWORK 76      //RETORNO
RET

NETWORK 77      //POSICION 3 DE LA CAJA DE BOTELLAS
SBR   8

NETWORK 78      //ACTIVA POLARIDAD INVERSA
LD    M0.4
=     M5.0

NETWORK 79      //BAJA MUÑECA
LD    M0.4
A     M5.0
LPS
AN    T48
=     M5.4
```

```
LPP
TON   T48, +16

NETWORK 80        //LEVANTA MAS EL HOMBRO
LD    M0.4
A     M5.0
A     M5.4
LPS
AN    T49
=     M5.2
LPP
TON   T49, +18

NETWORK 81        //BAJA EL CODO
LD    M0.4
A     M5.0
AN    M5.2
AN    M5.4
LPS
AN    T50
=     M5.3
LPP
```

```
TON   T50, +12

NETWORK 82      //RESETEO DE LAS MEMORIAS USADAS Y SETEO DE
LAS DE SALIDA
LD    M0.4
AN    M4.3
A     T50
S     M0.5, 1
R     M0.4, 1
R     M5.0, 1
R     M5.2, 1
R     M5.3, 1
R     M5.4, 1
R     M1.2, 1
S     M1.3, 1

NETWORK 83      //RESETEO DE LAS MEMORIAS USADAS Y SETEO DE
LAS DE SALIDA
LD    M0.4
A     M4.3
A     T50
S     M0.5, 1
```

```
R    M0.4, 1
R    M5.0, 1
R    M5.2, 1
R    M5.3, 1
R    M5.4, 1
R    M6.2, 1
S    M6.3, 1

NETWORK 84        //RETORNO
RET

NETWORK 85
SBR  9

NETWORK 86
LD   M0.3
CALL 10

NETWORK 87
LD   M0.3
R    M5.0, 1
```

NETWORK 88

```
LD   M0.3
LPS
AN   T51
=    M5.1
LRD
TON  T51, +10
LRD
AN   T52
=    M5.2
LPP
TON  T52, +10
```

NETWORK 89

```
LD   T52
R    M0.3, 1
S    M0.5, 1
R    M5.1, 1
R    M5.2, 1
```

NETWORK 90

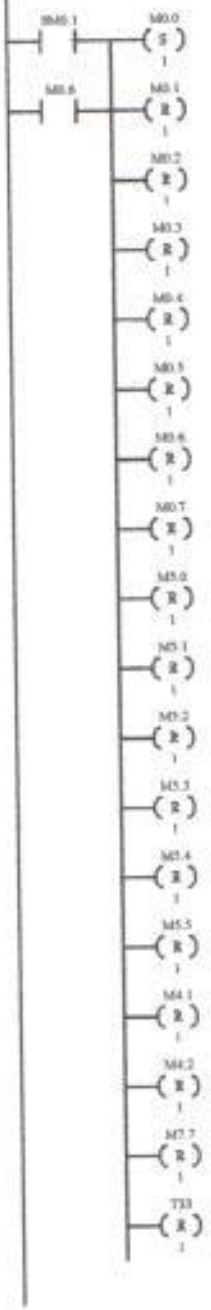
```
RET
```

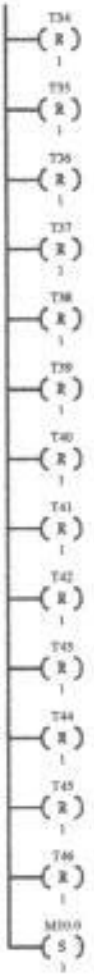
```
NETWORK 91        //SUBRUTINA DE FALLOS DE FUNCIONAMIENTO
SBR  10

NETWORK 92
LD   I0.3
O    I0.4
O    I0.5
O    I0.6
O    I0.7
O    M4.2
STOP

NETWORK 93
RET
```

4.2.5. PROGRAMA EN KOP

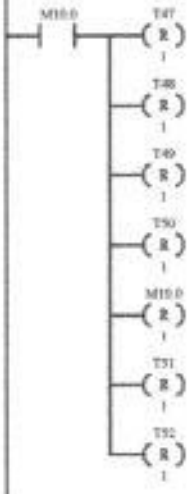
Este sistema es muy usado por su facilidad en comprensión ya que es similar a un diagrama de contactos.

Segmento 1 PROGRAMA PRINCIPAL

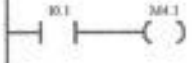



T34
R
1
T35
R
1
T36
R
1
T37
R
1
T38
R
1
T39
R
1
T40
R
1
T41
R
1
T42
R
1
T43
R
1
T44
R
1
T45
R
1
T46
R
1
M10.0
S
1

Segmento 2

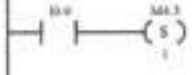


Segmento 3 EL MICRORRUPTOR DE PRESENCIA DE BOTELLA USA MEMORIA PARA MANIPULAR MEJOR

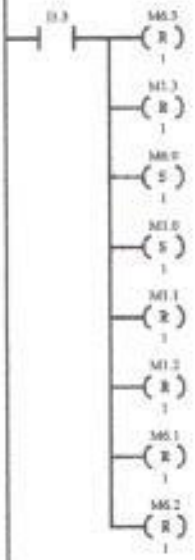


Segmento 4 EL FINAL DE CARRERA DE LA BASE USA UNA MEMORIA COMO REFLEJO PARA MANIPULARLA MEJOR

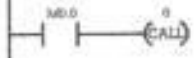
Segmento 5 EL FINAL DE CARRERA DE LA BASE USA UNA MEMORIA COMO REFLEJO PARA MANIPULARLA MEJOR



Segmento 6 CAMBIO DE CAJAS

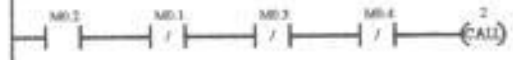


Segmento 7 LLAMA A RUTINA DE DESCANSO



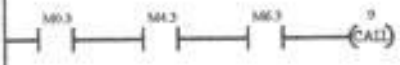
Segmento 8 LLAMA A RUTINA DE DESPLIEGUE

Segmento 9 TOMA LA BOTELLA Y LA LEVANTA

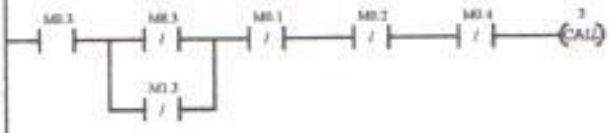


Segmento 10 RECHAZA LA BOTELLA DEL TIPO A

Segmento 11 RECHAZA LAS BOTELLAS DEL TIPO B

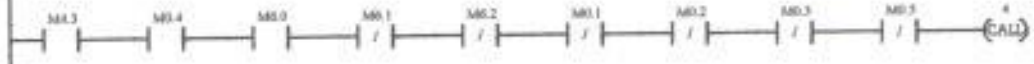


Segmento 12 GIRO DEL BRAZO HACIA A O B

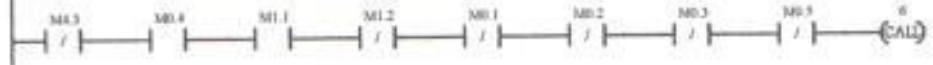


Segmento 13 POSICIONA LA BOTELLA EN LUGAR 1 TIPO A

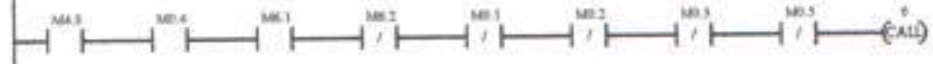
Segmento 14 POSICIONA LA BOTELLA EN LUGAR 1 TIPO B



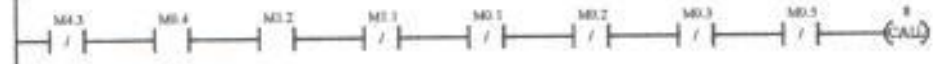
Segmento 15 POSICIONA LA BOTELLA EN LUGAR 2 TIPO A



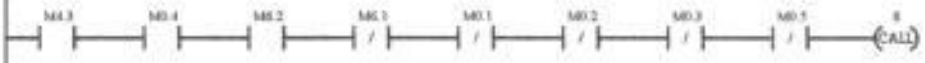
Segmento 16 POSICIONA LA BOTELLA EN LUGAR 2 TIPO B



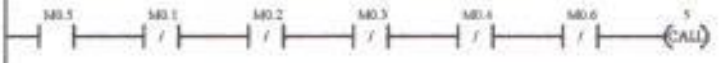
Segmento 17 POSICIONA LA BOTELLA EN LUGAR 3 TIPO A



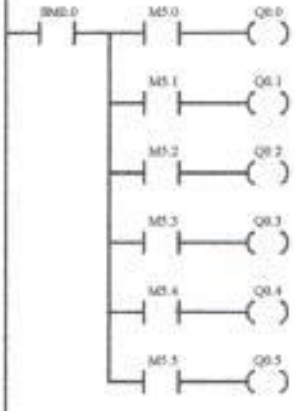
Segmento 18 POSICIONA LA BOTELLA EN LUGAR 3 TIPO B



Segmento 19 ABRE LA MANO PARA QUE CAIGA BOTELLA Y LEVANTA LIGERAMENTE EL HOMBRO



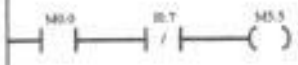
Segmento 20 A CADA MEMORIA LE CORRESPONDE UNA BOBINA DE SALIDA



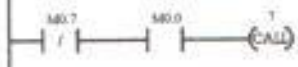
Segmento 21

Segmento 22 RUTINA DE DESCANSO

Segmento 23 CIERRA LA MANO HASTA SU LIMITE

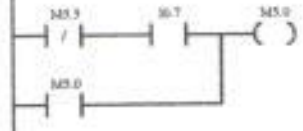


Segmento 24 GIRA LA BASE HASTA SU EXTREMO DERECHO

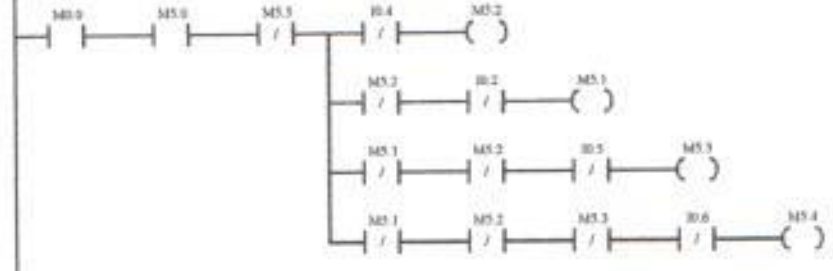


Segmento 25 SI LA BASE NO LLEGO AL EXTREMO DERECHO NO SE EJECUTE NADA DE LO SIGUIENTE

Segmento 26 ACTIVO POLARIDAD INVERSA



Segmento 27 RETRAIGO HOMBRO, BASE AL MEDIO, CODO, MUÑECA



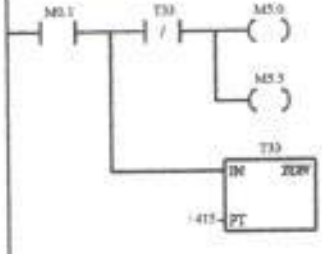
Segmento 28 RESETEO MEMORIAS USADAS AQUI Y SOLO DEJO ACTIVA LA MEMORIA DE SALIDA DE SUBRUTINA

Segmento 29 META DEFINIDA DEL SALTO

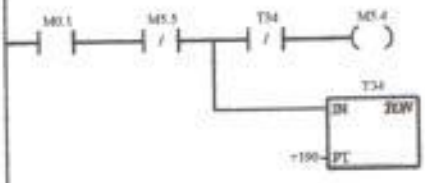
Segmento 30 RETORNO AL PROGRAMA QUE LO LLAMO

Segmento 31 ******* DESPLIEGA EL BRAZO HASTA LA BOTELLA **********

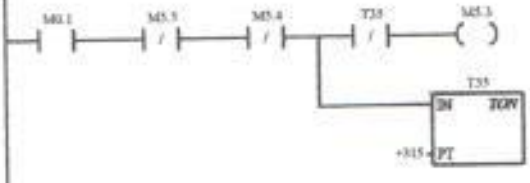
Segmento 32 HABRE LA MANO BOTELLA PRESENTE



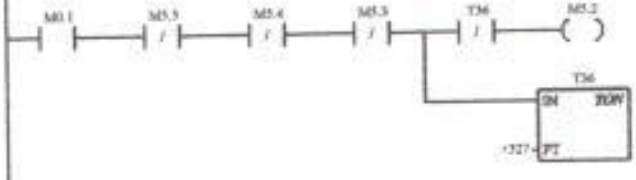
Segmento 33 DESPLIEGUA MUÑECA BOTELLA PRESENTE



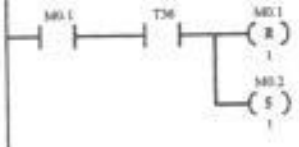
Segmento 34 DESPLIEGE DEL CODO BOTELLA PRESENTE



Segmento 35 DESPLIEGUE DEL HOMBRO BOTELLA PRESENTE



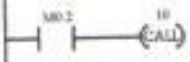
Segmento 36 RESETEO DE LO USADO Y SETEO DE LA MEMORIA DE SALIDA



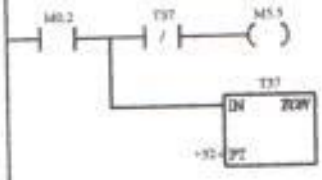
Segmento 37 RETORNO AL PROGRAMA QUE LLAMO

Segmento 38 TOMA LA BOTELLA Y LA LEVANTA

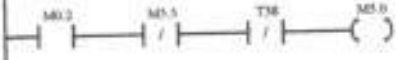
Segmento 39 CONTROL DE ERRORES



Segmento 40 CIERRA LA MANO PARA ATRAPAR BOTELLA

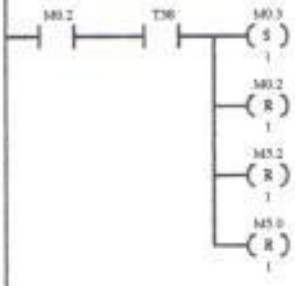


Segmento 41 ACTIVA LA POLARIDAD INVERSA



Segmento 42 LEVANTA EL HOMBRO CON BOTELLA EN MANO

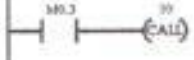
Segmento 43 SETEO DE MEMORIA DE SALIDA Y RESETEO DE LAS DEMAS



Segmento 44 RETORNO

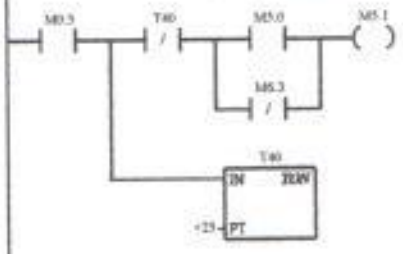
Segmento 45 ********* POSICION 1 DE GIRO TIPO A **********

Segmento 46 CONTROL DE ERRORES

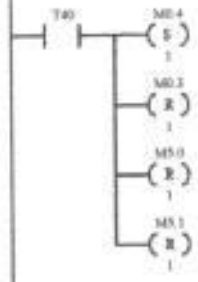


Segmento 47 ACTIVA POLARIDAD INVERSA

Segmento 48 ACTIVA GIRO HACIA LA IZQUIERDA O DERECHA DEPENDIENDO EL TIPO DE BOTELLA



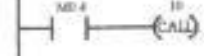
Segmento 49 RESETEA M(0.3) Y SETEA M(0.4) Y ASEGURA QUE LA POLARIDAD INVERSA NO QUEDE ACTIVA



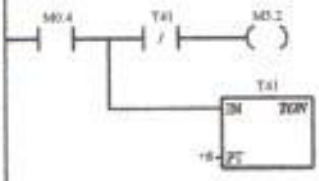
Segmento 50 RETORNO AL PROGRAMA QUE LLAMO

Segmento 51 ********** BAJA HOMBRO A LA POSICION 1 DE LA CAJA **********

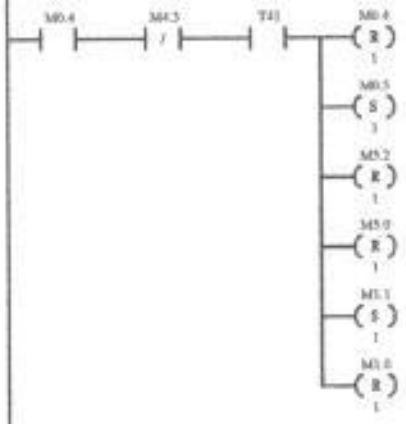
Segmento 52 CONTROL DE ERRORES



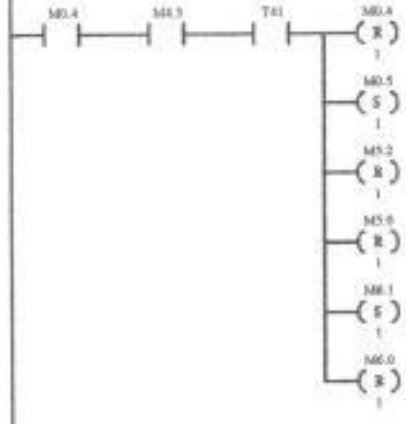
Segmento 53 BAJA EL HOMBRO CON BOTELLA EN MANO



Segmento 54 SETEO DE MEMORIA DE SALIDA Y RESETEO DE LAS DEMAS



Segmento 55 SETEO DE MEMORIA DE SALIDA Y RESETEO DE LAS DEMAS

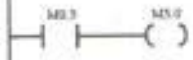


Segmento 56 RETORNO

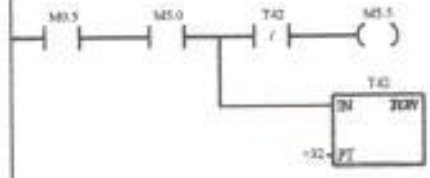
Segmento 57 ********** SUELTA LA BOTELLA DESDE LA MANO **********

Segmento 58 CONTROL DE ERRORES

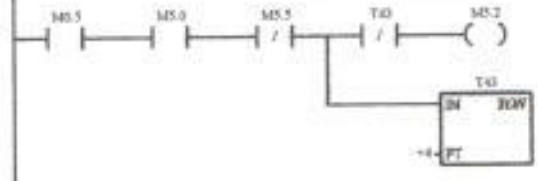
Segmento 59 ACTIVA POLARIDAD INVERSA



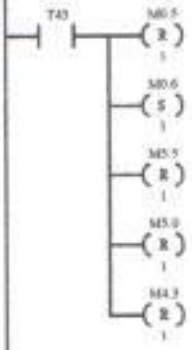
Segmento 60 ABRE LA MANO Y SUELTA BOTELLA



Segmento 61 LEVANTA EL HOMBRO UN POCO A FIN DE NO MOLESTAR EN NADA AL REGRESO DEL BRAZO A SU ORIGEN



Segmento 62 SETEO DE MEMORIA DE SALIDA Y RESETEO DE LAS DEMAS

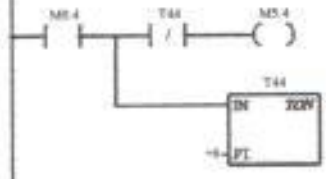


Segmento 63 RETORNO

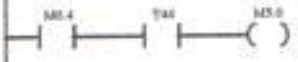
Segmento 64 POSICION 2 DE BOTELLAS

Segmento 65 CONTROL DE ERRORES

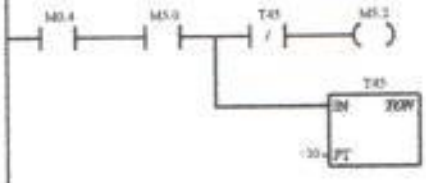
Segmento 66 LEVANTA MUÑECA



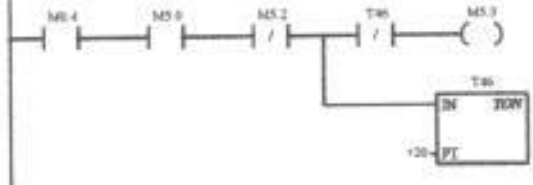
Segmento 67 ACTIVA POLARIDAD INVERSA



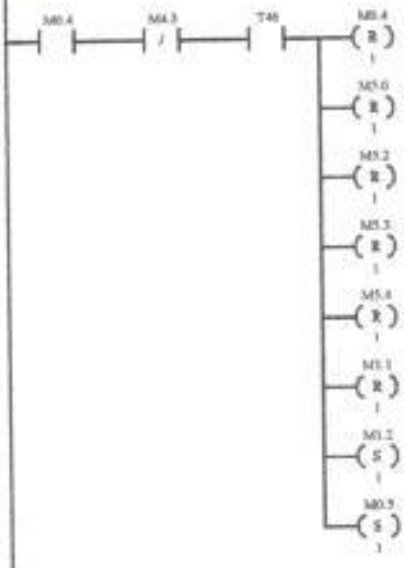
Segmento 68 LEVANTA MAS EL HOMBRO



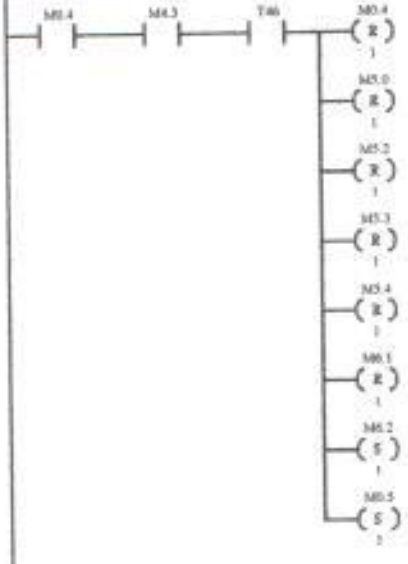
Segmento 69 BAJA EL CODO HACIA CAJA



Segmento 70 RESETEO DE LO USADO



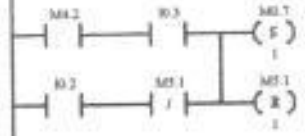
Segmento 71 RESETEO DE LO USADO



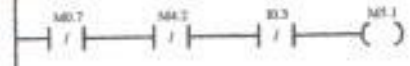
Segmento 72

Segmento 73 ********** MANDA LA BASE HASTA SU LIMITE Y RETORNA EL CONTROL A SUB0 **********

Segmento 74 A MENOS QUE ESTE ESTACIONADO EN EL MEDIO LA BASE LA MOVERA HASTA SU EXTREMO DERECHO

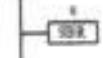


Segmento 75 MOVER BASE A SU EXTREMO DERECHO

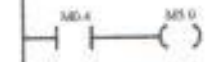


Segmento 76 RETORNO

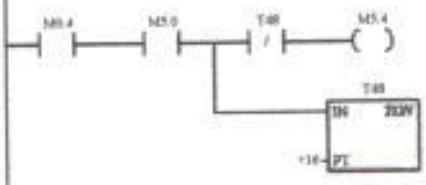
Segmento 77 POSICION 3 DE LA CAJA DE BOTELLAS



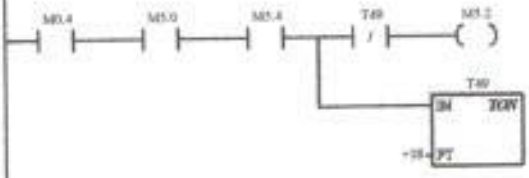
Segmento 78 ACTIVA POLARIDAD INVERSA



Segmento 79 BAJA MUÑECA

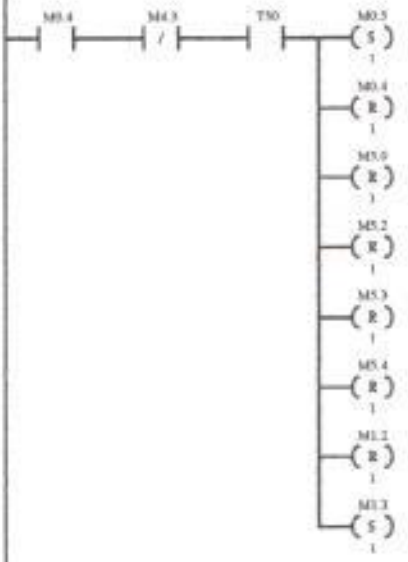


Segmento 80 LEVANTA MAS EL HOMBRO

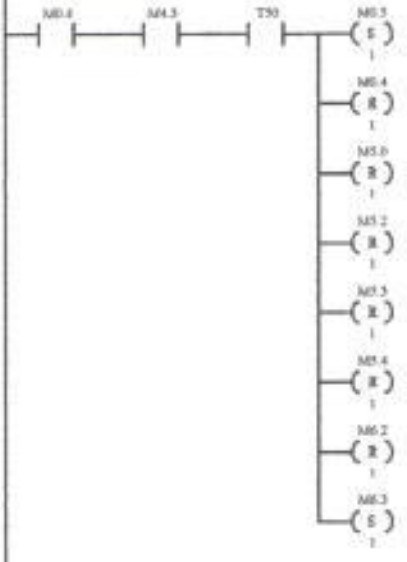


Segmento 81 BAJA EL CODO

Segmento 82 RESETEO DE LAS MEMORIAS USADAS Y SETEO DE LAS DE SALIDA



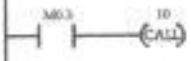
Segmento 83 RESETEO DE LAS MEMORIAS USADAS Y SETEO DE LAS DE SALIDA



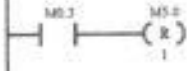
Segmento 84 RETORNO

Segmento 85

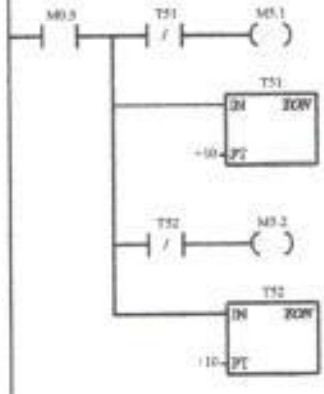
Segmento 86



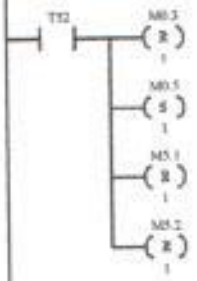
Segmento 87



Segmento 88



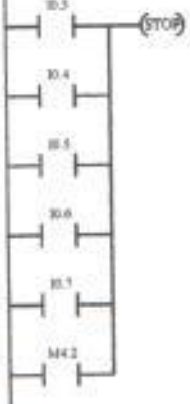
Segmento 89



Segmento 90

Segmento 91 SUBRUTINA DE FALLOS DE FUNCIONAMIENTO

Segmento 92



5. CONCLUSIONES Y RECOMENDACIONES

1) Mediante la lectura de la tesis hemos podido ver hasta donde podemos aplicar esta herramienta. También podemos ya tener la idea que necesitamos para programarlo y las consideraciones que hay que tener en cuenta cuando se haga. De igual forma está descrito dimensiones, y procesos de fabricación del brazo y finalmente el programa que lo controla lo que hace fácil su estudio.

2) El autómata es una herramienta tan necesaria que convendría que todos los que fueran a recibirse de ingenieros en electrónica no desconozcan su manejo. Pues este pequeño computador puede suplir en muchas oportunidades a circuitos muy voluminosos como en ocasiones pueden ser simplemente circuitos digitales,

contamos en el interior de un autómata con: contadores, temporizadores, inversores, comparadores, sumadores, multiplicadores, divisores binarios, manejo de los principales sistemas numéricos, convertidor analógico digital, servicio de interrupción, generación de tren de pulsos, convertidores de binario a BCD, convertir a ASCII, manejo de pila FIFO, LIFO; manejadores de interfaces de salida, facilidad en la programación (alto nivel). Ya que contamos con tantas herramientas dentro de este CPU, solo es cuestión de saber combinarlos mediante un programa los componentes digitales.

3) Como los demás países con un desarrollo electrónico, hoy las universidades más prestigiosas miran demostrar su poderío y nivel, mediante proyectos que envuelvan ya no solo una disciplina, sino que reúna a varias de ellas como ser dualidad entre mecánica y electrónica, pues como vemos las universidades son contratadas por firmas internacionales para elaborar robótica, maquinaria semiautomática, maquinarias digitales. En mi parecer sería de mucha utilidad que una vez que el alumno está capacitado para el diseño, o manejo de circuitos unan fuerzas con los ingenieros de mecánica para que juntos se levante el nombre de la institución.

4) Es de suma importancia que aprendamos ha hacer proyectos de tesis, pero, de una vez viendo las necesidades de la industria, es decir que la industria use y necesite de la universidad para desarrollar su nivel de industrialización, que todas las

maquinarias sean de la ESPOL y se asegure que mediante esos proyectos el alumno ya reciba una ganancia al igual que la institución. También es importante lograr que puedan ser vendidos cada uno de los proyectos de tesis. Así como este proyecto logré que fuera patentado por una empresa privada cuyo nombre pidió quedara en reserva, al igual que el papel que desarrolla dentro de la empresa. Hasta donde puedo informar es una fabrica de plásticos y usos de polímeros para la fabricación de botellas plásticas, embaces plásticos, bujes de nilón, piezas en general, y como en ese proceso hay muchos movimientos del personal que son repetitivos y monótonos han preferido colocar este robot en el lugar de un personal de planta, ya que puede realizar este trabajo repetitivo sin necesidad de que en algún momento haya que detener la máquina por necesidades humanas.

6. ANEXOS

6.1. ANEXO A

A Lista de operaciones

A.1 Lista de operaciones

A.1.1 Juego de operaciones básicas

☒ para módulos de organización (OB) ☒ para módulos funcionales (FB)

☒ para módulos de organización (OB) ☒ para módulos funcionales (FB)

Ope-Ración (AWL)	Operandos Admisibles	V 1	K 2	E 3	AG S5 Tiempo de perif. int.	- 90U ejec. En us perif. ext.	AG S5 Tiempo de perif. int.	- 95U ejec. en us perif. ext.	Descripción de la función
Operaciones combinacionales (lógicas)									
U	E, A	N	S	N	1...2	3...5	1...2	3...5	Combinación Y: Consulta al estado de señal "1"
	M	N	S	N	3...5		3...5		
	T	N	S	N	6...10		6...10		
	Z	N	S	N	3...6		3...6		
UN	E, A	N	S	N	2	3...5	2	3...5	Combinación Y: Consulta al estado de señal "0"
	M	N	S	N	3...5		3...5		
	T	N	S	N	6...10		6...10		

	Z	N	S	N	3...6		3...6		
O	E, A	N	S	N	1...2	3...5	1...2	3...5	Combinación O: Consulta al estado de señal "1"
	M	N	S	N	3...5		3...5		
	T	N	S	N	6...10		6...10		
	Z	N	S	N	3...6		3...6		
ON	E, A	N	S	N	2	3...5	2	3...5	Combinación O: Consulta al estado de señal "0"
	M	N	S	N	3...5		3...5		
	T	N	S	N	6...10		6...10		
	Z	N	S	N	3...6		3...6		
O		N	S	S	2...5		2...5		Combinación O de funciones Y
U(		N	S	S	4...8		4...8		Combinación Y de expresiones entre paréntesis (6 niveles)
O(		N	S	S	4...8		4...8		Combinación O de expresiones entre paréntesis (6 niveles)
)		N	S	S	4...10		4...10		Cerrar paréntesis (conclusión de una expresión entre paréntesis)
Operaciones de memoria (biestables)									
S	E, A	S	N	S	2	5...8	2	5...8	Poner el operando a "1"
	M	S	N	S	5...8		5...8		(activar el operando)

* 1 ¿ Depende del VKE? 2 ¿ Influencia del VKE? 3 ¿ Influye el VKE?

Ope-Ración (AWL)	Operandos Admisibles	V 1	K 2	E 3	AG S5 Tiempo de perif. int.	- 90U ejec. en us perif. ext.	AG S5 Tiempo de perif. int.	- 95U ejec. en us perif. ext.	Descripción de la función
Operaciones de memoria (biestables) (continuación)									
R	E, A	S	N	S	2	5...8	2	5...8	Poner el operando a "0" (borrar el operando)
	M	S	N	S	5...8		5...8		
=	T	N	N	S	2	4...7	2	4...7	Asignar al operando el valor del VKE
	Z	N	N	S	4...7		4...7		
Operaciones de carga									
L	EB	N	N	N	5	11	5	11	Cargar un byte de entrada de la PAE en el AKKU 1
L	AB	N	N	N	5	11	5	11	Cargar un byte de salida de la PAA en el AKKU 1
L	EW	N	N	N	5	15	5	15	Cargar una palabra de entrada de la PAE en el AKKU 1: Byte n → AKKU 1(Bits 8-15) Byte n + 1→ AKKU 1 (Bits 0-7)
L	AW	N	N	N	5	15	5	15	Cargar una palabra de salida de la PAA en el AKKU 1: Byte n → AKKU 1(Bits 8-15) Byte n + 1→ AKKU 1 (Bits 0-7)
L	PBO...31 PBO...127	N	N	N	—		39		¡Solo admisible en el OB 13! Cargar de la PAE de alarmas en el AKKU 1 un byte de entrada de los mod. de entrada dig / analóg.

	PB32/33 PB34/35 PB36...39 PB40...55	N	N	N	8/11 — — —	40...48 40...48 45...60 105	Cargar en el AKKU 1 un byte de entrada de los módulos de entrada dig. / analóg.
L	PW0...30 PW64...126	N	N	N	—	42	¡Solo admisible en el OB 13! Cargar de la PAE de alarmas en el AKKU 1 un byte de entrada de los mod. De entrada dig. / analóg.
	PW32 PW33...35 PW36...39 PW40...45	N	N	N	17 — — —	50...67 50...67 55...80 104	Cargar en el AKKU 1 una palabra de entrada de los modulos dentrada dig. / analóg.
L	MB	N	N	N	11	11	Cargar en el AKKU 1 un byte de marcas
L	MW	N	N	N	15	15	Cargar en el AKKU 1 una palabra de marcas Byte n → AKKU 1(Bits 8-15) Byte n + 1→ AKKU 1 (Bits 0-7)
L	DL	N	N	N	33	33	Cargar en el AKKU 1 una palbra de datos (byte izquierdo) del modulo de datos actual

1 ¿ Depende del VKE? 2 ¿ Influencia del VKE? 3 ¿ Influye el VKE?

Ope-Ración (AWL)	Operandos admisibles	V 1	K 2	E 3	AG S5 - 90U Tiempo de perif. int.	ejec. en us perif. ext.	AG S5 - 95U Tiempo de perif. int.	ejec. en us perif. ext.	Descripción de la función
Operaciones de carga (continuación)									
L	DR	N	N	N	35		35		Cargar en el AKKU 1 una palabra de datos (byte derecho) del módulo de datos actual
L	DW	N	N	N	35		35		Cargar en el AKKU 1 una palabra de datos del DB actual Byte n →AKKU 1 (Bits 8-15); Byte n + 1→AKKU 1 (Bits 0-7);
L	KB	N	N	N	5		5		Cargar en el AKKU 1 una constante (n°de un byte)
L	KC	N	N	N	5		5		Cargar en el AKKU 1 una constante (2 caracteres en código ASCII)
L	KF	N	N	N	5		5		Cargar en el AKKU 1 una constante (n°en coma fija)
L	KH	N	N	N	5		5		Cargar en el AKKU 1 una constante (hexadecimal)
L	KM	N	N	N	5		5		Cargar en el AKKU 1 una constante (configuración binaria)
L	KY	N	N	N	5		5		Cargar en el AKKU 1 una constante (n° de dos bytes)
L	KT	N	N	N	5		5		Cargar en el AKKU 1 una constante (Temporización codificada en BCD)
L	KZ	N	N	N	5		5		Cargar en el AKKU 1 una constante (Ajuste de contador

							codificado en BCD)
L	T, Z	N	N	N	14	14	Cargar en el AKKU 1 una temporización o un ajuste de contador (Codificados en binario)
LC	T	N	N	N	58	58	Cargar en el AKKU 1 una temporización o un ajuste de
	Z	N	N	N	59	59	contador (codificados en BCD)

Operaciones de transferencia

T	EB	N	N	N	2	5	2	5	Transferir el contenido del AKKU 1 a un byte de entrada (a la PAE)
T	AB	N	N	N	2	5	2	5	Transferir el contenido del AKKU 1 a un byte de salida (a la PAA)
T	EW	N	N	N	4	12	4	12	Transferir el contenido del AKKU 1 a una palabra de entrada (a la PAE) AKKU 1 (Bits 8-15)→Byte n; AKKU 1 (Bits 0-7)→Byte n+1

• 1 ¿ Depende del VKE? 2 ¿ Influencia del VKE? 3 ¿ Influye el VKE?

Ope-Ración (AWL)	Operandos admisibles	V 1	K 2	E 3	AG S5 Tiempo de perif. int.	- 90U ejec. en us perif. ext.	AG S5 Tiempo de perif. int.	- 95U ejec. en us perif. ext.	Descripción de la función
Operaciones de transferencia (continuación)									
T	AW	N	N	N	4	12	4	12	Transferir el contenido del AKKU 1 a una palabra de salida: AKKU 1 (Bits 8-15) → Bite n; AKKU 1 (Bits 0-7) → Bite n + 1
T	PE0... 31 PE64... 127	N N	N N	N N	—		41		¡Solo admisible en el OB 13! Transferir el contenido del AKKU 1 a la PAA de alarmas con actualización de la PAA
	PE 32/33	N	N	N	5/2		38		Transferir el contenido del AKKU 1 a la salida con actualización de la PAA
T	PW0... 30 PW0... 126	N N	N N	N N	—		45		¡Solo admisible en el OB 13! Transferir el contenido del AKKU 1 a la PAA de alarmas con actualización de la PAA
	PW32 PW40	N N	N N	N N	10 —		49 133		Transferir el contenido del AKKU 1 a la salida con actualización de la PAA
	PW 36/32	N	N	N	—		40 ... 58		Se pone a "0" el contador
T	MB	N	N	N	5		5		Transferir el contenido del AKKU 1 a un byte de marcas
T	MW	N	N	N	12		12		Transferir el contenido del AKKU 1 a una palabra de marcas (a al PAA): AKKU 1 (Bits 8-15) → Bite n; AKKU 1 (Bits 0-7) → Bite n + 1

T	DL	N	N	N	25	25	Transferir el contenido del AKKU 1 a una palabra de datos (byte izquierdo)
T	DR	N	N	N	26	26	Transferir el contenido del AKKU 1 a una palabra de datos (byte derecho)
T	DW	N	N	N	34	34	Transferir el contenido del AKKU 1 a una palabra de Datos
0peraciones de tiempo							
SI	T	S	N	S	64	64	Arrancar como impulso una temporización (depositada en el AKKU 1) (limitación de la señal)

Ope-Ración (AWL)	Operandos admisibles	V 1	K 2	E 3	AG S5 Tiempo de perif. int.	- 90U ejec. en us perif. ext.	AG S5 Tiempo de perif. int.	- 95U ejec. en us perif. ext.	Descripción de la función
Operaciones de tiempo (continuación)									
SV	T	S	N	S	64		64		Arrancar como impulso prolongado una temporización (dep. en el AKKU 1) (lim. Y prolong. de la señal)
SE	T	S	N	S	65		65		Arrancar como retardo a la conexión una temporización (depositada en el AKKU 1)
SS	T	S	N	S	65		65		Arrancar como retardo a la conexión memorizada una temporización (depositada en el AKKU 1)
SA	T	S	N	S	64		64		Arrancar como retardo a la desconexión una temporización (depositada en el AKKU 1)
R	T	S	N	S	21		21		Reponer (borrar) una temporización

Operaciones de contaje

ZV	Z	S	N	S	35	35	Contaje hacia delante en 1
ZR	Z	S	N	S	40	40	Contaje hacia atrás en 1
S	Z	S	N	S	62	62	Activar (ajustar) un contador
R	Z	S	N	S	17	17	Borrar (reponer) un contador

Operaciones aritmèticas

+F		N	N	N	19	19	Sumar dos números en coma fija: AKKU 1 + AKKU 2. Resultado evaluable a través de ANZ 1/ANZ 0/OV.
+F		N	N	N	22	22	Restar dos números en coma fija: AKKU 2 - AKKU 1. Resultado evaluable a través de ANZ 1/ANZ 0/OV.

Operaciones de comparación

!=F		N	S	N	20	21	Comparar dos números en coma fija respecto a igualdad: Si AKKU 2 = AKKU 1, entonces VKE = "1". El resultado afecta a ANZ 1/ANZ 0
><F		N	S	N	22	22	Comparar dos números en coma fija respecto a desigualdad. Si AKKU 2 = AKKU 1, entonces VKE = "1". El resultado afecta a ANZ 1/ANZ 0

* 1 ¿ Depende del VKE? 2 ¿ Influencia del VKE? 3 ¿ Influye el VKE?

Ope-Ración (AWL)	Operandos admisibles	V 1	K 2	E 3	AG S5 Tiempo de perif. int.	- 90U ejec. en us perif. Ext.	AG S5 Tiempo de perif. int.	- 95U ejec. en us perif. ext.	Descripción de la función
Operaciones de comparación (continuación)									
>F		N	S	N	22		22		Compara dos números en coma fija respecto a superioridad : Si AKKU 2 > AKKU 1., entonces VKE = "1". El resultado afecta a ANZ 1/ANZ 0
>=F		N	S	N	22		22		Compara dos números en coma fija respecto a superioridad o igualdad : Si AKKU 2 ≥ AKKU 1 entonces VKE = "1". El resultado afecta a ANZ 1/ANZ 0
<F		N	S	N	22		22		Compara dos números en coma fija respecto a inferioridad: Si AKKU 2 < AKKU 1, entonces VKE = "1". El resultado afecta a ANZ 1/ANZ
<=F		N	S	N	22		22		Compara dos números en coma fija respecto a infer. o igualdad : Si AKKU 2 ≤ AKKU 1., entonces VKE = "1". El resultado afecta a ANZ 1/ANZ 0

Operaciones de llamada de módulo

SPA	PB	N	N	S	63	61	Salto absoluto (incondicional) a un módulo de programa
SPA	FB	N	N	S	65	63	Salto absoluto (incondicional) a un módulo de funcional
SPA	SB	N	N	S	—	61	Salto absoluto (incondicional) a un módulo de paso
SPA	PB	S	S	S	64	63	Salto condicional a un módulo de programa
SPA	FB	S	S	S	67	65	Salto condicional a un módulo funcional
SPA	SB	S	S	S	—	63	Salto condicional a un módulo de paso
SPA	DB	N	N	N	30	30	Llamada de un módulo de datos
SPA	DB	N	N	S	—	109	Crear o borrar un módulo de datos

Operaciones de retorno

BE		N	N	S	37	39	Terminar un módulo (fin de módulo)
BEB		S	S	S	38	40	Terminar módulo de forma condicional

Ope-Ración (AWL)	Operandos admisibles	V 1	K 2	E 3	AG S5 Tiempo de perif. int.	- 90U ejec. en us perif. ext.	AG S5 Tiempo de perif. int.	- 95U ejec. en us perif. ext.	Descripción de la función

Operaciones de retorno (continuación)							
BEA		N	N	S	37	39	Terminar módulo de forma absoluta (incondicional) (no utilizables en módulos de organización)
Operaciones nulas							
NOP 0		N	N	N	0	0	Operación nula (todos los bits borrados)
NOP 1		N	N	N	0	0	Operación nula (todos los bits activados)
Operaciones STOP							
STP		N	N	N	1	1	Stop: el ciclo es aún terminado. Se activa en el USTACK el identificador de error STS
Operaciones de estructuración de imagen							
BLD 130		N	N	N	0	0	Instrucción de estructuración de imagen para el PG: Crear una linia libre a traves de Carriage Retrun
BLD 131		N	N	N	0	0	Instrucción de estructuración de imagen para el PG: Canviar a lista de instrucciones (AWL)
BLD 132		N	N	N	0	0	Instrucción de estructuración de imagen para el PG: Canviar a esquema de funciones (FUP)

BLD 133		N	N	N	0	0	Instrucción de estructuración de imagen para el PG: Canviar a esquema de contactos (KOP)
BLD 255		N	N	N	0	0	Instrucción de estructuración de imagen para el PG: Terminar segmento

• 1 ¿ Depende del VKE? 2 ¿ Influencia del VKE? 3 ¿ Influye el VKE?

A.1.2 Operaciones complementarias las más comunes.

Ope-Ración (AWL)	Operandos admisibles	V 1	K 2	E 3	AG S5 Tiempo de perif. int.	- 90U ejec. en us perif. ext.	AG S5 Tiempo de perif. int.	- 95U ejec. en us perif. ext.	Descripción de la función
Operaciones combinacionales									
U=	Operando formal E, A, M, T, Z	S	N	S	—		43...64		Combinación Y : Consultar el operando formal al estado de señal "1" (formato de parámetro BI)
UN=	Operando formal E, A, M, T, Z	S	N	S	—		44...65		Combinación Y : Consultar el operando formal al estado de señal "0" (formato de parámetro BI)
O=	Operando formal E, A, M, T, Z	N	N	S	—		43...64		Combinación Y : Consultar el operando formal al estado de señal "1" (formato de parámetro BI)
ON=	Operando formal E, A, M, T, Z	N	N	S	—		44...65		Combinación Y : Consultar el operando formal al estado de señal "0" (formato de parámetro BI)
UW		N	N	N	16		16		Combinación Y (por palabras): AKKU 2 con AKKU 1 :resultado en AKKU1. El resultado afecta a ANZ 1/ANZ 0
OW		N	N	N	16		16		Combinación O (por palabras): AKKU 2 con AKKU 1 :

							resultado en AKKU1. ANZ 1/ANZ 0 evaluable
XOW		N	N	N	16	16	Combinación O exclusiva (por palabras): AKKU 2 con AKKU 1 :resultado en AKKU1. El resultado es evaluable en ANZ 1/ANZ 0
Operaciones de prueba DE BIT							
P		N	N	N	—	5	Probar si está a "1" un bit de una palabra de temporización o ajuste de contador
P		N	N	N	—	32	Probar si está a "1" un bit de una palabra de datos
P		N	N	N	—	5	Probar si está a "1" un bit de una palabra de datos dentro de la zona de datos del sistema
PN		N	N	N	—	5	Probar si está a "0" un bit de una palabra de temporización o ajuste de contador

A-1 A-2

6.2. ANEXO B

6.2.1. Operaciones más comunes del la familia S7

Abreviatura	Descripción:
LD	Cargar un valor binario
U	Es la unión lógica Y
O	Es la unión lógica O
=	Asignar un valor binario
S	Setea una valor binario poniéndolo a 1
R	Resetea un valor binario poniéndolo a 0
NOT	Negar el primer valor de la pila
LBL	Definir meta
JMP	Saltar a meta
CALL	Llamar a subrutina
SBR	Definir subrutina
RET	Retornar
INT	Comenzar subrutina de interrupción
END	Fin condicional
MEND	Fin de programa principal

STOP	Para a STOP
NOP	Operación Nula
TON	Temporización de retardo a la conexión.
ENI	Habilitar interrupción
DISI	Inhivir interrupción
PLS	Salida de impulsos
ZV	Contar adelante
RETI	Retornar de subrutina de interrupción
FOR, NEXT	Bucles

6.3. ANEXO D:

6.3.1. Abreviaturas usadas durante la elaboración del trabajo

Abreviación	Descripción
A, Q	Salidas
AB	Byte de salida
AI	Cantidad de entradas analógicas que deben de leerse
AKKU1,2	Acumulador del CPU. Al cargar AKKU1, se desplaza su contenido al 2
ANZ 0 / 1	Indicación de resultado 0/1
AW	Palabra de salida
AWL	Lista de instrucciones STEP5
BF	Constante de Byte
BS	Zona de datos del sistema
D	Dato (1 bit)
DB	Módulo de datos
DL	Palabra de datos (byte izquierdo)
DR	Palabra de datos (byte derecho)
DW	Palabra de datos

E, I	Entradas
EB	Byte de entrada
EF	Buzón de recepción
EW	Palabra de entrada
FB	Módulo funcional
FUP	Esquema de funciones
I, E	Entradas
IN	Alarma de flanco negativo
INP	Alarma de flancos negativo y positivo
IP	Alarma de flanco positivo
IPN	Alarma con flanco positivo y negativo
KB	Constante 1 byte
KBE	Parámetro del DB1. (emisión)
KBS	Parámetro del DB1. (recepción)
KC	Constante (2 caracteres)
KF	Constante de coma fija
KH	Constante hexadecimal
KM	Configuración binaria 2 bytes
KOP	Esquema de contactos
KT	Constante (temporizador)
KY	Constante 2 bytes

KZ	Constante (valor contador)
M	Marca
MB	Byte de Marca
MW	Palabra de marca
NT	Cantidad de temporizadores procesados
OB	Módulos de organización para aplicaciones especiales
OBA	Identificador de bloque en DB1 para entradas analógicas integradas
OBC	Identificador de bloque en DB1 para contadores integrados
OBI	Identificador de bloque en DB1 para alarma integrada
OHE	Liberar contador horas operación
OHS	Ajustar contador horas operación
OP	Aparato de operación
OV	Indicador de desbordamiento
PAA	Imagen de proceso de salidas
PAE	Imagen de proceso de entradas
PB	Módulo de programa
PB o PY	Byte de periferia
PG	Aparato de programación
PW	Palabra de periferia
Q, A	Salidas
SAV	Salvar hora tras la última transición de STOP a RUM

SB	Módulo de paso
SET	Ajustar hora, fecha
SF	Situación del buzón de emisión
SM	Marcas especiales de memoria
STP	Actualizar hora en STOP
STW	Situación de la palabra de estado (reloj – calendario integrado)
T	Temporizadores
TFB	Indicador de bloque en DB1 para módulo funcional de temporizadores
TIS	Parámetro del DB1 para ajustar hora de alarma
VKE	Resultado de la combinación
Z	Contadores

7. BIBLIOGRAFIA

1 BALCELLS, JOSEP Y ROMERAL, JOSÉ, Autómatas Programables, Marcombo, Barcelona – España, 1997, Pgs. 145 hasta 153 ; 351 hasta 384.

2 SIEMENS ALEMANIA, Manual Autómata Programable (configuración, instalación y datos de CPU), 1995, Pgs. 1-1 hasta 3-7.

3 SIEMENS ALEMANIA, Manual Sistematic S7-200 SET DE PRACTICAS.

4 SIEMENS ALEMANIA, Manual Sistematic Software (Manual de Referencia), 1996, Pgs. 1–1 hasta 7-48.

5 ESCUELA PROFECIONAL SALECIANA DE SERRIÁ "Barcelona España", Manual de Autómata Programable del Instituto Politécnico, 1997, Pgs. Completo.

6 Manual Sistemátic S5, SIEMENS ALEMANIA, 1995, Pgs. Completo.