**UNIVERSIDAD DEL VALLE DE GUATEMALA**
Facultad de Ciencias y Humanidades
Departamento de Ingeniería Electrónica.

# Interface humano-máquina para control de un robot.

Mynor Antonio Joachín Godínez



Guatemala
1998.

Vo. Bo.:

(f) _____.

Dr.- Ing. Manuel Antonio López Valdés.

Asesor.

Terna examinadora:

(f) _____.

Dr.- Ing. Manuel Antonio López Valdés.

(f) _____.

Ingeniero Luis Furlán Collver.

(f) _____.

Ingeniero Daniel Sandoval Monzón.

Fecha de aprobación: 06 de Agosto de 1,998.

**UNIVERSIDAD DEL VALLE DE GUATEMALA**

Facultad de Ciencias y Humanidades

Departamento de Ingeniería Electrónica.

# INTERFACE HUMANO-MAQUINA PARA CONTROL DE UN ROBOT.

Mynor Antonio Joachín Godínez

Trabajo de graduación presentado para optar
al grado académico de:
LICENCIATURA EN INGENIERÍA ELECTRÓNICA.

Guatemala
1998.

# ÍNDICE

***Dedicatoria:***

A Dios por darme fuerza y fortaleza en cada paso de mi vida.

A la Virgen María.

A mis padres, *Otoniel* y *Fidelina*, por su apoyo y esfuerzo desde el día que nací, los cuales han permitido ser lo que soy ahora.

A mi esposa e hijas: *Ericka*, *María José* y *María Isabel*.

***Agradecimientos:***

Al *Centro Universitario Ciudad Vieja* (CUCV) por la permitirme residir y formarme durante cuatro años.

# I. Sumario.

Este documento presenta los principios generales de la nueva tendencia en el área de la automatización industrial, en lo que corresponde al control, generación de tareas (*job*) y visualización; permitiendo de esta forma un mejor desempeño entre el operador y la maquinaria. Para lograr esta mejor relación entre el operador y maquinaria es utilizando como medio de comunicación un *PC* (computadora personal), ésta a su vez utiliza el sistema de interfase gráfico usuario (*GUI* Visual Basic).

La aplicación desarrollada en Visual Basic utiliza el puerto paralelo (RS 232 / RS 485) para comunicarse con un controlador lógico programable (PLC) el cual dependiendo de la información que envíe o solicite la aplicación, el controlador se encargará de recolectar información de los dispositivos de campo (sensores inductivos, magnéticos, fotoeléctricos, etc) y ejercer control sobre elementos actuadores (electroválvulas discretas o analógicas, motores, etc) para así desarrollar el control de la maquinaria, que en este específico caso es para un montacargas. Dicha aplicación es llamada interface humano-máquina (HMI).

La interface humano máquina es parte de todo un gran sistema de administración orientada a la automatización. La interfase humano máquina forma parte de un gran sistema de automatización el cual está estructurado por jerarquías.

Este sistema en la parte más alta de jerarquía tiene la tarea de delegar secuencias de producción, estas secuencias de producción se encargan de gobernar celdas o células y estas a su vez están compuestas de varios equipos, los cuales son gobernados por interfaces humano-máquina. Es en este punto donde se propone una solución viable para la industria guatemalteca que esta creciendo y mejorando sus técnicas de producción sin tener que hacer una inversión económica alta.

# II. Introducción.

El continuo desarrollo que ha hecho el ser humano en busca de la perfección en sus actividades del diario vivir, ha implicado que las herramientas utilizadas para ejecutar dichas actividades tuvieran y tengan una continua evolución, con el fin de ser más eficientes y productivas.

Hasta hace algunas décadas se empezaron a introducir sistemas que realizan cierto tipo de tareas. Estas tareas que por su complejidad de manejo y riesgo que implican para el ser humano, con lo cual eran difíciles de desarrollar continua, homogénea y eficazmente.

Sistemas que desencadenaban una serie de acciones electromecánicas con base en descripción físicamente establecida, fueron sustituidos por equipos que se describía mediante un lenguaje ensamblador la secuencia a desarrollar, sin necesidad de crear otra distribución de conexiones físicas.

El advenimiento de sistemas abiertos y eficaces ha logrado alcanzar una estructuración en todos los campos y la industria de la automatización es una de ellas. Se forman capas estructuradas para la comprensión general del sistema las cuales son actualmente:

- Sistemas de ejecución de plantas (*MES*).
- Interface humano-máquina (HMI).
- Controladores lógicos programables (*PLC*).
- Dispositivos de campo y/o periferia (*CAN*)

La interfase humano-máquina empezó a tener un mayor énfasis a partir de la necesidad de hacer más amigable la administración de

unidades productivas con los operadores de dichos equipos; teniendo un desarrollo gradual en la utilización de botoneras, manijas, selectores, lámparas, desplegadores de texto, etc., hasta llegar al uso procesadores para dichas tareas especificas ( ser el intermediario entre el ser humano y la maquinaria).

Desde el punto de vista del diseñador, la interface humano-máquina es una imagen virtual y amigable de lo que está sucediendo dentro del controlador lógico programable, en el cual una variable que indica un parámetro del proceso, puede desplegarse como un gráfico con animación de cambio de color, posición, dimensiones etc.

## III. RESULTADOS PRELIMINARES.

1.1 La pantalla principal ha brindado una herramienta poderosa para el operario en lo que respecta a la administración del montacargas, permitiendo una distribución más organizada en las celdas de almacenaje.

1.2 La interfase humano-máquina permite que ahora la persona que realizaba la tarea de transportar las paletas de una fuente a un destino y quien organiza la distribución de dicha paleta, sea la misma.

1.3 Al existir niveles de acceso para operar la interfase, permite un mejor control del tipo de personas que generen y anulen tareas.

1.4 Al existir la interfase humano máquina nos permite que el PLC no utilice módulos adicionales de entrada y salidas digital, para recibir señal de manijas botoneras, bombillos, esto implicada la reducción de cableado y accesorios, con lo cual se reducen costos.

1.5 Al utilizar el puerto RS 485 del PLC como medio de comunicación con la interfase humano-máquina, optimizamos el funcionamiento de dicho puerto, porque su función principal es para configurar y programar el autómata, lo cual se realiza y utiliza únicamente cuando se parametriza y programa inicialmente o cuando se necesita hacer una mejora al programa.

# I. Antecedentes

## A. Marco teórico

### a. Niveles y jerarquías de CIM

**Sistema de planeación en manufactura (MPS):**

El primer sistema de planeación en manufactura fue desarrollado a finales de 1960 y fue llamado: Planeación de requerimientos de materia prima (*MRP*). Este sistema incluía dos módulos en funcionamiento de lotes: Planeamiento de requerimientos de materia prima y Planeamiento de capacidad requerida. Durante 20 años el MRP tuvo dos a tres generaciones. El nuevo sistema MRP incluye: órdenes de entrada, planeación de distribución, planeamiento de recursos, horarios de producción, planeamiento de materia prima requerida, control de producto en piso de la planta.

A principios de 1980 cuando fueron incorporados los módulos de contabilidad y finanzas, el nombre fue cambiado por MRP II dando a conocer las nuevas capacidades desarrolladas.

Ahora el sistema incluye la habilidad para la administración de multi-plantas y operaciones multi-países, adicionalmente tiene un módulo de mantenimiento preventivo, un módulo avanzado de planeamiento y calendarización.

Es un método eficaz de operación y planeamiento de recursos en la fabricación, utilizado para las hojas de operación eficaz de todos los recursos de una compañía de fabricación puesta en ejecución. Este sistema es desglosado en unidades y hojas de operación de producción, hojas de requerimiento de materiales, los sistemas de ayuda de las hojas de operación de requisitos de capacidad productiva, capacidad para la ejecución y de material.

La salida de este sistema se integra con informes financieros tales como el plan de negocios, el informe de la consolidación de compras, el presupuesto que envía, y las proyecciones del inventario. Las hojas de operación de recursos de fabricación son una consecuencia y una extensión directa de las hojas de operación de requisitos de materiales que forman un circuito cerrado.

La hoja de operación permite fijar la prioridad a la hoja técnica de operación para utilizar el horario de fabricación (plan principal de fabricación) qué determina que componentes deben ser pedidos (de un surtidor externo o de un nivel inferior del departamento) y cuándo deben realizarse estos pedidos. Permite así generar un horario específico de los componentes necesarios en una secuencia ordenada para cada componente, para que este disponible en el momento de arribar al respectivo nivel de ensamblaje. El sistema lleva una cuenta de los materiales, de los datos de inventario y del plan de fabricación principal para calcular los requisitos para las piezas.

(Century Manufacturing, Thomas G. Gunn)

**b. Sistema de ejecución en manufactura (MES):**

El termino *MES* (*Manufacturing execution system*) fue concebido en 1990 para describir una colección de funciones que reside entre un sistema de planeamiento y un sistema de control y supervisión de manufactura.

El desarrollo de estas dos disciplinas (Sistema de planeamiento y supervisión/control de manufactura), crea un espacio cultural entre éstas. Este espacio causó la inhabilidad para intercambiar información pertinente entre estos ambientes. El MES es un puente entre el espacio de sistemas

orientados a negocios y sistemas orientados al control en tiempo real de manufactura en una planta.

El MES típicamente reside entre el sistema *MRPII/ERP* y el sistema *HMI/SCADA* (*Supervisory control software*), la interface entre MRPII/ERP y HMI/SCADA están usualmente utilizados para pasar apropiadamente información hacia abajo, es decir, desde ERP hacia el MES y desde el MES hacia HMI/SCADA, y de similar forma hacia arriba.

Para un MES, la ejecución es la herramienta que toma la demanda del consumidor en forma de ordenes y traslada a un plan de ejecución basado en una evaluación en tiempo real del proceso en la planta el cual requerirá materia prima para funcionar. Este intercambio típicamente ocurre entre el ERP y el MES.

Adicionalmente el MES intercambia información con el sistema de control, trasladando parámetros para que el equipo efectúe las tareas y recibe retroalimentación en tiempo real del desarrollo de una tarea y evaluación del proceso. El MES recibirá información tal como una orden, listado de materiales, recetas, diagramas, requerimientos de recursos, rutinas y planes de proceso, evaluaciones de inventarios y un procedimiento estándar de operación o una instrucción de trabajo desde el sistema ERP.

El MES puede trasladar demanda del consumidor en un plan de manufactura que optimiza el sistema manufacturero como un todo, basado en la evaluación de tiempo real del proceso y asignan un específico recurso, maquinaria y herramientas; éste a su vez recibirá actualizado de la maquinaria, el proceso y el personal como se desplazó el producto a través de las líneas de manufactura hasta ser completado.

En muchas oportunidades, el MES está integrado a un sistema HMI (*Human-machine interface*) de una planta para leer, escribir y monitorear las variables de tiempo real del proceso, el sistema está críticamente especializado para la colección y reconstrucción histórica del proceso, esto permite tener un historial de la producción para ayudar acertadamente que condiciones del proceso fueron las que generaron tal defecto en el producto.

(MES Operational Excellence Enabler, Erick Marks)

El MES es un parte de una trilogía de productos para la organización de fábrica. Las implementaciones del MES incrementa al MPS, y se enfoca a los dos extremos de la producción, el primero a planeamiento, y el otro a sistemas de supervisión, este último esta enfocado al control físico del equipo. Estratégicamente esta colocado entre los sistemas corporativos de planeación y el equipo de piso en la fábrica. La principal función es trasladar planes en prioridades de ejecución que accionan la producción, y comunican los datos de producción generados en el nivel de planta hacia gerencia para la toma de decisiones críticas. Adicionalmente el resumen de producción ayuda a la gerencia de producción en el diseño de estrategias y así proveer de un proceso continuo de producción.

El MES efectivamente recibe las ordenes como un requerimiento de producción en la planta, administra todo el proceso de información relevante para satisfacer que ordenes son planeadas, de cual calendarización o recalendarización, que ruta, que instrucciones de trabajo, monitoreo de trabajos en proceso, aseguramiento de calidad y la maquinaria y utensilios utilizados.

(MES Functionalities & MRP to MES Data Flow, White Paper 2)

### c. Sistemas de supervisión de manufactura (MSS):

El desarrollo de la interface del sistema HMI/SCADA provee acceso a los dispositivos de accionamiento del PLC, de una forma gráfica que permite un mejor monitoreo y control del proceso en un medio más integral. (Mas adelante se verá con profundidad este punto). Esta capa está íntimamente relacionada con la de control, por lo que puede encontrarse en otros documentos Sistemas de supervisión / control de manufactura.

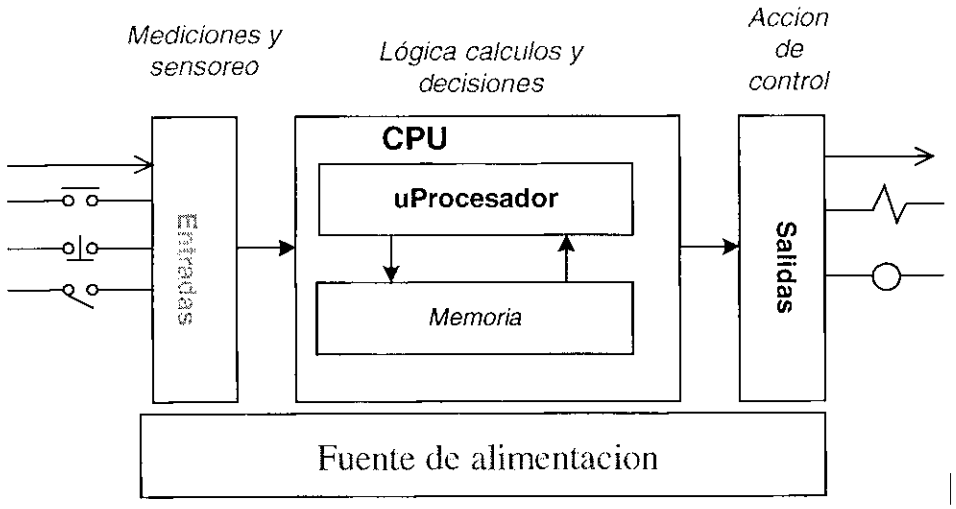
### d. Sistemas de control de manufactura (MCS):

Esta capa es la parte baja de un sistema de manufactura, la cual es la que tiene el contacto directo con la maquinaria. Está constituida por sub-capas, las cuales son:

1) Controlador lógico programable.
2) Red de control de sensores y actuadores (CAN)

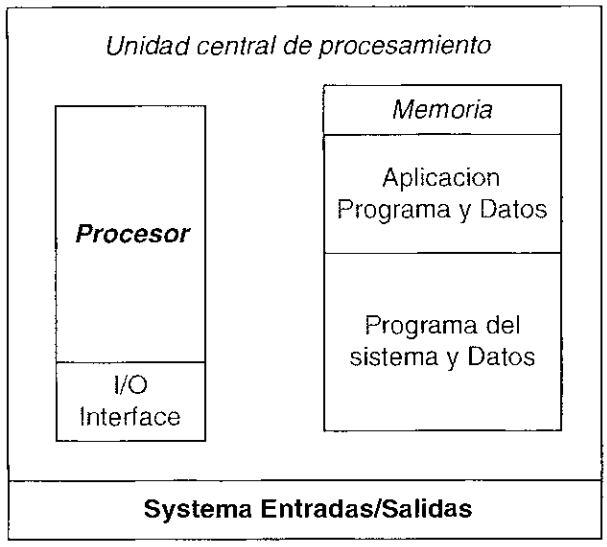
El controlador lógico programable (PLC), es un dispositivo de control industrial, diseñado específicamente para controlar maquinaria y procesar bajo una secuencia almacenada en memoria que es retroalimentada por los sensores de campo y dispositivos actuadores. Un controlador lógico programable es un aparato electrónico digital con una memoria de programación para almacenar instrucciones para implementar funciones específicas, secuencias, tiempos, conteo, operaciones aritméticas, para controlar maquinaria y proceso.

Los componentes básicos incluyen: sistema de entradas / salidas, el cual provee la interface al campo de sensores y dispositivos actuadores; la unidad central de proceso, la cual procesa el programa. El programa realizará las decisiones de control; y el sistema de alimentación de energía, el cual suministra potencia a la unidad central de proceso y el sistema de entradas y salidas.




*Unidad central de proceso:*

La unidad central de procesamiento es el cerebro del controlador programable. Este hace todas las decisiones relacionadas al proceso y la maquinaria. Durante la operación, el CPU recibe entradas de varios dispositivos de campo que realizan decisiones lógicas basadas en el programa almacenado en memoria, y controla los dispositivos de salida acorde al resultado de la lógica programada. Este proceso de lectura de entradas, ejecución del programa y control de salidas es conocido como ciclo de barrido.




Los elementos de la CPU proveen inteligencia del controlador, pero dicho atributo reside en el procesador y todo soporte de circuitos que es el cerebro y dirigente del PLC. Por esta razón los términos procesador y CPU son utilizados intercambiablemente.

El procesador central del PLC, es similar a una computadora de hoy en día, está basada en un microprocesador. Mientras los controladores pequeños utilizan generalmente un microprocesador sencillo, los medianos y grandes controladores incorporan microprocesadores complejos para formar una unidad centralizadora de procesamiento. En los microprocesadores pequeños todas las operaciones matemáticas, lógicas, control de salidas / entradas y operaciones de comunicación; algunas de éstas deberán ser realizados por procesadores separados pero con trabajo simultáneo. Para el intercambio de control y procesamiento, una técnica conocida como: *procesamiento paralelo.*

Otros tipos de arreglos de procesadores están referidos a: procesamiento distribuido, el cual utiliza módulos entradas / salidas

inteligentes. Este microprocesador basado en módulos tiene su propia memoria y puede ser programado para desarrollar control, comunicación u otras tareas especificas independientes del procesador principal. Un módulo típico es el proporcional-integral-diferencial- (PID), el cual realiza control de lazo cerrado independiente del procesador central.

La unidad procesadora determina toda la funcionalidad del PLC. También los puntos centrales de inteligencia son responsables del cálculo, toma de decisiones, y coordinación de soporte de los componentes del PLC.

La unidad procesadora deberá también monitorear la integridad de los componentes del sistema, incluyéndose a sí mismo. El procesador realiza esta pesada tarea a través de ejecutar el programa almacenado de la memoria del sistema.

El sistema de memoria consta de dos secciones básicas: 1) sistema operativo y 2)sistema de datos y área de trabajo del procesador.

El sistema operativo (*software*), también referido al sistema *firmware*, es un conjunto de programas que típicamente residen en el integrado EPROM, este dispositivo está localizado en el modulo del CPU. Este tiene almacenado permanentemente programas, los cuales son considerados parte de la máquina en si, proporcionando al controlador una única capacidad. El sistema *firmware* determina que el CPU es capaz de hacer.

Las funciones básicas incluyen: monitoreo y control de entradas / salidas, ejecución del programa, comunicación con el programador y otros dispositivos, diagnóstico, y otra variedad de funciones de rutina. Para ejecutar el programa del sistema, el procesador es capaz de realizar todas estas operaciones, incluyendo el procesamiento del programa de aplicación.

*Sistema Entradas / salidas:*

El sistema de entradas / salidas es la sección del PLC en el cual los sensores y dispositivos actuadores son conectados y a través del PLC realiza monitoreo y controla la maquinaria o el proceso. Está proporcionada para distintas señales de campo hacia el controlador y las salidas del controlador, también provee un aislamiento entre el PLC y ambientes ruidosos.

Cada sistema de entradas y salidas está provisto de varias tarjetas de entradas y salidas que proveen interface a un gran rango de corrientes estándares y señales de voltaje. Los módulos de entradas aceptan señales de dispositivos tales como: límite de carrera, selectores, botoneras, contactos de relays y sensores analógicos. Los módulos de salidas controlan las señales que activan dispositivos tales como: motores, arrancadores, solenoides, luces indicadoras, y válvulas de posición.

*Sistema de alimentación de energía:*

El tercer componente es la fuente de alimentación, la cual provee niveles bajos de voltaje DC requeridos para activar los circuitos electrónicos de otros componentes del controlador lógico programable. Esta convierte voltajes de línea (115VAC, 230VAC) a valores bajos (5VDC, 10VDC,24VDC), requeridos por el CPU y los módulos de entradas y salidas. Un importante punto de notar es que la alimentación del PLC esta diseñada únicamente para proveer potencia para operaciones internas de los módulos de entradas / salidas y CPU. La potencia requerida para operar dispositivos de campo es normalmente proveída de fuentes externas de alimentación.

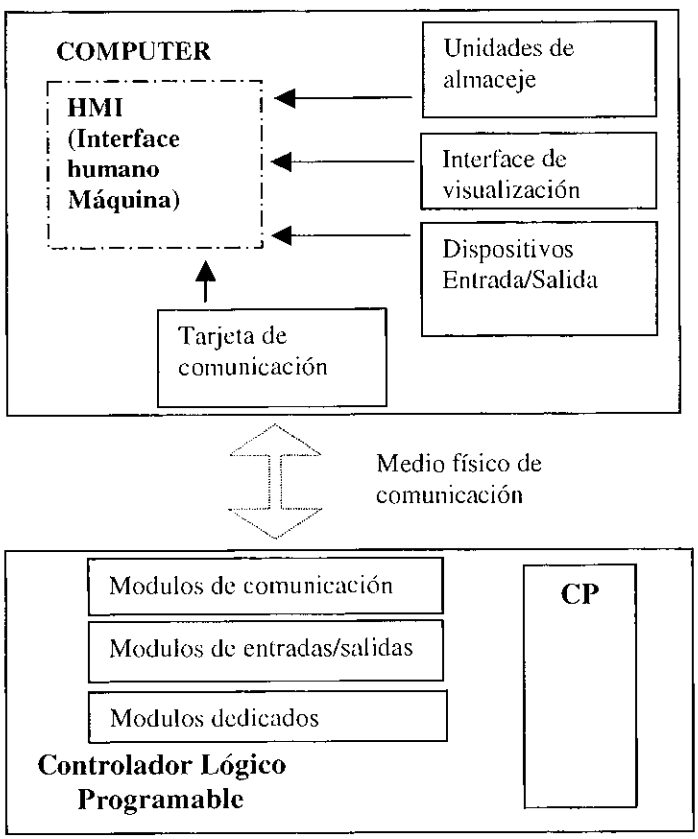
(Programable logic controller, Jones).

## 2. Interface humano máquina (HMI):

### a. Descripción y funcionalidad:

Esta es una herramienta que está asociada directamente con los controladores lógicos programable (PLC) y la automatización. Por lo tanto provee un acceso directo a los dispositivos que accionan el controlador lógico programable (PLC).

El HMI es una interface gráfica que despliega a través de la pantalla de un monitor diferentes ventas conceptualmente divididas a las necesidades del proceso de manufactura. Está orientado a un mejor desempeño entre el operador y el equipo a controlar y utiliza las facilidades que brinda los nuevos sistemas de computadoras, tal es el caso de manejo de gráficos, base de datos, puertos de comunicación.



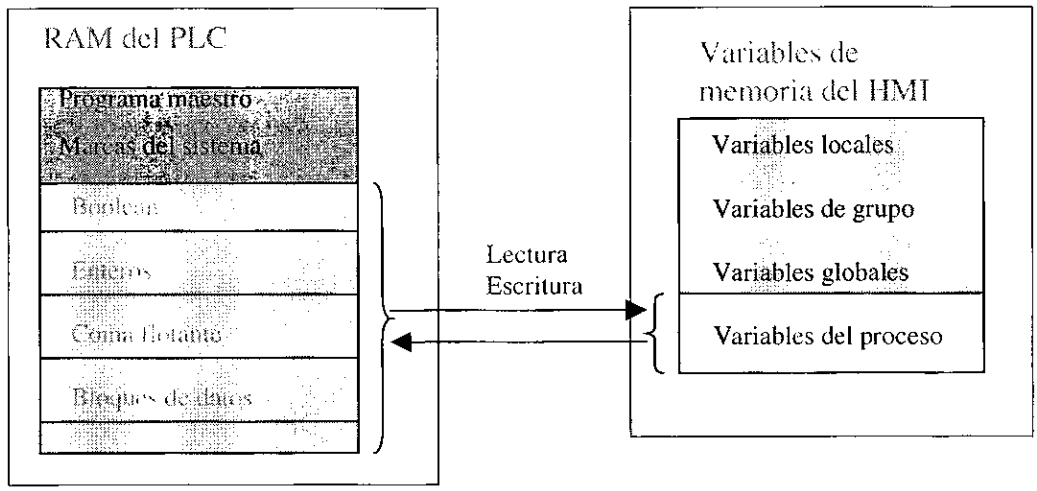
La interface humano gráfica tiene una imagen de las marcas, posiciones de memoria especificadas, registros, bloques de datos del controlador, los cuales se actualizan en cada ciclo del programa del PLC, y a su vez la interface puede modificar cualquiera de estos valores y descargarlo en el siguiente ciclo. Esto permite tener un monitoreo y control del PLC en tiempo cuasí-real. Cada uno de los datos de memoria del controlador se manejan según el programa almacenado en memoria del PLC. Por lo tanto aunque se puede modificar cualquier dato en el controlador únicamente es aconsejable realizarlos en los datos que se han definido en el programa del controlador.

El controlador lógico programable maneja varios tipos de datos según la capacidad de la CPU, por lo general éstos son:

- Boléanos (entradas / salidas digitales, banderas)
- Enteros (entradas / salidas analógicas, cálculos numéricos)
- Coma flotante
- Tiempo y contadores
- Bloques de datos

Estas variables que son simplemente posiciones de memoria agrupados según el manejo que realice la unidad central. Estas posiciones de memoria son consultadas y modificadas por una porción de memoria en la cual se encuentran las instrucciones a realizar según la información depositada en las variables. Por lo tanto aunque externamente se realiza una modificación de alguna variable, en el siguiente ciclo, el programa maestro ejecutará las instrucciones y colocará o asignará el valor que corresponda. No todas las variables son modificadas a través del

programa, existen grupos de variables que el programador deja para evaluación, las cuales podrá la interface humano-máquina modificar y sobre la base de éstas el programa maestro tomar cierta acción como si fuesen señales de entradas de alguna tarjeta.




En la parte que corresponde a la interface son creadas las variables que sé moni torearán y / o modificarán, según el tipo al que corresponda su imagen, en controlador lógico programable. Por ejemplo, en el controlador existe una tarjeta de entradas digitales, la cual tiene conectado a un borne, un sensor de proximidad. Esta entrada digital, tiene una posición booleana en memoria del controlador, de tal forma que al activarse o desactivarse el sensor, la posición de memoria tomará su respectivo valor de 0 o 1. En la interface humano-máquina se crea una variable, llamada: ***variable del proceso***, la cual es una imagen del valor depositado en la memoria del controlador. A la variable del proceso se le asigna un evento a una gráfica de la interface, la cual puede ser un bombillo, el bombillo tomará la propiedad de color rojo cuando el valor en

memoria de cotrolador sea 1, es decir cuando el sensor de posición sea activado, y tomará la propiedad de color gris cuando el valor sea 0.

Para el caso de escritura, es decir donde la variable del proceso es modificada en la interface y descargada en la memoria del controlador, por ejemplo: existe una variable del proceso en la interface, la cual es un valor numérico, que puede ser ingresado por el operador de la interface a través del teclado, sea este valor de consigna (setpoint) para un control de lazo cerrado de temperatura. El operador escribe el valor de consigna en la variable del proceso, este valor es enviado a la memoria del controlador (coma flotante), el controlador a su vez utiliza este valor para aplicarlo en la función de lazo de regulación y lograr la temperatura deseada.

El flujo de información que existe entre la memoria del controlador y las variables de la interface permiten desarrollar un grupo de servicios y opciones de monitoreo y control del proceso:

- Animación y visualización.
- Seguridad.
- Alarmas/ Eventos.
- Tendencia histórica y Tendencias de tiempo real.
- Recetarios.

*Animación y visualización:* Es la representación gráfica y animada de las variables del proceso (i.e.: la imagen de la memoria del controlador lógico programable), modificando las propiedades de los objetos existentes en la interface.

*Seguridad*: Asignación de nombres y códigos de acceso al sistema para poder gobernar el sistema.

*Alarmas y eventos:* Es asignación de eventos visuales, sobre la pantalla, a valor de alarmas correspondientes al controlador lógico programable.

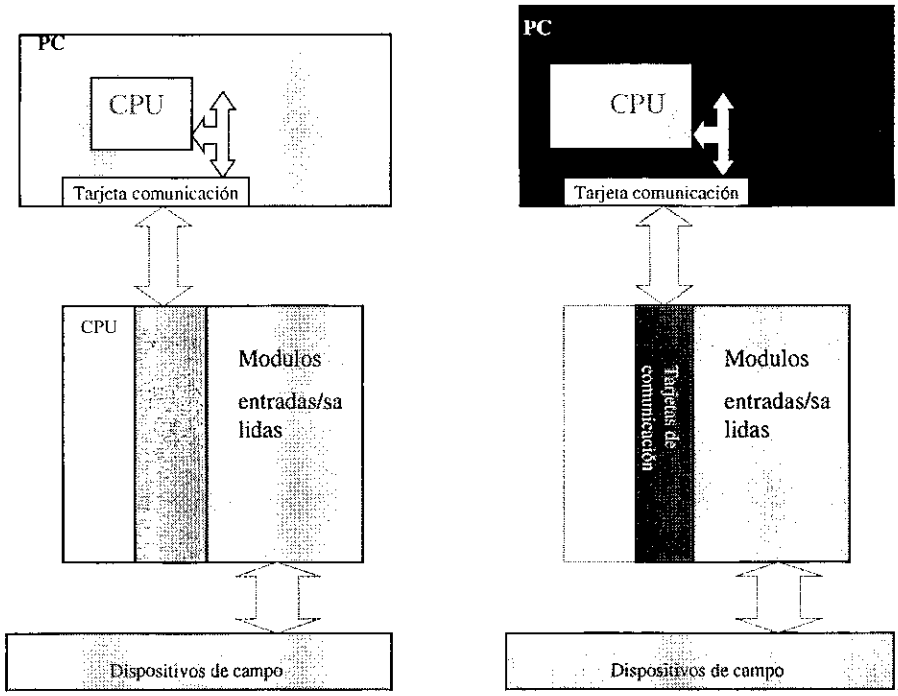
*Tendencias históricas*: Consisten en almacenar y graficar variables del proceso, las cuales es necesario tener un historial.

*Recetarios*: Conjunto de valores determinados para asignar a las variables del proceso, según la selección del operador.

(KC Automation, cap. 3)

**b. Aplicaciones y Tendencias del HMI:**

Debido al potencial desarrollado en las computadoras (tiempo de ejecución de instrucciones, capacidad de almacenaje, medios de comunicación) y al desarrollo que ha tenido el hardware en los controladores con respecto a las tarjetas de comunicación, es notorio que se utilizan dos microprocesadores, uno para las tareas de la interface y otro para las tareas del controlador. La tendencia es utilizar un único microprocesador, el cual estará localizado en la computadora de la PC y absolverá las tareas del PLC, para ello se sustituirá una tarjeta de comunicación en lugar de la CPU del PLC.




(Wonderware Factory Suite, cap 2)

# 3. Controlador lógico programable (S7- 214)

## a. Comunicación puerto:

El protocolo del sistema S7-200 se denomina interface punto a punto (PPI) y se basa en la intercomunicación de sistemas abiertos (OSI). El protocolo PPI es un protocolo maestro esclavo implementado en un bus de Token con niveles de señalización RS-485. La definición del bus Token responde al estándar de operación del bus de campo (PROFIBUS) como se define en el documento del DIN 19245.

La conexión física se realiza con un conectador D de nueve pines. La velocidad de comunicación está fijada en 9600 baudios.

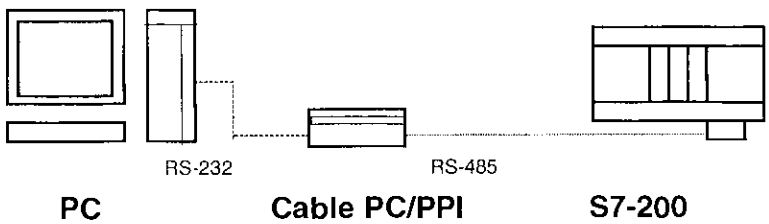
El protocolo (PPI) es un protocolo de caracteres que utiliza un carácter de bit de once, con un bit de arranque, ocho bits de datos, un bit de paridad y un bit de parada. El bloque de comunicación (a excepción del carácter simple de acuse de recibido) depende de los caracteres especiales de arranque y parada, de las estaciones de origen y destino, de la longitud del bloque y del carácter del total de control (suma de verificación) para comprobar la integridad de los datos.

La CPU 214 es una unidad esclava como lo es la CPU 212, sin embargo, cuando la CPU 214 está en modo RUN recibe un soporte Token habilitando la comunicación PPI. Una vez habilitadas las comunicaciones PPI, podrán enviar los mensajes a otro autómata programable para utilizar las operaciones de escritura, lectura de red (NETR) y escritura de red (NETW).

El protocolo PPI soporta las conexiones entre un maestro y diversos esclavos así como entre varios maestros y varios esclavos. Por ésto, el protocolo PPI proporciona un mecanismo de comunicación tanto potente como fiable para los autómatas programables S7-200.

**b. Conexión de cable PC / PPI al S7-200:**

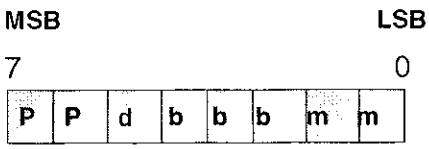
El cable PC / PPI proporciona el nivel necesario de RS-232 a RS-485 para permitir la comunicación a establecer entre computadora personal y el CPU S7-200.




Modo puerto libre (free *port*):

El SMB30 es el primer byte SM que puede leerse y modificarse. Este byte configura al puerto de comunicación *Freeport* o de sistema.

**Byte de control de Freeport**




**pp Paridad**

00 Sin paridad

01 paridad par

10 sin paridad

11 paridad impar

**d Bit de datos por carácter**

0 8 bits por carácter

1 7 bits por carácter

**bbb Tasa de baudios**

00038,400 baudios (CPU 214)

19,200 baudios (CPU 212)

00119,200 baudios

0109600 baudios

0114800 baudios

1002400 baudios

1011200 baudios

110600 baudios

111300 baudios

**mmm Protocolo**

00 Protocolo interface punto-punto

01 Protocolo de *Freeport.*

10 PPI utiliza NETW/NETR

11 Reservado (preajuste PPI)

Al elegir el modo *Freeport*, el programa KOP controla el funcionamiento de l puerto de comunicación mediante interrupciones de recepción (o también la interrupción de transmisión) y la operación de transmisión (XMT). El protocolo de comunicación entero es controlado por el programa KOP mientras se opera en *Freeport*. Otros bits determinan la tasa de baudios y la paridad. El modo *Freeport* utiliza un bit de arranque, siete u ocho bits de información y uno de parada.

Cuando el autómata se encuentra en STOP se inhibe el modo Freeport, con lo cual se restablece la comunicación normal (acceso a través de la unidad de programación)

## 4. Interface gráfico-usuario (GUI):

**Formato:**

Es una ventana de operación que utiliza la interface para depositar todos los controles que se aplicarán

**Caja de herramientas:**

Provee de un conjunto de herramientas que son usadas para diseñar y colocar los controles en el formato. Existen varios controles los cuales son: Pointer, Picture box, Label, Text box, Frame, Command button, Check box, List box, Horizontal scroll bar, vertical scroll bar, Timer, Directory list box, File list box, Shape, Line, Image, Data, Grid, OLE, Common dialog.

Label: Despliega texto en el cual el usuario no puede interactuar o modificar.

Text box: Provee un área de ingreso o despliegue de texto.

Frame: Provee un portador visual y funcional para controles.

Command button: Provee una acción o comando cuando un usuario lo selecciona..

Option button: Es una parte de un grupo de opciones, despliega multiples opciones, de la cual el usuario puede seleccionar cualquiera.

List box: despliega una lista de items que el usuario puede elegir.

Timer: Ejecuta eventos de tiempo en específicos eventos de tiempo.

Shape: Agrega rectángulos, cuadrados, elipses o círculos..

Lines: Agrega segmentos de líneas rectas a un formato.

Se ha definido únicamente las que se utilzarón para la interface, para mayor información del resto de controles ver Programmer's Guide VB, Capítulo 3.

**Utilización de List Box:**

El objeto List Box permite la presentación de una lista de opciones al usuario. La selección es presentada verticalmente en una columna simple, aunque puede configurarse de varias columnas. Si el número de elementos excede el tamaño seleccionado, se agregará una barra de desplazamiento vertical automáticamente. El usuasario puede desplazarse hacia arriba y hacia abajo.

Para agregar elementos al objeto *List Box*,se utiliza la opción del método ***AddItem***, el cual tiene la siguiente sintaxis:

*Box. AddItem item[,index].*

El nombre de la lista está definido por Box, el grupo de caracteres a ser insertados está definido por *item* y la posición donde se desea insertar esta definido por *index*.

Para borrar un elemento de la lista se utilizarán los métodos ***RemoveItem*** y ***Clear***. El primero utiliza el argumento index para especificar cuál elemento será borrado. El segundo borrar todos los elementos de la lista.

La sintaxis es la siguiente:

*Box.* ***RemoveItem*** *Index*

*Box.* ***Clear***

(VB Programer's Guide, Cap 3,49)

**Archivos de proyecto**:

Existe un archivo con extensión .MAK, el cual contiene la lista de los archivos que se utilizan en el proyecto, adicionalmente donde están localizados dichos archivos.

Existen tres tipos de archivos: formas, módulos y controles.

Las formas tienen la extensión .FRM, estas contiene la descripción gráfica de los objetos incluyendo las propiedades. Adicionalmente contienen declaración de tipos, constantes, variables y procedimientos externos (subrutinas de manipulación de eventos).

Los módulos con extensión .BAS, contienen declaraciones locales o globales de tipos, constantes, variables, procedimientos externos y procedimientos globales.

Los controles que tienen la extensión .VBX, contiene la información básica para proveer un nuevo control en la barra de herramientas.

(VB Programer's Guide, Cap. 4,113)

La aplicación contiene Formas, Módulos y Controles. Las formas contienen los elementos visuales de una forma, incluyendo todos los controles de la forma y el código asociado con las formas. Los módulos contienen básicamente código.

Los módulos y formas pueden contener: Declaraciones, Procedimientos de eventos, Procedimientos generales. Los procedimientos de eventos son los procedimientos llamados: ***Sub*** los cuales son ejecutados en respuesta a un evento del usuario o del sistema. Los procedimientos de eventos ocurren únicamente en las formas.

Existen procedimientos que no están directamente asociados a un evento.

Los procedimientos general esen una forma son localmente ejecutados, y

no pueden ser invocados desde otro forma. Todos los procedimientos en un modulo son procedimientos generales y ellos pueden ser invocados desde cualquier modulo en aplicación.

**Variables**

Es necesario almacenar valores temporales de información cuando se realizan cálculos, se pueden declarar variables con el código ***Dim***:

***Dim*** *variablename.* ***As*** *Type*

Existen declaraciones implícitas las cuales no necesitan tener declaración antes de ser utilizadas. Para evitar problemas de pérdida de nombres de variables es aconsejable declarar formalmente las variables.

Las variables pueden ser locales, de nivel o globales. Las variables locales están reconocidas en el procedimiento en el que aparecieron. Esta es una buena elección para cualquier tipo de variable temporal. Las variables de nivel combinan información con todos los procedimientos existentes en una forma o modulo. Por último las variables globales que tienen total cobertura. Los valores en las variables son accesibles en cualquier procedimiento, formula o modulo de la aplicación. La declaración de las variables locales se utiliza código **Global**. Las variables globales pueden ser declaradas en cualquier modulo. Para el propósito de la organización del código, es aconsejable que dicha declaración sea confinada a un número de módulos que sean fáciles de ser accesados.

(Programer's Guide, Cap.6)

Los arreglos (*Arrays*) permiten referir a una serie de variables con el mismo nombre y únicamente seleccionando específicamente el elemento a través de un número índice que indica cual apartado se refiere.
La sintaxis para utilizar es la siguiente:

***Global [Dim]*** *Variable(number)* ***As*** *Type*

Algunas veces puede no conocerse exactamente que tan largo será el arreglo. Por lo tanto los arreglos tienen la capacidad de cambiar el tamaño durante la ejecución de la aplicación. A este tipo de arreglos son llamados: Arreglos dinámicos, los cuales pueden ser redimensionados en cualquier momento. Para permitir la característica de redimensionamiento se utiliza la instrucción:

***ReDim*** *Array (Newsize)*

Cuando se utiliza la instrucción *ReDim* todos los valores almacenados en el arreglo son perdidos. Esto es útil cuando se desea preparar el arreglo para un grupo nuevo de datos. Sin embargo puede hacerse el cambio de tamaño sin perder los datos en el arreglo, para ello se utiliza la instrucción *ReDim* con la palabra clave ***Preserve.***

(Programmer's Guide, Cap 7)

**Acceso a bases de datos**

Con los controles de datos, pueden crear aplicaciones para desplegar, editar, y actualizar la información de distintos tipos de bases de datos existentes. VB implementa acceso a las bases datos incorporando el mismo tipo de base de datos utilizada por Access. Para dicho acceso debe especificarse el nombre de la base de datos que desea ser abierta. Esto puede ser echo en el desarrollo de la aplicación.

Las propiedades que maneja el control datos son: *Connect, DatabaseName, Exclusive, Options, ReadOnly, RecordSource.*

Utilizando el control datos, se puede conectar a la aplicación de VB a una archivo de base de datos para revisar un grupo de registros. El control de datos puede realizar las siguientes tareas sin necesidad de utilizar código de instrucciones:

- Conectarse a la base de datos locales y remotos.
- Abrir una tabla de base de datos o definir un conjunto de registros basados en SQL de tablas.
- Trasladar campos de datos cuando se desplaza o cambia valores.
- Actualizar una base datos basado en cualquier cambio que realiza a para desplegar datos en los controles delimitadores.
- Cerrar base de datos.

Los controles de datos utilizan los métodos **UpdateRecord**, actualiza el grupo de registros especificados con los controles delimitadores.

**Control de comunicación:**

El control de comunicación provee comunicación serial a la aplicación, permitiendo la transmisión y recepción vía el puerto serial. El control de comunicación provee las siguientes dos opciones:

- Evento-acción: es un método poderoso para manipular el puerto serial. En varias situaciones deberá ser notificada el momento en el que toma lugar cuando un carácter arriba o un cambio ocurre el la portadora (CD) o un requerimiento para enviar (RTS). En tal caso, se deberá utilizar *OnComm event*, adicionalmente controlan los problemas de error.
- *Poleo*: se realiza poleo de eventos y errores a través de chequear el valor del CommEvent después de cada función crítica en el programa.

*Propiedades*:

*CommPort:* Asigna y retorna el número de puerto de comunicación.

*Settings:* Asigna y retorna la velocidad en bauds, paridad, bit de datos.

*PortOpen:* Asigna y retorna el estado de comunicación del puerto. Adicionalmente habilita o inhabilita un puerto.

*Input:* Regresa y remueve los caracteres que han sido recibidos en el buffer.
*Output*: Escribre una cadena de strings al buffer de transmisión.
*CommEvent:* Regresa el error más reciente de evento de comunicación. Esta propiedad no esta disponible en el tiempo de diseño y únicamente es de lectura en el tiempo de desempeño.
Aunque el evento *OnComm* es generado en cualquier error de comunicación o evento, este evento mantiene el código numérico para cual error o evento desempeño.

| Valor | Descripción |
|---|---|
| 1001 | Una ruptura de señal fue detectada |
| 1007 | Tiempo expirado de *Carrier Detect*. El CD fue abajo para *CDTimeOut* |
| 1002 | Tiempo expirado para *Clear to Send*. |
| 1003 | Tiempo expirado para *Data Set Ready* |
| 1004 | Error en la trama. El *hardware* ha detectado error. |
| 1006 | Port overrun |
| 1008 | Receive Buffer overflow |
| 1009 | Parity error |
| 1010 | Transmit buffer full |
| | |

*InBufferCount:* Retorna el número de caracteres esperando a ser leídos del buffer de recepción. Esta propiedad no es disponible en tiempo de diseño.
InBufferCount refiere al número de caracteres que tiene recibidos por el módem y están esperando en el buffer de recepción para que sea retirados. El InBufferCount puede ser limpiado asignándole el valor de cero a la variable.
*Input:* Regresa y remueve la cadena de caracteres del buffer de recepción.

*InputLen:* Asigna y retorna el número de caracteres que serán leídos del buffer de recepción. El valor de inicio es cero. Al tener este valor asignado el control de comunicación lee el contenido entero del buffer recibido cuando la propiedad *Input,* es utilizada.

(Professional Features Book1Cap.5.)

# II. Objetivos.

## A. Objetivos generales:

1) Crear una interfase humano-máquina como una herramienta moderna para el control de equipo industriales.

2) Proporcionar una herramienta fácil de aprendizaje y manejo para operadores o usuarios del equipo y así mejoras la eficiencia y productividad.

3) Proporcionar una solución económica favorable para la pequeña empresa en el campo de la automatización industrial.

4) Crear el uso computadora personal como interfase visual para descargar de tareas de visualización a los controladores lógicos programables.

5) Motivar a los estudiantes por la investigación en el área de automatización industrial que brinde una solución acorde a las necesidades de nuestro país.

6) Mediante el uso de una interfase humano máquina se pretenderá que el operador tenga una relación más directa con la maquinaria que esta utilizando.

7) Permitirá reducir la cantidad de accesorios físicos de manipulación del equipo, tal es el caso de botones, led, bombillos, desplegadores de mensaje, manijas, selectores, potenciómetros.

8) La reducción de accesorios físicos de manipulación del equipo permite reducir el mantenimiento preventivo o correctivo sobre alguna pieza de las anteriores mencionadas que sufriere daños.

9) La reducción de accesorios físicos de campo permite reducir cableado hacia el controlador lógico programable y a su vez menor cantidad de módulos de entrada y salida digitales y analógicas.

10) El operador tendrá al alcance visual mediante una pantalla, teclado y un ratón (*mouse*) el control de la maquinaria sin necesidad de desplazarse sobre todo el piso para poder echar a funcionar su equipo, únicamente debe cambiar de pantallas y desplazarse virtualmente por todo su equipo.

11) El tener una interfase humano-máquina permitirá reducir tiempo en capacitación del personal que operar el equipo.

12) La interfase humano-máquina permitirá brindar seguridad de acceso al control de la máquina, evitando que cualquier persona sin la debida autorización manipule el equipo.

## B. Objetivos específicos:

1) Crear una interfase humano-máquina utilizando las ventajas que brinda Visual Basic con la característica de interfase gráfica de usuario.

2) Se creará una ventana principal la cual permitirá al operador tener el acceso de todas las opciones del sistema en forma ergonómica.

3) Brindar seguridad al sistema utilizando nombre y contraseña para poder tener acceso al programa.

4) Existirá una base de datos con el nombre, contraseña y nivel de acceso para poder ejercer control del sistema.

5) La pantalla principal proveerá al operador de la visualización de la mesa de estación de trabajo y de los gabinetes de destino, de tal manera que para definir una tarea (*job*), únicamente deberá recoger la paleta de la estación de trabajo y "arrástrala" al destino (drag *and drop*).

6) Existirá una ventana que dará información del estado de las tareas generadas, la hora, prioridad y persona que generó dicha tarea.

7) Proporcionar una animación visual de los movimientos del montacargas a través de su recorrido, desde la estación de trabajo hasta los gabinetes de destino y utilizar para ello los sensores que

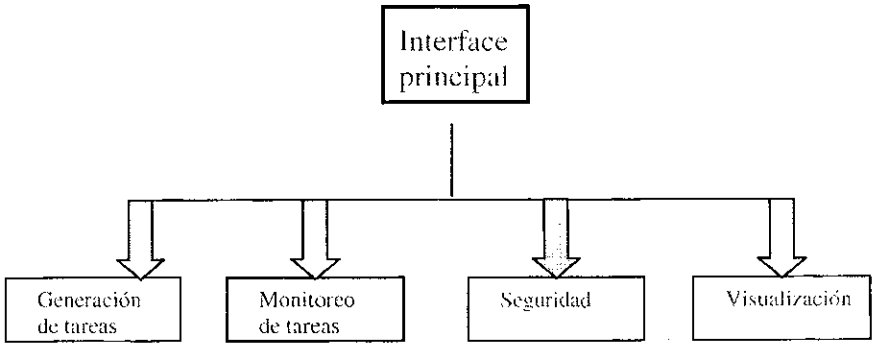
son usados para controlar la dirección del equipo. Esta información será proporcionada por el PLC a la interface.

8) La comunicación entre el PLC y la interface será establecida por medio la interface RS 232 C y configurando el puerto del controlador en modo *Freeport.*

# VI. METODOLOGIA DEL TRABAJO

La interface human-máquina es una herramienta orientada a un mejor desempeño entre el operador y la maquinaria, provee de una ruta sencilla y al alcance visual de las operaciones.

A continuación se presenta un diagrama esquemático de las ventanas de trabajo que utiliza el operador para trabajar.




**A. Interface Gráfica del Usuario:**

*Variables manejadas por el operador:*

El operador manejará variables sencillas relacionadas con los eventos cotidianos del equipo. Trabajando principalmente con el evento: **tarea**.

La tarea consiste de una serie de acciones que debe tomar el montacargas para llevar un producto (*pallete*) de un punto especificó de origen a otro de destino, los cuales son seleccionados por el usuario; cada tarea esta compuesta por un grupo de información, los cuales son:

- Fuente.
- Destino.
- Prioridad.
- Estado.

- Fecha.
- Nombre del operador.

A continuación se definirá un grupo de conceptos que permitirá una mejor comprensión del sistema.

*Paleta*: Elemento físico sobre el cual se deposita el producto para ser almacenado y / o transportado.

*Fuente:* Es la dirección lógica asignada a un punto específico de la estantería o la mesa de trabajo, en el cual deberá ser recogida la paleta.

*Destino*: Es la dirección lógica asignada a un punto específico de la estantería o mesa de trabajo, en el cual deberá depositarse la paleta.

*Prioridad:* Es el grado de importancia que se le asigna a una tarea a desarrollar (evaluada de 0 a 5, 0 baja prioridad y 5 máxima prioridad).

*Estado:* Es la variable que indica el desarrollo de una tarea. Una tarea puede estar clasificada en: generada, delegada, ejecutándose o en desarrollo y finalizada. La tarea generada es aquella donde se ha definido la fuente y el destino y ha sido cargada a la memoria de la interface. La tarea delegada es aquella que después de ser generada es enviada al controlador lógico programable y ha sido cargada en la cola de tareas de dicho controlador. La tarea en desarrollo es aquella que está en la cola del controlador y que esta en la rutina de ejecución. La tarea finalizada es aquella que ha tomado la paleta del punto de origen y la ha depositado en el destino sin ningún problema.

*Fecha*: Esta variable almacena la fecha y hora en que fue generada la tarea.

*Operador*: Esta variable almacena el nombre del operador al que fue permitido el acceso al sistema y generó la tarea.

**Eventos generados por el operador:**

El operador para poder iniciar con su tarea cotidiana, deberá realizar dos eventos indispensables al inicio del proceso, estos son:

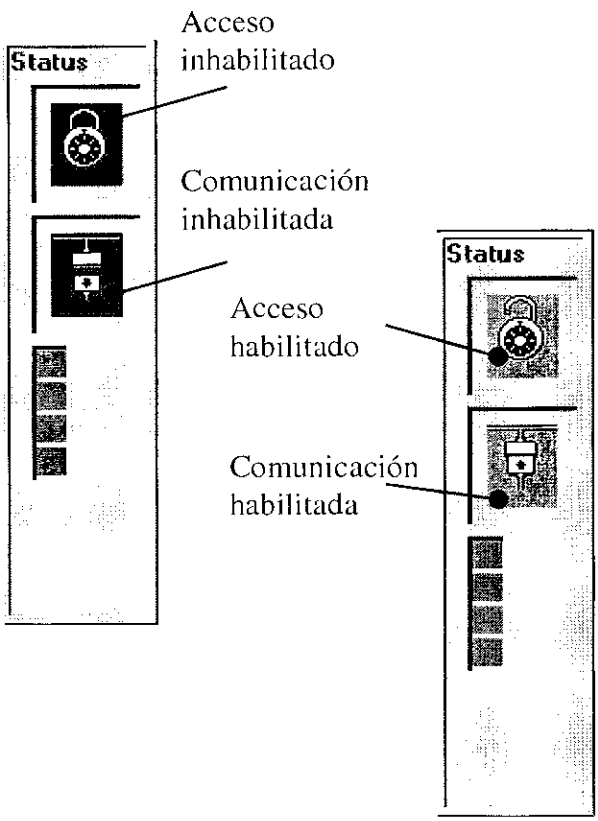
1. Acceso al sistema.
2. Establecimiento de comunicación.

*Acceso al sistema:* Como medida de seguridad al sistema, para poder manipular la interface, el operador deberá ingresar el nombre asignado y una palabra de acceso para habilitar las diferentes opciones de la interface. Debiendo para ello presionar el icono de acceso de sistema (el cual esta representado por una llave), localizados en el campo de control.

Si el nombre y la palabra de acceso es la correcta, en el campo de estado una figura con un candado abierto en un fondo verde será desplegado, en caso contrario permanecerá la figura del candado cerrado en fondo rojo.

*Comunicación:* El operador deberá establecer comunicación con el controlador lógico programable, para ello deberá presionar con el ratón el icono de conexión (el cual esta representado por un par de cables conectados), localizados en el campo de control. Al ser presionado el icono un paquete de información será enviado al controlador solicitando conexión con la interface, por su parte el controlador al recibir un requerimiento de conexión, responderá con el paquete de información, indicando a la interface que la comunicación esta habilitada. En la interface será desplegada una ventana de información indicando que la comunicación ha sido establecida y una figura de un conector en fondo verde será sustituida. En el caso que no exista respuesta la interface

permanecerá desconectada y la figura del conector desconectado en un fondo rojo permanecerá.

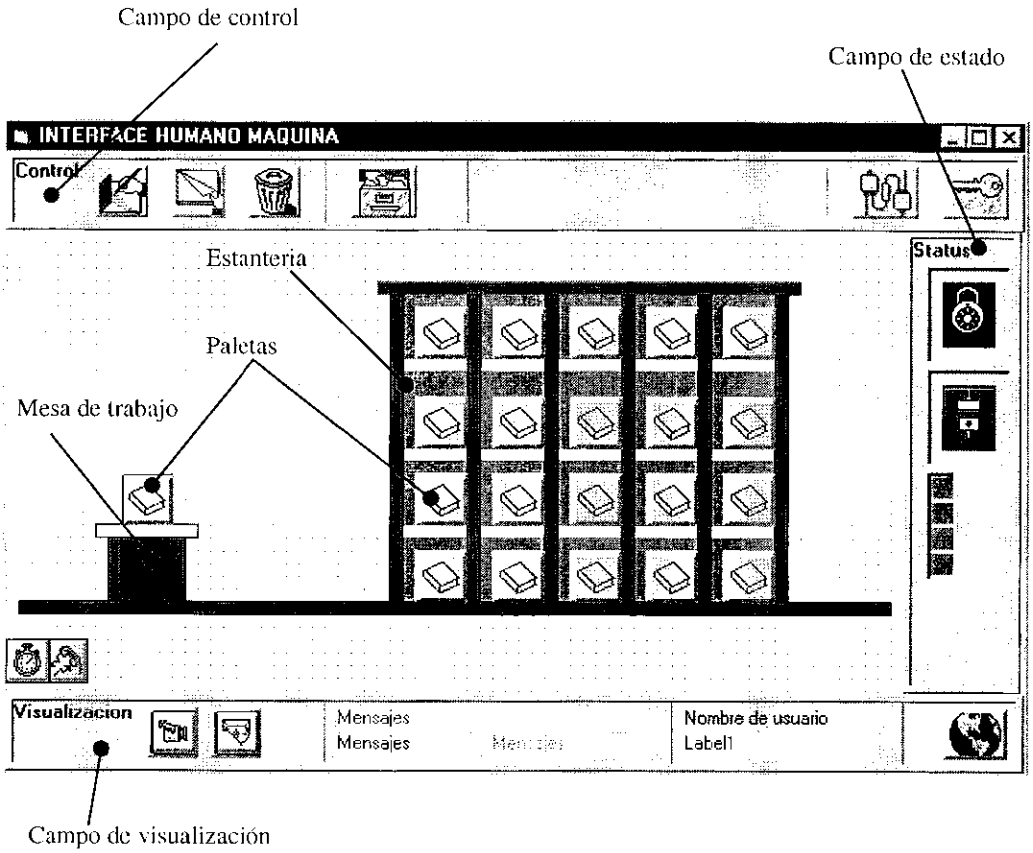



Los eventos subsiguientes dependerán de las decisiones del operador para el manejo de la interface, estas son:

I. Generación de tareas.
II. Lista de tareas.
III. Copia de respaldo del listado de tareas.
IV. Visualización de tarea en desarrollo.

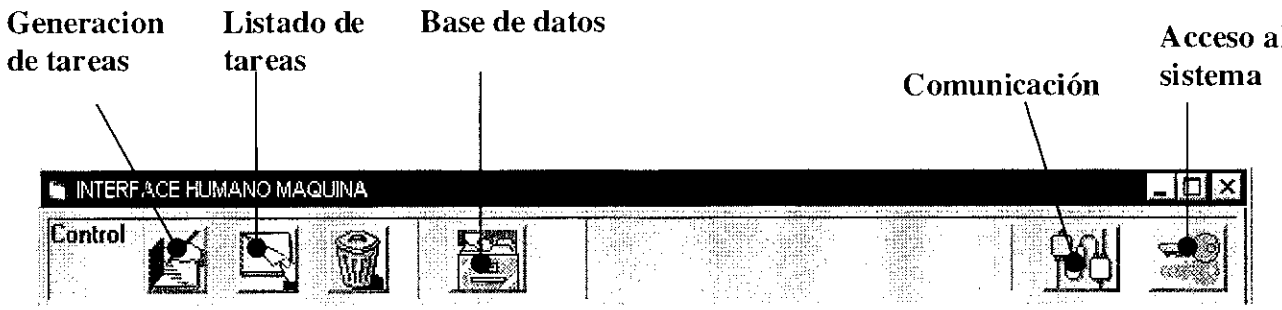
*Generación de tareas*: Este evento consiste en asignar una tarea específica a realizar por el montacargas. Esto significa asignar una fuente, y destino de la tarea adicionalmente la interface le agrega la fecha y el nombre del operador cargado en el sistema. El operador tiene dos formas de realizar dicho evento. La primera consiste en utilizar la opción *drag and drop*, la cual deberá ser posicionado el apuntador del ratón en la casilla deseada y se presiona *enter* sostenido, luego se desplazará a la casilla destino soltando el *enter*, existiendo una representación gráfica de la mesa de trabajo y la estantería, luego de soltar el enter aparecerá una ventana en la cual están los datos de la tarea generada, pudiendo el operador modificar la prioridad y confirmar la tarea generada. Tiene opción de aceptar o denegar la tarea generada.

La segunda forma es a través de presionar el icono de generación de tareas, el cual está localizado en el campo de control, representado por la figura de una libreta de escritura. Luego de presionar el icono, aparecerá la ventana de generación de tareas, en la cual el operador de una forma manual (utilizando el teclado) asignará la fuente, destino y prioridad.




*Listado de tareas*: Este evento permite desplegar todas las tareas que están generadas, delegadas, en desarrollo y finalizadas. Para poder accesar esta ventana el operador deberá presionar el icono de lista de tareas, localizado en el campo de control y representado por una libreta de lectura. La ventana de listado de tareas presentará dos campos, el primer campo, llamado de tareas, contiene seis columnas, las cuales son: fuente, destino, prioridad, usuario, estado, fecha, Y el segundo campo, llamado de control, contiene los iconos para delegar una tarea al controlador o para borrar del paquete. Debe hacerse la salvedad que únicamente las tareas con el estado: "generado", podrán ser enviadas al controlador, en caso contrario la interface no permitirá que se envíe al controlador; de similar

forma únicamente las tareas con los estados: "generada" o "finalizada" podrán ser borradas de la lista.




*Copia y respaldo de la lista:* Este evento permite llevar un registro de respaldo (base de datos) de las tareas que han sido cargas en la lista. Permitiendo tres opciones para su respectiva administración. Para ello se presiona el icono localizado en el campo de control y representado por la figura de un archivo. La primera opción corresponde a borrar datos, esto permite borrar todas las tareas almacenadas en la base de datos. La segunda opción permite tomar las tareas existentes en la lista y agregarlas a las tareas grabadas con anterioridad en la base de datos. La tercera permite extraer las tareas almacenadas en la base de datos y hacer una copia a la lista de tareas.

*Visualización de tarea en desarrollo:* Este evento habilitado por el operador pero generado por el controlador nos permite mostrar en una forma animada el desplazamiento del montacargas durante su ruta al ejecutar una tarea específica. Para ello se presionó el icono localizado en el campo de visualización, representado por una cámara.

**Desarrollo de la interfase:**

El primer paso para crear una aplicación con VB es crear la interface: los formatos, los controles. En las aplicaciones o procedimientos tradicionales, un evento para un control específico era una porción de código a ejecutar. La ejecución arrancaba en la primera línea del código ejecutable y continuo un camino definido por la aplicación, llamando procedimientos o como fuese necesario.

Un evento es una acción reconocida por un formato o control. La aplicación de tipo evento-acción ejecuta un código básico en respuesta a un evento, cada forma y control tiene definido un conjunto de eventos. Si uno de estos eventos ocurre, se ejecutará el código que está asociado a este evento.

Muchos objetos reconocen similares eventos, aunque diferentes objetos pueden ejecutar diferentes eventos.

Para comprender qué es lo que sucede con una aplicación típica de evento-acción, son desarrollados cuatro pasos:

1. La aplicación inicia automáticamente carga y despliega el formato de arranque.
2. Un formato o control recibe un evento. El evento puede ser causado por el usuario (por ejemplo la presión de una tecla) o por el sistema (evento de un temporizador).
3. Si existe un procedimiento correspondiente a tal evento, este es ejecutado.
4. La aplicación espera por el siguiente evento.

En los programas del tipo evento-acción, la acción del usuario o evento del sistema ejecuta un procedimiento de eventos. Por lo tanto, el orden en

el cual el código se ejecuta depende en cual evento ocurre, en nuestro caso dependerá de lo que realice el usuario. Esta es la esencia de la interface gráfica de usuario y de la programación evento-acción, el usuario esta con la decisión y el código es el que responde. Por lo tanto no se puede predecir lo que el usuario realizará, el código debe hacer ciertas suposiciones del estado exterior cuando se este ejecutando.

El programa VB provee las herramientas apropiadas para los diferentes aspectos de desarrollo de una interface gráfico de usuario. Pueden crearse interfaces gráficas para la aplicación a través de dibujar objetos de una forma gráfica. Se asignan las propiedades de los objetos a utilizar para afinar la apariencia y el comportamiento. Permitiendo de esta forma que la interface reaccione según el código escrito, el cual responderá a eventos que ocurran en la interface.

Para desarrollar una interface deben ser dibujados los objetos, los cuales podrán ser*: text boxes y command bottons* sobre una ventana llamada *form*. Una vez creados los controles y la forma son parametrizadas las propiedades para que la forma y los controles tengan el color, tamaño deseados. Finalmente se escribe el código para dar animación

Cuando se crea un objeto (forma o control), VB asigna a este un nombre y propiedades

Existen tres pasos principales para crear una aplicación:

1. Crear la interface
2. Parametrizar las propiedades
3. Escribir el código.

**Rutina de arranque e inicialización:**

Esta rutina es la que permite que la interface genere todas las variables necesarias del sistema para poder interactuar con el operador. Esta está conformada de dos estructuras básicas:

1. Definición de variables.
2. Inicialización de las variables.

*Definición de variables:*

Aquí se definen las variables que utilizará el sistema para la administración de información que establecerá la comunicación operador-interface e interface-controlador.

En la primera estructura (ver 7.3.1.1) se declara un tipo de variable *Tarea*, la cual está conformada por los campos: *fuente, destino, prioridad, status, fecha y usuario.*

Luego de ello, se define la variable *HMIBaseDato* que permite manejar la base de datos de respaldo de las tareas.

Se define la variable *TareaGenerada* la cual almacenará la tarea que el operario desea ingresar en la lista de tareas, cual está definida con el nombre de: *ListaTareas.*

Las variables *Cantidad* y *Puntero* son utilizadas para el manejo de la lista de tareas, tal es el caso de lecturas, escrituras y/o modificación de dicha lista.

Las variables *IngresoUsuario* y *NivelUsuario* permiten administrar la seguridad del sistema.

Las variables: *Comunica, Cabeza, Cword, Fuen, Des, Cola* son utilizadas para la comunicación entre la interface y el controlador.

La variable Carro, del tipo Posición, es utilizada para la visualización del desplazamiento del montacargas al ejecutar una tarea.

*Inicialización de variables:*

La variable (ver 7.3.1.2) *Comunica* es utilizada para evaluar el estado de la comunicación entre la interface y el controlador, por lo cual 0 indica comunicación inactiva y 1 comunicación activada.

La lista de tareas es dimensionada a una tarea, la cual es generada por el sistema.

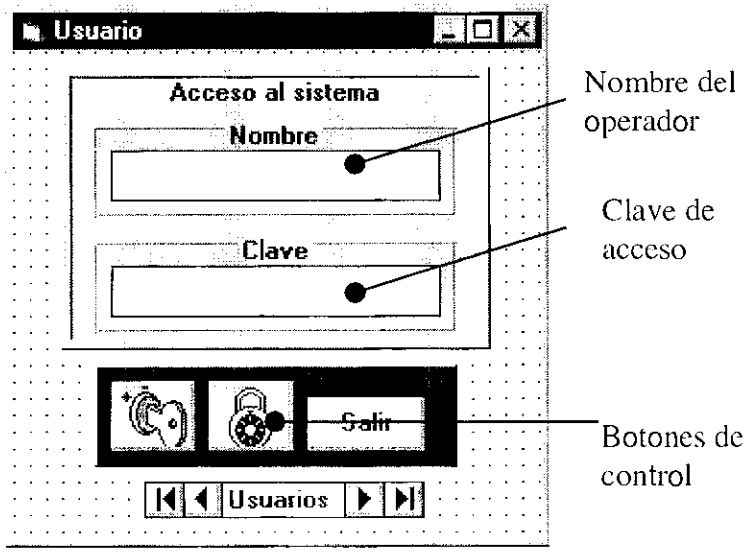
Para poder permitir la seguridad al sistema, se inicializa la variable del operador por *Ninguno*, la cual permite informarle a la interface bloqueo de cualquier operación hasta que no sea ingresado el nombre y palabra de seguridad al sistema.

Los botones de cada función del sistema son inhabilitados.

Para la configuración del puerto serial, lo primero que debe hacerse es definir con qué puerto se trabajará, para ello se utiliza la variable *Comm1.CommPort*. Para configurar el puerto a 9600 bps, 8 bits de datos, 1 bit de paro, sin paridad, se utiliza el *Comm1.Settings*. Finalmente para está habilitar el puerto y así este listo para transmitir y / o recibir información se utilizando la propiedad *Comm1.PortOpen*.

## 1. Acceso al sistema:

Este conjunto de subrutinas permiten habilitar el uso del sistema a un



operador. Para ello el operador deberá presionar el botón de acceso al sistema. Internamente al ser presionado dicho botón es llamada una subrutina, *AccesoUsuario_Click* (7.3.1.3) la cual se encarga de inhabilitar la pantalla principal, cargar la ventana de acceso al sistema y presentarla en la pantalla. Inmediatamente aparecerá la pantalla de Acceso, la cual está conformada por dos campos, el primero es el de ingreso de datos y el segundo es aceptación de datos. Al ser ingresados los datos correspondientes correctamente y presionar el botón de aceptación representado por la figura de una llave, es cargada la subrutina *Ingresar_Click ()*.

Esta subrutina está compuesta de tres partes esenciales:

1. Inhabilitación de acceso
2. Verificación de datos ingresados.
3. Retorno a pantalla principal.

*Inhabilitación de acceso*: esta parte de la rutina se encarga de reducir el nivel de acceso al mínimo y colocar como operador del sistema a "Ninguno", y confirmando visualmente como acceso denegado, para así pasar a la siguiente etapa que es la verificación de datos ingresados.

*Verificación de datos ingresados:* esta sección de la rutina verifica el nombre del operador y clave ingresada en la base de datos del sistema, que realiza para ello un lazo de búsqueda y comparación. Si los valores son iguales entonces serán habilitadas las opciones del sistema y visualmente será desplegado el mensaje de "sistema accesado" y el nombre del operador en el campo de visualización, adicionalmente, en el campo de estado será visualizada la figura de candado abierto en fondo blanco y utiliza para ello la propiedad de visibilidad y texto de mensaje.

*Retorno a pantalla principal:* utilizando las propiedades de mostrar y habilitación es cargada la pantalla principal del sistema.

En el campo de control también se encuentra el botón de liberar; al ser presionado será ejecutada la rutina *Liberar_Click()*, la cual consultará al operador la confirmación de la acción a ejecutar que utiliza la instrucción *Msgbox,* en el caso de ser valedera la respuesta, la rutina se encarga de deshabilitar los botones de la pantalla principal, reducir el nivel del usuario a cero y colocar como operador a "Ninguno".

## 2. Generación de tareas:

### Generación de tareas vía animada:

La generación de tareas (ver apéndice 7.3.1.4) en forma animada es el evento que realiza el operador y es en el cual de una forma amigable y sencilla el operador genera una tarea. El operador únicamente debe localizarse en la pantalla principal, en la cual se presenta la mesa de trabajo y la estantería con sus respectivas divisiones. Deberá desplazar el puntero del ratón a la casilla de origen y presionará intro sostenido, luego se desplazará hacia la casilla destino deseada y soltará el botón de intro.

Para realizar esta sencilla operación conocida como "*drag-drop*", se define un objeto de VB llamado *SSCommand,* el cual puede utilizar la propiedad de índice para generar varios de ellos con el mismo nombre y propiedades pero identificándose uno de otro a través de un índice, este objeto para nuestra aplicación como: *Mercadería;* ya definido el objeto se utilizó el procedimiento "*drag-drop*", el cual utiliza la propiedad de índice, el cual define que objeto fue la fuente. Con ello es establecida la misma y utilizado el índice donde se soltó el puntero, permite seguir la ruta de la

tarea. Luego de definir la ruta de la tarea, son asignados los campos de nombre del usuario y fecha. Después es inhabilitada la pantalla principal, es cargada la pantalla *GeneraTarea*, la cual tiene por fin confirmar la información de la tarea generada. La misma será explicada en la siguiente sección.

**Generación de tarea de forma manual:**

Este evento permite al operador ingresar los datos necesarios para una tarea y agregarla a la lista de tareas. El operador debe presionar el botón *SSCommand,* de nombre *GeneraTarea,* el cual utiliza el procedimiento *Click,* cargando la rutina *Genera_Click ().* Este procedimiento inhabilita la pantalla principal y es cargada la ventana *GeneraTarea.*

Esta ventana es compuesta de dos campos: Tarea y Control. En el primero se encuentra la información de la tarea generada, pero aun no insertada en la lista de tareas, el operador podrá modificar los datos: fuente, destino y prioridad. Esto se realiza utilizando objetos llamados: *Textbox.* Al ser ingresados y / o modificados los datos, el operador tendrá dos opciones:

1. Confirmar la información de la tarea.
2. Desechar la tarea.

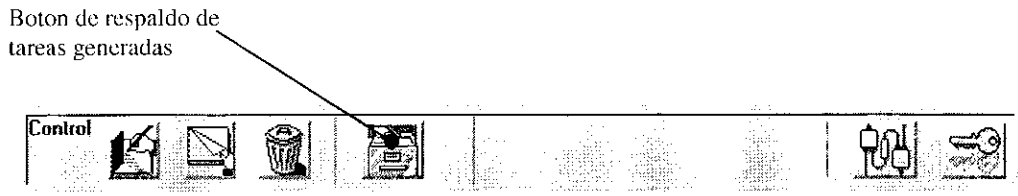
*Confirmación de la tarea:* esta será ejecutada cuando el operador presione el botón de *Aceptar,* el cual es un objeto de tipo *SSCommand.*

Al ser presionado se ejecutará el procedimiento *Click,* que llama a la rutina: *Aceptar_Click()* (ver 7.3.1.4). Esta rutina se encarga de incrementar en uno la variable que indica la cantidad de tareas cargas a la lista, luego

dimensionará la lista de tareas con un dato más, pero indica que deberá preservar los datos que con anterioridad se encontraban, para ello se utiliza la instrucción *ReDim*, después es asignado cada dato que el operador ingresó a su respectivo campo. Al terminar esta parte, la pantalla es descargada y es habilitada nuevamente la pantalla principal.

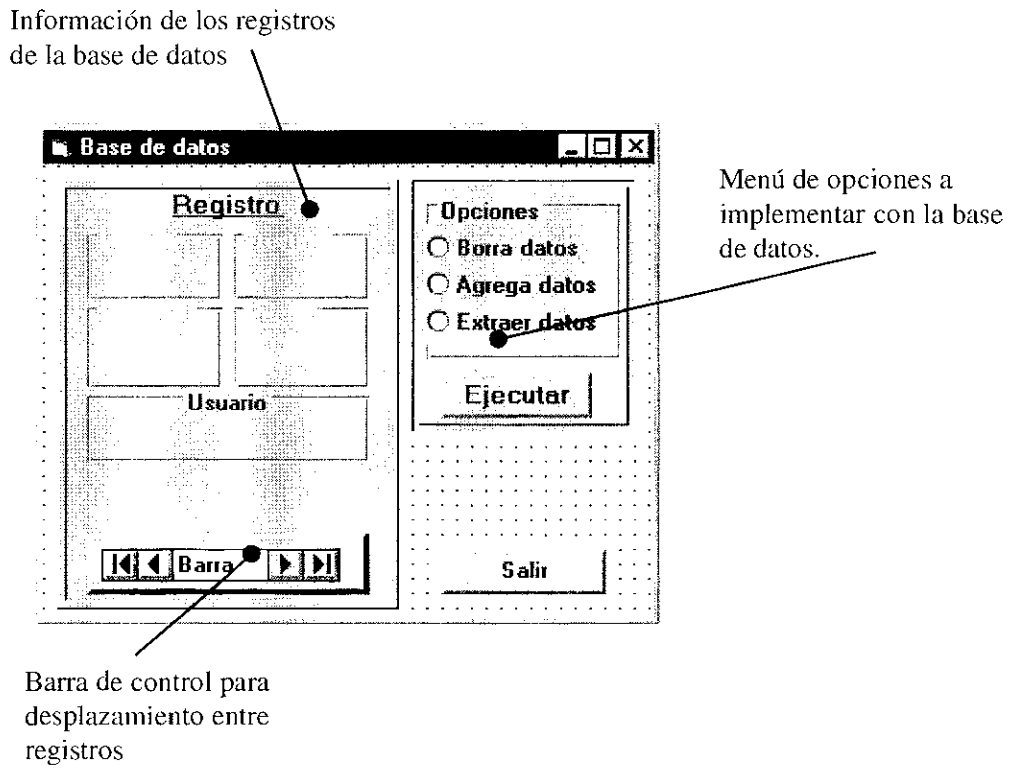
*Deshacer tarea:* esta es ejecutada cuando el operador presione, el botón *Denegar*, un objeto de tipo *SSCommand*, el cual únicamente ejecuta la rutina *Negar_Click ()*, la cual descarga la ventana GeneraTarea y habilita la pantalla principal, sin alterar ningún dato de la lista de tareas.

### 3. Respaldo del listado de tareas:




Esta opción le permite al operador (ver apéndice 7.3.1.5) tener una copia de respaldo en el disco duro, un archivo para un formato de base datos de *Access*.

Para accesar esta opción, el operador deberá presionar el botón de respaldo, un objeto de tipo SSCommand llamado *CopiaDB*, al presionar el botón se ejecutará la rutina *CopiaDB_Click()*, la cual se encarga de inhabilitar la ventana principal y cargar la ventana: Base de datos. Esta ventana contiene dos campos: Registros y Control.




El campo de registros está compuesto por cinco objetos de tipo *Label*, utilizados para desplegar cualquier tipo de información, que en nuestro caso será la información de cada registro almacenada en la base de datos. Adicionalmente está constituido por un objeto de tipo *Data*, asignándole el nombre *DB,* este control tiene una propiedad llamada: *Databasename,* la cual contiene la ruta y el nombre del archivo donde está localizada la base de datos (*HMI.DB*),

Para poder accesar los registros de la base de datos a través de los objetos tipo Label, se deben configurar ciertas propiedades esenciales, las cuales son:

1. *DataSource:* la cual la dirección (directorio) en que se encuentra la base de datos, y

2. *Datafield:* la cual especifica el campo al que hará referencia.

Por lo tanto al presionar el objeto Data (fechas de izquierda o derecha), se desplazará de un registro a otro en la base de datos que despliega los objetos tipo *label.*

El campo de control permite la administración de la base de datos y su interrelación con la lista de tareas. Esta administración está relacionada con el respaldo que provee la base de datos a la lista de tareas:

1. Borrar datos.
2. Agregar datos.
3. Extraer datos.

Para realizar estas opciones de selección exclusiva, se utilizan tres objetos de tipo *SSOption,* los cuales al ser seleccionados activan una propiedad, *SSOption.Value,* que adquiren un valor booleano de verdadero. Pero al colocar tres objetos tendríamos ocho combinaciones de ello y como se desea que sea exclusivo, son colocados en un objeto de tipo *SSFrame,* que nos brinda la exclusividad en la selección. Al ser seleccionada la opción a ejecutar, el operador deberá presionar el botón de tipo *SSCommand*, el cual ejecutará la subrutina *Command3D1_Click()*, esta compara cual opción deberá ejecutarse.

*Borrar datos:* Esta opción permite (ver 7.3.1.5) borrar todos los registros al ser seleccionado el objeto *SSFrame*, de nombre *BorrarDatos* . Al ser verdadera la propiedad *BorrarDatos.Value,* se pide una reconfirmación de la rutina a ejecutar a través de *MsgBox*, si la opción resulta valedera, es decir Respuesta = 6, entonces la rutina desplaza el puntero de la base de

datos, DB, al primer registro, que utiliza la instrucción *DB.Recordset.MoveFirst*, seguido de este desplazamiento realiza un lazo de ejecución siempre que aun existan registros en la base de datos, desplazándose al próximo registro con la instrucción *DB.Recordset.MoveNext*, y luego borrándolo con la instrucción *DB.Recordset.Delete.*

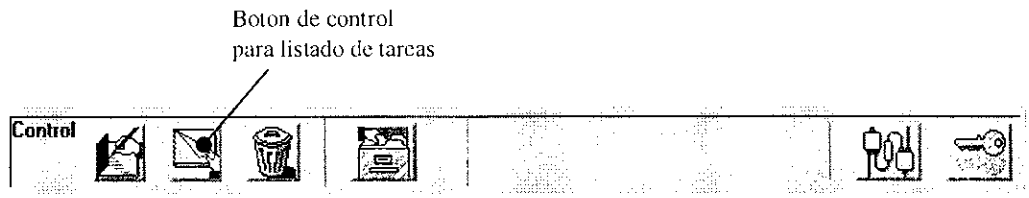
*Agregar datos:* Esta opción permite agregar las tareas existentes en la lista a la base de datos DB. Al ser seleccionado el objeto *SSFrame*, de nombre *BorrarDatos,* la rutina se encarga de generar un ciclo desde uno hasta la cantidad de tareas existentes en la lista, la cual esta depositada en la variable *Cantidad*; lo primero que se ejecuta en la rutina en cada ciclo es crear un registro adicional en DB que usa la instrucción: DB.RecordSet.AddNew; una vez creado el nuevo registro es depositado en los valores de la tarea en curso la posición del puntero llamado: *Índice.*

*Extraer datos:* Esta opción permite vaciar el registro de la base de datos en la lista de tareas, esto implica que es borrada la información almacenada con anterioridad en la lista de tareas. Por lo tanto al ser seleccionada esta opción solicita una reconfirmación al operador que utiliza la instrucción *MsgBox.* Al ser afirmativa la reconfirmación, la rutina redimensiona la lista de tareas, pero sin preservar los datos existentes con anterioridad, con la instrucción *Dim,* luego desplaza el puntero de la base de datos al primer registro y traslada registro por registro a la lista de tareas, este ciclo se repetirá hasta que el puntero de la base de datos encuentre el último registro. En la rutina se traslada elemento por elemento de cada registro a la lista de tareas, se actualiza la cantidad de datos grabados en la lista, se desplaza el puntero de la base datos al

siguiente registro y por último se redimensiona la lista de tareas para el otro registro a trasladar.

Cuando se ha ejecutado cualquiera de las opciones arriba explicadas, la rutina Command3D1.Click (), actualiza la base de datos que han sido agregados con la instrucción *DB.Recordset.Update*; seguido de ésto es descargada la ventana *BaseDatos* y habilitada la ventana principal. El botón de salir únicamente descarga la ventana BaseDatos y habilita la ventana principal.

**Listado de tarea:**



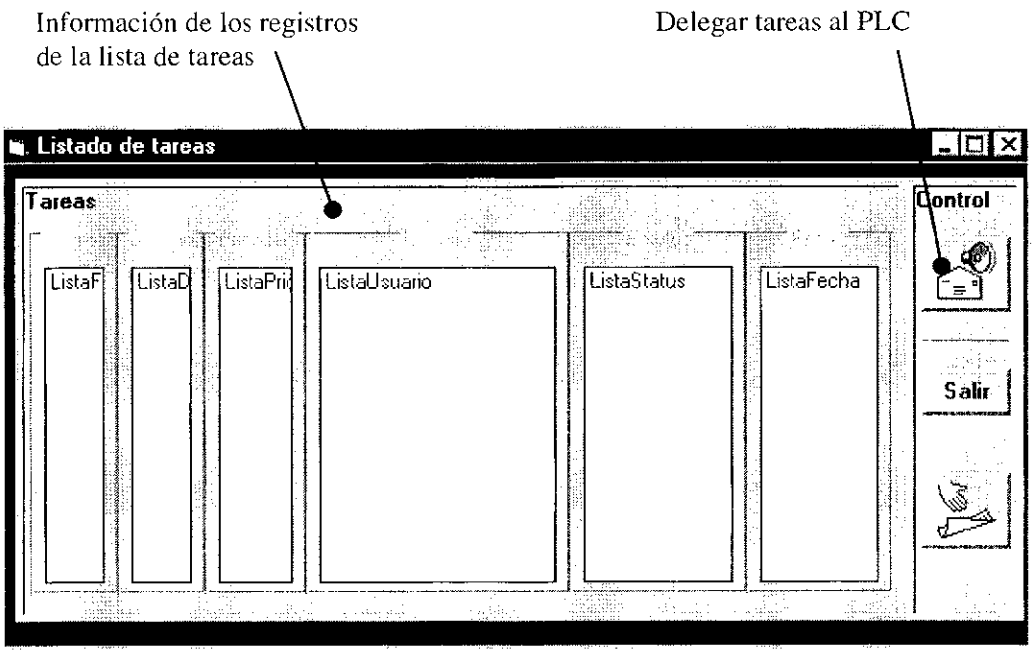

Esta opción permite desplegar (ver apéndice 7.3.1.6) la información de la lista de tareas, con todos los campos y principalmente presenta el estado en que se encuentran las tareas existentes en la lista. Para accesar esta ventana el operador deberá presionar el botón: *BotonListado*, de tipo SSCommand, el cual ejecuta la subrutina *BotonListado_Click ()*, que carga la ventana *Listado.*

La ventana *Listado*, contiene dos campos, el primero llamado de *Tareas* y el segundo de *Control.*

El campo de Tareas, está compuesto por seis objetos de tipo *ListBox,* los cuales despliegan información en forma de lista, dichos objetos tienen los nombres: *ListaFuente*, *ListaDestino*, *ListaPrioridad*, *ListaUsuario*, *ListaStatus* y *ListaFecha.*

La subrutina *BotonListado_Click()*, se encarga de tomar la información de cada registro de la lista de tareas, *ListaTareas(),* y transferirla a cada objeto *ListBox*, asignado cada campo a su respectivo *ListBox*, esta función es ejecutada con la instrucción: *ListBox. AddItem.*

Los *ListBox*, son independientes uno de otro, por lo tanto al seleccionar un elemento de un *ListBox*, únicamente selecciona a ese elemento por lo tanto deberá realizarse una rutina para que seleccione los registros en los otros *ListBox*. Para realizar esta operación, cuando es seleccionado un elemento del cualquier *ListBox*, se crea una variable llamada *Puntero*, que indica la posición numérica del elemento seleccionado, se asigna dicho valor a los demás *ListBox* que utiliza la instrucción *ListBox.Index*.



El campo de Control contiene tres opciones para la administración del campo Tareas, las cuales son:

1. Enviar tarea al controlador.
2. Eliminar tarea de la lista.
3. Salir.

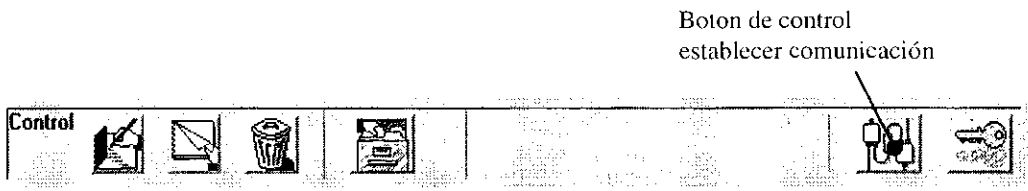
*Enviar tarea al controlador:* esta función se encarga de enviar una tarea al controlador lógico programable, que ejecuta la subrutina EnviarPLC_Click( ), es decir que la tarea cambia de estado: Tarea Generada a Tarea Delegada. Por lo tanto únicamente las tareas con estado "*Generada*", clasifican para dicha operación. Adicionalmente a esta operación de comparación, la rutina se encarga de solicitar confirmación al operador, que utiliza la operación *MsgBox*. La solicitud de confirmación al ser valedera procederá la rutina a estructurar la trama de información que se enviará al controlador, mas adelante se ampliará la explicación referente a la comunicación entre la interface y el controlador.

*Eliminar tarea:* esta función se encarga de retirar de la lista a las tareas de estado: *Generada* o *Finalizada*, que ejecuta la subrutina *BotonBorrar_Click()*, esa rutina primero verifica que los estados de la tarea seleccionada cumplan con el requerimiento de ser: Generada o Finalizada, luego solicita confirmación para ejecutar dicha acción. De ser verdadera procederá a retirar los elementos de los *ListBox*, para utilizar la instrucción *ListBox.RemoveItem*. Pero aun no se ha retirado de la lista de tareas, lo cual se realiza al ejecutar el procedimiento: *BorraElemento*.
El procedimiento *BorraElemento*, se encarga de desplazar los registros hacia un registro anterior y sobrescribe los datos sobre el registro a

eliminar. Luego de este desplazamiento se redimensiona la lista con un valor menor y preserva los datos.

*Salir:* Esta rutina se encarga de descargar la ventana Listado y habilitar la ventana principal.

## 4. Comunicación entre el controlador y la interface:




Este grupo de subrutinas permiten (ver apéndice 7.3.1.6) establecer el intercambio de información entre el controlador lógico programable y la interface. Este intercambio se logra a través del objeto de tipo *MSComm*, el cual permite la transmisión y recepción de datos vía el puerto serial, que utiliza RS 232 C.

El controlador lógico programable tiene un puerto serial el cual utiliza RS 485, pero adicionalmente tiene un convertidor RS 232/RS 485, llamado PC / PPI, que permite transparencia en la comunicación.

Cuando un paquete de datos arriban al puerto, la variable *Comm1.InBufferCount*, es activada a partir de ellos se ejecuta la subrutina Timer1_Timer(), el cual se encarga de separar byte por byte y clasificar el tipo de información que proviene del controlador.

El paquete de información está estructurado de la siguiente forma:

| D | PC | F | DS | PR |
|---|---|---|---|---|

*Destino (*D*):* Es un byte quel se encarga de definir a quien corresponde la trama a enviar, para nuestro caso como únicamente existe comunicación entre dos elementos, por lo tanto existirán dos direcciones:

**C** = Interface humano- máquina (HMI)
**P** = Controlador lógico programa (PLC).

*Pal.Control (*PC*)*: Es el byte que identifica el tipo de información que lleva el resto de la trama de información. Existen las siguientes posibilidades:

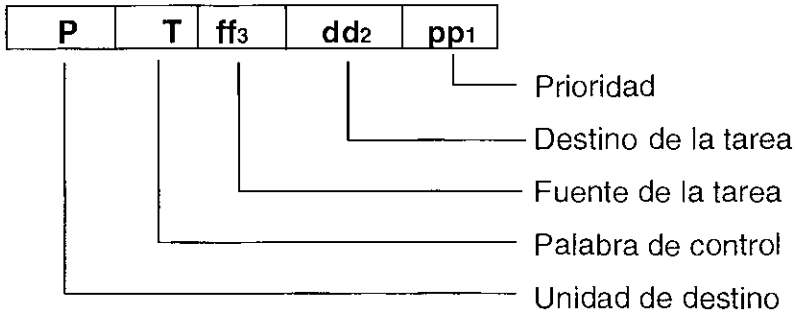
**J** = el paquete tiene información de una tarea a ser delegada al Controlador.
**C** = el paquete es utilizado para establecer comunicación.
**P** = el paquete es utilizado para configuración de la comunicación.
**T** = el paquete indica que la tarea delegada ha sido recibida Correctamente.
**E** = el paquete informa un error detectado en el controlador.
**I** = el paquete indica que la tarea ha sido iniciada.
**F** = el paquete indica que la tarea ha sido finalizada.
**P** = el paquete contiene información de sensores de posición.

Los campos F, PS y PR dependerá del tipo de información, es decir de la palabra de control.

La subrutina *Timer1.Time ()*, inicia y verifica *Comm1.InBufferCount,* al ser verdadera esta variable la rutina recoge la información almacenada en el buffer (Comm1.Input) del puerto y transfiriéndose la variable TramaRecibida, al ser recogida esta información el contenido del buffer el borrado y Comm1.InBufferCount es falsa.

*Establecer comunicación:* Para iniciar la comunicación se deberá presionar el botón de conexión, de tipo SSCommand, el cual se encargará de enviar un bloque de datos [CC00C], con el cual se hace un requerimiento de activar comunicación, al controlador. El controlador al recibir el bloque de datos, inmediatamente responderá con el bloque [PC00C] el cual indicará que la comunicación ha sido establecida. Para visualizar este evento se utiliza *MsgBox* para informar al operador que se ha establecido la comunicación.

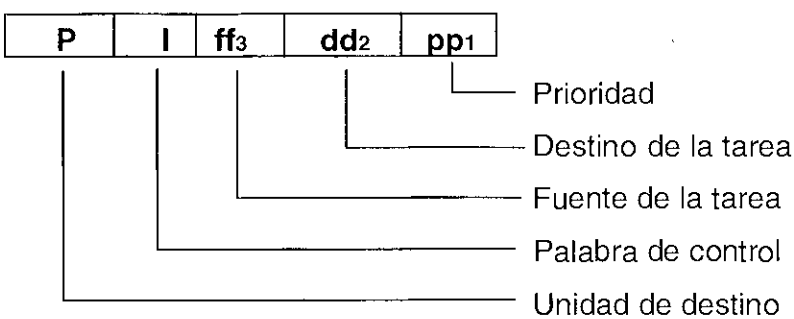
*Reconcimiento de tarea delegada:* Cuando una tarea es delegada al controlador, se envía un paquete de la siguiente estructura:




Este paquete es enviado por el puerto serial al controlador al escribir en el *buffer* de salida del controlador, el cual es: *Comm1.Output.* Al ser

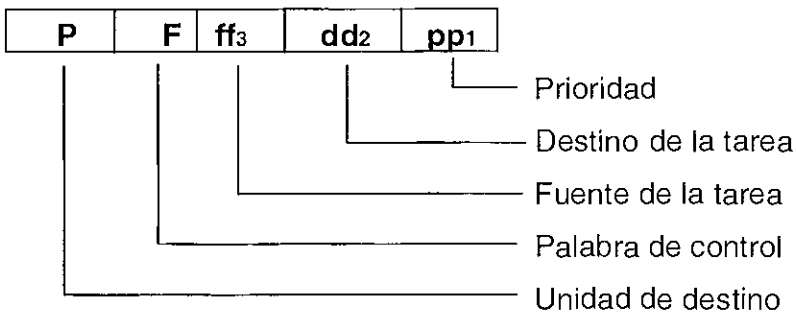
recibida la tarea por el controlador y sin detectar fallo el tamaño y estructura de la trama es enviado un paquete de información de la estructura: [CT99C]. Cuando ingresa esta información al puerto de la interface, es desplegado un mensaje en la pantalla que informa que la tarea ha sido delegada, seguidamente es cambiado el estado de la tarea: Generada -> Delegada. Al ser cambiado el estado de la tarea, es cargada la pantalla de *Listado* la cual confirmará el nuevo estado de la tarea en mención.

*Mensaje de inicio de tarea:* Cuando una tarea ha sido descargada en la cola de tareas a ejecutar por el controlador, éste al principiar una nueva tarea envía una mensaje a la interface, indicándole que una nueva con estado: Delegada, ha iniciado. El bloque de información que envía es:



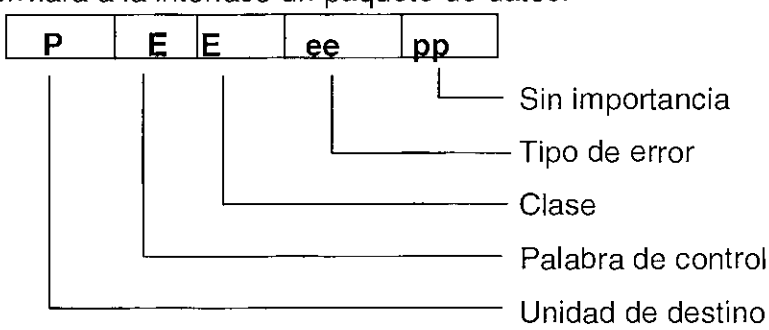

El controlador le envía la tarea pero con la palabra de control **I,** la cual indica que dicha tarea ha iniciado su desarrollo. Luego de ser desplegado el mensaje, es cargado el procedimiento: *StatusDesarrolla ()*, el cual se encarga de hacer un ciclo de búsqueda en la lista de tareas, al coincidir con los datos, es modificado el estado de dicha tarea por: "Desarrollo".

*Mensaje de tarea finalizada:* Cuando una tarea ha sido descargada en la cola de tareas del controlador, el puntero del controlador la toma e inicia la tarea y la desarrolla. Finalmente al terminar la tarea, el controlador envía un mensaje a la interface, indicándole que una tarea con estado: Desarrollo, ha finalizado. El bloque de información que envía es:



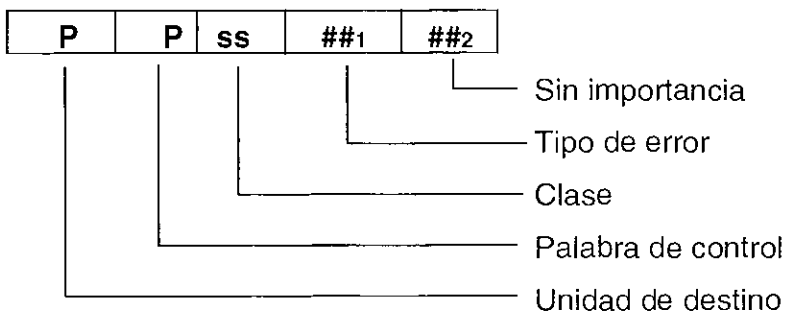

El controlador envía la información de la tarea con la palabra de control **F,** la cual indica que dicha tarea ha finalizado. Luego de ser desplegado el mensaje, es cargado el procedimiento: *StatusFinaliza ()*, el cual se encarga de hacer un ciclo en busca de la lista de tareas, compara cual de ellas coincide con los datos, al encontrarlo es modificado el estado de dicha tarea por: "*Finalizado*".

*Mensajes de error:* Cuando un error es detectado por el controlador, este enviará a la interface un paquete de datos:



Al recibir esta información, la interface toma el valor **ee** y despliega el tipo de error detectado por el controlador.

*Información de sensores:* Este paquete de información es enviado por el controlador a la interface, y la información que contiene el cual ha sido el sensor que ha detectado el controlador. Esto quiere decir que cuando el controlador detecta un sensor de posición del sistema, adicionalmente a tomar las medidas de control necesarias, éste enviará un bloque de información a la interface para que ésta la utilice para representación animada visual de forma discreta (esta pantalla será explicada más adelante).




Donde **ss** podrá ser:

**S** = Sensores sobre el riel del sistema.

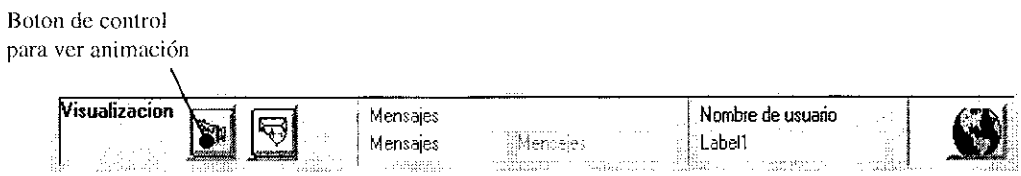
**M** = Sensores de la mesa de trabajo.

**G** = Sensores de la estantería posición baja.

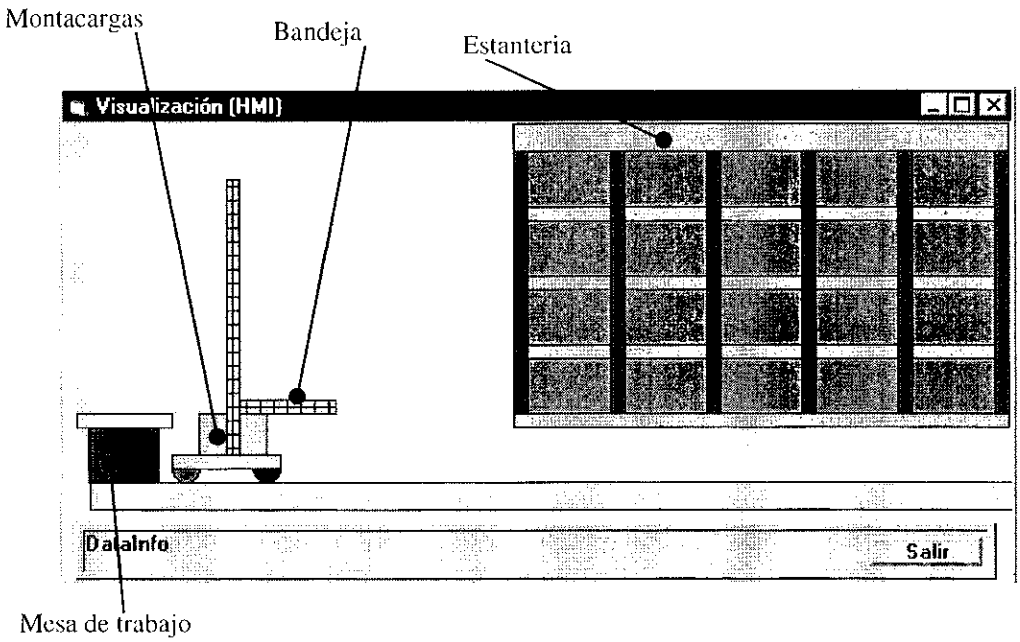
**g** = Sensores de la estantería posición alta.

Los bytes **##1** y **##2** a cuál de los sensores se refiere según los subgrupos.

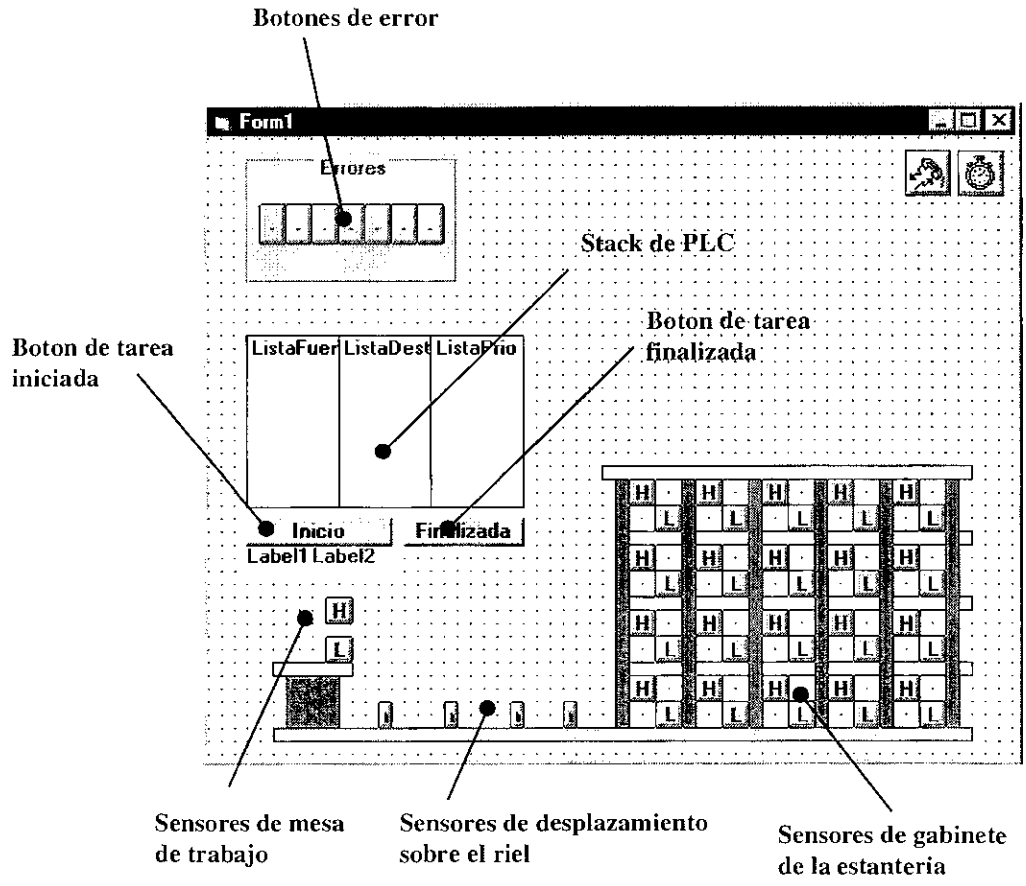
## 5. Visualización:




Esta ventana permite visualizar de una forma animada pero no continua el desplazamiento que realiza el montacargas al ejecutar una tarea. Para esto se aprovecha, que el controlador tiene conectadas a su entrada digitales de los sensores de posición. Al ser activada una entrada en el controlador, este ejecutará alguna acción, adicionalmente a ésto el controlador enviará un bloque de datos a la interface, este bloque de datos será tomado por el objeto Comm1Port, aquí se evaluará a qué grupo de sensor se refiere la información y a cual especificó, permitiéndonos desplazar los objetos (rectángulos circunferencia) de tipo *Shape*, a una posición de la pantalla que permite de esta forma desplegar en la pantalla el desplazamiento del sistema. Las propiedades que son modificadas son: *Shape.Top* y *Shape.Left*. Estos objetos son un grupo de Shape que permiten dar forma a un montacargas.


Montacargas
Bandeja
Estanteria
Visualización (HMI)
DataInfo
Salir
Mesa de trabajo

## 6. Simulador:




El simulador es otra aplicación desarrollada en VB la cual se ejecuta en otra PC y utilizan ambas el puerto Comm1, para intercambiar información.

Básicamente está constituida en una forma visual, el manejo que realizará el PLC en la memoria:

Marcas de error.

Cola de tareas delegadas.

Acciones de inicio y finalización de tareas.

Registro de la tarea en desarrollo.

Botones de los diferentes sensores.

*Marcas de errores:* Las marcas de errores son un conjunto de errores que podrá enviar el controlador al PLC.

*Cola de tareas de legadas:* Es una posición de memoria que maneja el PLC en la cual se almacenan las tareas delegadas por la interface al controlador.

*Acciones de inicio y finalización de tareas:* Son dos botones que simulan el evento cuando el PLC toma una tarea de la lista y la empieza a ejecutar. El otro botón simula el evento cuando el PLC ha terminado una tarea.

*Registro de la tarea en desarrollo:* es una posición de memoria del controlador en la cual se almacena la tarea que ha sido tomada por el controlador para ser ejecutada.

*Botones de los diferentes sensores:* Es un conjunto de botones que simulan los sensores que tiene conectados el PLC y que estan distribuidos acorde a los puntos estratégicos que el controlador para ejecutar una tarea.

# VII. RESULTADOS:

1) La interface humano-máquina fue desarrollada en el sistema de programación como un lenguaje de programación sencillo de aprendizaje e implementación en nuestro medio.

2) La información acerca de los usuarios, contraseña y niveles de acceso se encuentran almacenadas en la base de datos de tipo Access versión 2.0, para brindar seguridad al sistema.

3) La cantidad de tareas que es capaz de almacenar la interface humano-máquina en el arreglo de listas de tareas es de dos cientos cincuenta y cinco (251) tareas y una (1) que genera el sistema al iniciarse la aplicación.

4) El flujo de información entre el controlador y la interface es por ráfagas (*burst*) en la cual al enviarse tareas a delegar al controlador, únicamente una tarea es enviada por cada paquete de información.

5) La cantidad de tareas a delegar al controlador no tiene límite, debido a que el tamaño de la lista del controlador no está definida. Dicho cálculo no tiene por objetivo este trabajo.

6) La visualización del desarrollo de las tareas (visualización del proceso) se realiza de una forma discrentralizada y no continua, esto se debe a que los no que se definieron sensores de desplazamiento, únicamente se están utilizando los sensores que al controlador le son útiles para desarrollar una tarea.

# VIII. RECOMENDACIONES

1) La interface humano-máquina es capaz de manejar más controladores lógicos, para ésto deberá crearse una red de PLC en RS 485, y asignarle a través de la trama de información, la dirección a cada uno de estos.
2) La interface de humano-máquina es una unidad (celda) de trabajo que puede pertenecer a un sistema como el MES, con lo que únicamente tendría que desarrollarse la rutina del manejo de DDE para intercambiar información con el nivel superior.
3) Es importante notar que la aplicación está diseñada para ser utilizada en sistemas de almacenamiento de alta densidad donde se basa el manejo de colas FILO.

# IX. BIBIOGRAFIA


1. Programmable Logic Controler. Jones, C.T. 1996. 1ª edición Patrick – Turner, Atlanta, USA.
2. Programmer's Guide VB. Microsoft co.,1994. 3.0 version
3. Language Refence VB. Microsoft co.,1995. 3.0 version.

IX. Profesional Features Book 1. Microsoft co.,1996. 3.0 version.

X. Gunn, Thomas. Century Manufacturing 21st. Vol 13 #3 (Spring 1997).

XI. While Paper Numer 2: MES Functionalities and MRP to MES Data Flow MESA International 1994.

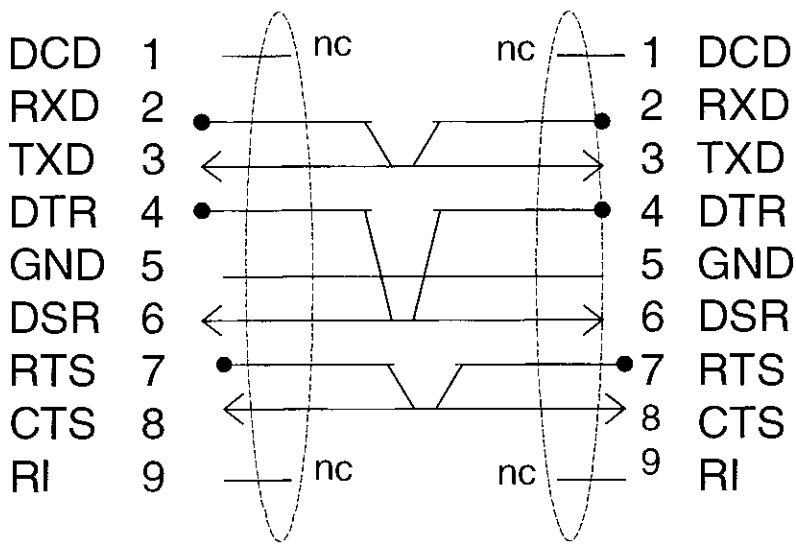
XII. Marks Eric. "MES as an Operational Enabler". Information Strategy: " Jornal 17 # 4( Summer 1997).

8. Step S7 Micro, Manual Reference. Automation Group Siemens 1996.

# X. APENDICE

## A. Diagrama de conexión del simulador:

9-pin D:



## B. Características de CPU 214:

- 2048 palabras de memoria de programa (almacenadas en memoria no volátil de lectura / escritura)
- 14 entradas digitales y 10 salidas digitales en el aparato central
- Asiste siete módulos de expansión E / S adicionales
- 128 temporizadores (4 con resolución de 1ms, 16 de 10 mseg,, y 100 de 100mseg.)
- 128 contadores combinables discrecionalmente.
- 256 marcas internas
- 688 marcas especiales
- 4 operaciones aritméticas y cálculo con coma flotante.
- Capacidad de interrupciones: 4 entradas de interrupción de hardware en flancos crecientes y decrecientes, 2 interrupciones temporizadas, 7 interrupciones de contador rápido, 2 interrupciones para tres de impulsos.
- 1 contador rápido con 2 kHz de entrada de reloj.
- 2 contadores rápidos con 7 kHz de entrada de reloj.
- 2 salidas de impulsos para funciones de salida de impulsos modulado en frecuencia (PTO) o en ancho de banda (PWM máximo 4kHz).
- 2 potenciómetros analógicos incorporados (conversión interna a valores digitales).
- Ejecución rápida de instrucciones (0.8 microsegundos por instrucción).
- La memoria de datos es respaldada por el condensador de alta potencia típica 190 horas.
- Sistema de seguridad a 3 niveles de contraseña
- Reloj de tiempo real
- Módulo de acumulador opcional.

## C. Código de Programación:

## 1. Código de inicialización:

```
Type Tarea
   Fuente As Integer
   Destino As Integer
   Prioridad As Integer
   Status As Variant
   Fecha As Variant
   Usuario As String
End Type

Dim HMIBaseDato As Database
Posici TareaGenerada As Tarea
Posici ListaTareas()  As Tarea

Posici Cantidad As Integer
Posici Puntero As Integer

Posici IngresoUsuario As String
Posici NivelUsuario As Integer

Posici Comunica As Integer
Global Cabeza As String
Global Cword As String
Global Fuen As String
Global Dest As String
Global Cola As String

Type Posicion
   Izq As Integer
   Arriba As Integer
End Type
Posici Carro(10) As Posición
```

## 2. Código de inicialización de variables:

```
Sub Form_Load ()

   Comunica = 0

   CandadoOn.Visible = False
   CandadoOff.Visible = True
   Cantidad = 1

   ReDim ListaTareas(Cantidad)
   ListaTareas(Cantidad).Fuente = 0
   ListaTareas(Cantidad).Destino = 0
   ListaTareas(Cantidad).Prioridad = 0
   ListaTareas(Cantidad).Status = "Generada"
```

```
    ListaTareas(Cantidad).Usuario = "Sistema"
    ListaTareas(Cantidad).Fecha = Now

    IngresoUsuario = "Ninguno"

    Command3D1.Enabled = False
    BotonListado.Enabled = False
    BorrarTarea.Enabled = False
    RecuperarDB.Enabled = False
    Animacion.Enabled = False

    Comm1.CommPort = 1
    Comm1.Settings = "9600,N,8,1"
    Comm1.InputLen = 0
    Comm1.PortOpen = True
    Mensajes2.Caption = "Desconectados"

End Sub
```

## 3. Código de acceso al sistema

```
Sub AccesoUsuario_Click ()
    HMI.Enabled = False
    IngresoUsuario = "Ninguno"
    Load Acceso
    Acceso.Visible = True
End Sub

Sub Ingresar_Click ()

    HMI.CandadoOff.Visible = True
    HMI.CandadoOn.Visible = False
    IngresoUsuario = "Ninguno"
    NivelUsario = 0

    DBUsuario.Recordset.MoveFirst

    Do Until DBUsuario.Recordset.EOF
      If Nombreusuario.Text = DBUsuario.Recordset("Nombre") Then
        If Claveusuario.Text = DBUsuario.Recordset("Clave") Then
          IngresoUsuario = Nombreusuario.Text
          NivelUsuario = DBUsuario.Recordset("Nivel")
          HMI.CandadoOff.Visible = False
          HMI.CandadoOn.Visible = True
          HMI.Command3D1.Enabled = True
          HMI.BotonListado.Enabled = True
          HMI.BorrarTarea.Enabled = True
          HMI.RecuperarDB.Enabled = True
          HMI.Animacion.Enabled = True
          HMI.Mensajes1.Caption = "Acceso habilitado al sistema"
        End If
      End If
```

```
    DBUsuario.Recordset.MoveNext
    Loop

    Unload Acceso
    HMI.Show
    HMI.Enabled = True
    HMI.Nombreusuario.Caption = IngresoUsuario

End Sub
Sub Liberar_Click ()
     Respuesta = MsgBox("¿Desea cerrar acceso?", 36, "Acceso al sistema")
     If Respuesta = 6 Then
     HMI.Command3D1.Enabled = False
     HMI.BotonListado.Enabled = False
     HMI.BorrarTarea.Enabled = False
     HMI.RecuperarDB.Enabled = False
     HMI.Animacion.Enabled = False
     HMI.CandadoOff.Visible = True
     HMI.CandadoOn.Visible = False
     IngresoUsuario = "Ninguno"
     NivelUsario = 0
     Unload Acceso
     HMI.Show
     HMI.Enabled = True
     HMI.Mensajes1.Caption = "Acceso bloqueado al sistema"
End If

End Sub
```

## 4. Código de generación de tareas

```
Sub Mercaderia_DragDrop (Index As Integer, Source As Control, X As Single, Y As
Single)
    HMI.Mensajes1.Caption = "Generando tarea...."
    TareaGenerada.Fuente = Source.Index
    TareaGenerada.Destino = Mercaderia(Index).Index
    TareaGenerada.Usuario = IngresoUsuario
    TareaGenerada.Fecha = Now
    HMI.Enabled = False
    Load GeneraTarea
    GeneraTarea.Show
End Sub

Sub Aceptar_Click ()
      Cantidad = Cantidad + 1
      ReDim Preserve ListaTareas(Cantidad)
      ListaTareas(Cantidad).Fuente = TareaGenerada.Fuente
      ListaTareas(Cantidad).Destino = TareaGenerada.Destino
      ListaTareas(Cantidad).Prioridad = TextoPrioridad.Text
      ListaTareas(Cantidad).Status = "Generada"
      ListaTareas(Cantidad).Usuario = TareaGenerada.Usuario
```

```
    ListaTareas(Cantidad).Fecha = TareaGenerada.Fecha
  Unload GeneraTarea

  HMI.Show
  HMI.Enabled = True
  HMI.Mensajes1.Caption = " Tarea generada en el sistema"
End Sub

Sub Negar_Click ()
  Unload GeneraTarea
  HMI.Show
  HMI.Enabled = True
  HMI.Mensajes1.Caption = "Tarea denegada"
End Sub
```

### 5. Código de respado de listado de tareas:

```
Sub RecuperarDB_Click ()
  HMI.Enabled = False
  Load BaseDatos
  BaseDatos.Visible = True
End Sub

Sub Command3D1_Click ()
  If Borradatos.Value Then
    Respuesta = MsgBox("¿Desea borrar base de datos?", 20, "Base de datos")
    If Respuesta = 6 Then
     DB.Recordset.MoveFirst
     Do Until DB.Recordset.EOF
        DB.Recordset.Delete
        DB.Recordset.MoveNext
     Loop
    End If
  End If
  If Agregadatos.Value Then
    For Indice = 1 To Cantidad
        DB.Recordset.AddNew
        DB.Recordset("Fuente") = ListaTareas(Indice).Fuente
        DB.Recordset("Destino") = ListaTareas(Indice).Destino
        DB.Recordset("Prioridad") = ListaTareas(Indice).Prioridad
        DB.Recordset("Usuario") = ListaTareas(Indice).Usuario
        DB.Recordset("Fecha") = ListaTareas(Indice).Fecha
        DB.Recordset.Update
     Next Indice
  End If
  If Extraedatos.Value Then
    Respuesta = MsgBox("Sobreescribir en lista de tareas", 20, "Base de datos")
    If Respuesta = 6 Then
    ReDim ListaTareas(1)  As Tarea
    DB.Recordset.MoveFirst
```

```
      Indice = 1
      Do Until DB.Recordset.EOF
         ListaTareas(Indice).Fuente = DB.Recordset("Fuente")
         ListaTareas(Indice).Destino = DB.Recordset("Destino")
         ListaTareas(Indice).Prioridad = DB.Recordset("Prioridad")
         ListaTareas(Indice).Status = "DB Extraida"
         ListaTareas(Indice).Fecha = DB.Recordset("Fecha")
         ListaTareas(Indice).Usuario = DB.Recordset("Usuario")
         DB.Recordset.MoveNext
         Indice = Indice + 1
         ReDim Preserve ListaTareas(Indice)

      Loop
      Cantidad = Indice - 1
      End If
   End If
   DB.Recordset.Update
   Unload BaseDatos
   HMI.Enabled = True
   HMI.Visible = True
End Sub
```

### 6. Código de listado de tareas:

```
Sub BotonListado_Click ()
      If IngresoUsuario <> "Ninguno" Then
      Load Listado
      Listado.Visible = True
      End If
End Sub

Sub Form_Load ()
   For Index = 1 To Cantidad
      ListaFuente.AddItem Str(ListaTareas(Index).Fuente)
      ListaDestino.AddItem Str(ListaTareas(Index).Destino)
      ListaPrioridad.AddItem Str(ListaTareas(Index).Prioridad)
      ListaStatus.AddItem ListaTareas(Index).Status
      ListaUsuario.AddItem ListaTareas(Index).Usuario
      ListaFecha.AddItem ListaTareas(Index).Fecha
   Next Index
End Sub

Sub ListaFuente_Click ()
   Puntero = ListaFuente.ListIndex
   ListaDestino.ListIndex = ListaFuente.ListIndex
   ListaPrioridad.ListIndex = ListaFuente.ListIndex
   ListaUsuario.ListIndex = ListaFuente.ListIndex
   ListaStatus.ListIndex = ListaFuente.ListIndex
End Sub
Sub ListaDestino_Click ()
```

```
   Puntero = ListaDestino.ListIndex
   ListaFuente.ListIndex = ListaDestino.ListIndex
   ListaPrioridad.ListIndex = ListaDestino.ListIndex
   ListaUsuario.ListIndex = ListaDestino.ListIndex
   ListaStatus.ListIndex = ListaDestino.ListIndex
End Sub
Sub ListaPrioridad_Click ()
   Puntero = ListaPrioridad.ListIndex
   ListaFuente.ListIndex = ListaPrioridad.ListIndex
   ListaDestino.ListIndex = ListaPrioridad.ListIndex
   ListaUsuario.ListIndex = ListaPrioridad.ListIndex
   ListaStatus.ListIndex = ListaPrioridad.ListIndex
End Sub
Sub ListaUsuario_Click ()
   Puntero = ListaUsuario.ListIndex
   ListaFuente.ListIndex = ListaUsuario.ListIndex
   ListaDestino.ListIndex = ListaUsuario.ListIndex
   ListaPrioridad.ListIndex = ListaUsuario.ListIndex
   ListaStatus.ListIndex = ListaUsuario.ListIndex
End Sub
Sub ListaStatus_Click ()
   Puntero = ListaStatus.ListIndex
   ListaFuente.ListIndex = ListaStatus.ListIndex
   ListaDestino.ListIndex = ListaStatus.ListIndex
   ListaPrioridad.ListIndex = ListaStatus.ListIndex
   ListaUsuario.ListIndex = ListaStatus.ListIndex
End Sub

Sub EnviarPLC_Click ()
    If (ListaTareas(Puntero + 1).Status) = "Generada" Or (ListaTareas(Puntero +
1).Status) = "DB Extraida" Then
    Respuesta = MsgBox("¿Desea enviar tarea al PLC?" & ListaTareas(Puntero +
1).Fuente & "->" & ListaTareas(Puntero + 1).Destino, 36, "Tareas PLC")
    If Respuesta = 6 Then
      Fuente = Chr(ListaTareas(Puntero + 1).Fuente)
      Destino = Chr(ListaTareas(Puntero + 1).Destino)
      Prioridad = Chr(ListaTareas(Puntero + 1).Prioridad)
      HMI.Comm1.Output = "PT" + Fuente + Destino + Prioridad
    End If
    Else
      MsgBox "No transferible por status...", 16
    End If
End Sub

Sub BotonBorrar_Click ()
   If ListaTareas(Puntero + 1).Status = "Generada" Or ListaTareas(Puntero + 1).Status
= "Finalizada" Then
    Respuesta = MsgBox("¿Desea borrar tarea generada?", 36, "Borra tarea")
    If Respuesta = 6 Then
      ListaFuente.RemoveItem Puntero
```

```
      ListaDestino.RemoveItem Puntero
      ListaPrioridad.RemoveItem Puntero
      ListaUsuario.RemoveItem Puntero
      ListaStatus.RemoveItem Puntero
      ListaFecha.RemoveItem Puntero
      BorraElemento
    End If
  Else
        MsgBox "..unicamente tareas generadas y finalizadas!", 48, "Borra tarea"
  End If
End Sub

Sub BorraElemento ()
  For Indice = Puntero + 1 To Cantidad - 1
    ListaTareas(Indice).Fuente = ListaTareas(Indice + 1).Fuente
    ListaTareas(Indice).Destino = ListaTareas(Indice + 1).Destino
    ListaTareas(Indice).Prioridad = ListaTareas(Indice + 1).Prioridad
    ListaTareas(Indice).Usuario = ListaTareas(Indice + 1).Usuario
    ListaTareas(Indice).Status = ListaTareas(Indice + 1).Status
  Next Indice
  ReDim Preserve ListaTareas(Cantidad - 1)
  Cantidad = Cantidad - 1
End Sub
```

**7. Código de comunicación controlador e interface:**

```
Sub Timer1_Timer ()
  If Comm1.InBufferCount Then
    TramaRecibida = Comm1.Input
     BarraCom.Height = 255
    If TramaRecibida = "PC00C" Then
      EnLinea.Visible = True
      SinLinea.Visible = False
      MsgBox "Comunicacion activada!", 48
      Mensajes1.Caption = "Comunicacion activada a PLC..."
      Mensajes2.Caption = "Conectado"
    End If
    BarraCom.Height = 975
    If TramaRecibida = "CT99C" Then
      Unload Listado
      ListaTareas(Puntero + 1).Status = "Delegada"
      Load Listado
      Listado.Visible = True
      MsgBox "Tarea delegada al PLC !", 64
      Mensajes1.Caption = "Tarea delegada al PLC"
    End If
    BarraCom.Height = 495
  If Len(TramaRecibida) = 5 Then
  ErrorTrama = 0
  Cabeza = Mid$(TramaRecibida, 1, 1)
```

```
CWord = Mid$(TramaRecibida, 2, 1)
Fuen = Mid$(TramaRecibida, 3, 1)
Dest = Mid$(TramaRecibida, 4, 1)
Cola = Mid$(TramaRecibida, 5, 1)
If (Cabeza + CWord + Fuen) = "PEE" Then
   MsgBox "Error detectado en PLC. Error #" & Asc(Dest), 16
End If
If (Cabeza + CWord + Fuen) = "PPS" Then
   Posicion
End If
 BarraCom.Height = 495
If (Cabez + CWord + Fuen) = "PPM" Then
   PosicionMesa
End If
BarraCom.Height = 735
If (Cabeza + CWord + Fuen) = "PPG" Then
   PosicionGabiHi
End If
 BarraCom.Height = 735
If (Cabeza + CWord + Fuen) = "PPg" Then
   PosiGabiLo
End If
If (Cabeza + CWord) = "PI" Then
   MsgBox "Tarea " & Asc(Fuen) & "->" & Asc(Dest) & "ha sido iniciada !", 64
   Unload Listado
   StatusDesarrolla
      Load Listado
      Listado.Visible = True
      Mensajes1.Caption = "Tarea inicia desarrollo..."

End If
If (Cabeza + CWord) = "PF" Then
   MsgBox "Tarea " & Asc(Fuen) & "->" & Asc(Dest) & "ha sido finalizada !", 64
   Unload Listado
   StatusFinaliza
      Load Listado
      Listado.Visible = True
      Mensajes1.Caption = "Tarea finalizada..."

End If

Else
   ErrorTrama = ErrorTrama + 1
   If ErrorTrama > 5 Then
      MsgBox "Error en comunicacion", 16
      ErrorTrama = 0
   End If
BarraCom.Height = 735
End If
End If
```

```
   BarraCom.Height = 0
End Sub

Sub Conexion_Click ()
   If Not (Comunica = 1) Then
      Comunica = 1
      HMI.Comm1.Output = "CC00C"
      HMI.Mensajes1.Caption = "Estableciendo comunicacion con PLC..."
   Else
      Respuesta = MsgBox("Terminar comunicacion con PLC?", 20, "Comunicacion")
      If Respuesta = 6 Then
      Comunica = 0
      EnLinea.Visible = False
      SinLinea.Visible = True
      HMI.Comm1.Output = "CC11D"
      HMI.Mensajes2.Caption = "Desconectado"
      HMI.Mensajes1.Caption = "Comunicacion finalizada con PLC"
      End If
   End If
End Sub

Sub StatusDesarrolla ()
   For k = 1 To Cantidad
      If Chr$(ListaTareas(k).Fuente) = (Fuen) And Chr$(ListaTareas(k).Destino) =
(Dest) Then
         ListaTareas(k).Status = "Desarrollo"
      End If
   Next
End Sub

Sub StatusFinaliza ()
   For k = 1 To Cantidad
      If Chr$(ListaTareas(k).Fuente) = (Fuen) And Chr$(ListaTareas(k).Destino)
= (Dest) Then
         ListaTareas(k).Status = "Finalizada"
      End If
   Next
End Sub
```

## 8. Estructura de la aplicación en VB

```
INTERHM.FRM
C:\WINDOWS\SYSTEM\GRID.VBX
C:\WINDOWS\SYSTEM\MSOLE2.VBX
C:\WINDOWS\SYSTEM\ANIBUTON.VBX
C:\WINDOWS\SYSTEM\CMDIALOG.VBX
C:\WINDOWS\SYSTEM\CRYSTAL.VBX
C:\WINDOWS\SYSTEM\GAUGE.VBX
C:\WINDOWS\SYSTEM\GRAPH.VBX
C:\WINDOWS\SYSTEM\KEYSTAT.VBX
C:\WINDOWS\SYSTEM\MSCOMM.VBX
C:\WINDOWS\SYSTEM\MSMASKED.VBX
C:\WINDOWS\SYSTEM\MSOUTLIN.VBX
C:\WINDOWS\SYSTEM\PICCLIP.VBX
C:\WINDOWS\SYSTEM\SPIN.VBX
C:\WINDOWS\SYSTEM\THREED.VBX
DECLARA.BAS
PROCED.BAS
GENERATA.FRM
ACCESO.FRM
LISTADO.FRM
BASEDATO.FRM
VISUALIZ.FRM
ProjWinSize=382,617,182,232
ProjWinShow=2
IconForm="HMI"
```

## D. Glosario de palabras y acrónimos.

CIM Computer integrated to manufacturing.

MPS Manufacturing planning system.

MRP Manufacturing resource planning.

ERP Enterprice resource planning

MES Manufacturing execution system.

MSS Manufacturing supervisory system.

SCADA Supervisory collection and adquitition data.

RTD Resistance temperature detector.

ITT Infrared temperature transmiter.

CAN Control area network.

PLC Programable logic controler.

VB Visual Basic

# XI. ANEXO

## B. Incremento de unidades de monitoreo.

El equipo es capaz de manejar hasta un máximo de 20 paletas en la estantería. Esta capacidad para un sistema real, no lograría cubrir la necesidad de almacenaje en una empresa mediana. Para ello es indispensable ampliar la cantidad de entradas digitales del PLC, lo cual se logra incrementando módulos al equipo y dispositivos de periferia (sensores). El módulo a utilizar será: EM221 DI 8x24VDC (6ES7 221-18F00-0XA0).

La cantidad de módulos estará definida por la cantidad de paletas adicionales (2 por paleta) y cada módulo es capaz de recibir 8 entradas. Entonces la cantidad de módulos a utilizar será de n/8, donde n es la cantidad de paletas para ampliar.

La interface humano máquina deberá redimensionar la variable de tipo global Carro del tipo posición, a la cantidad nueva de estanterías a utilizar (ver sección 3.3.8 y 7.3.1.1)

Todo lo anterior corresponde a la plataforma básica de "hardware" necesario para dicha implementación, adicional a ello y obligatorio es ampliar la lógica de direccionamiento y la matriz de ubicación dentro del controlador lógico.

### C. Matriz del estado real de estantería.

La matriz de estado real de estantería, consiste en bloques de información del tipo de producto, cantidad del producto, fecha de almacenaje, fecha de vencimiento (si fuese necesaria), modelo, número de parte, origen, etc. Esto permitirá que el usuario tenga la totalidad de la información para la administración del producto contenido en las paletas.

Las modificaciones que se necesitan realizar son:

1. Generar la variable de tipo global (Database) denominada DatoPaleta con que se definirá la cantidad de variables tipo string, date o etc. (ver sección 1.5.5 y 1.5.6), según las necesidades de la aplicación. Agregándose en la sección 7.3.1.1.
2. Desarrollo de las formas y el código de interacción con el usuario según la sección 1.5.4.

### D. Administración de las tareas delegadas controlada por el HMI.

La aplicación original para el manejo de 20 paletas en la estantería, es un valor corto de una aplicación a mediano tamaño, por lo tanto en un sistema donde la cantidad de tareas a delegar el sistema del controlador (ver sección VI.B.2 y VI.B.4) logrará saturar la memoria del controlador, es necesario trasladar esta información a la PC del HMI, ya que la cantidad de memoria de una PC supera la capacidad de nuestro equipo (ver sección VI.B y X.B). Es importante hacer notar que el término tarea delegada sería sustituido por tarea en cola de trabajo.

La ventaja de tener la administración de la cola de trabajo en la PC es que en el momento de la pérdida de suministro eléctrico en el controlador, esta información no se perdería como es el caso actual, ya que la cantidad de memoria remanente del controlador sólo logra cubrir 1/3 de las marcas especiales del sistema.

Los cambios necesarios para realizar dicha modificación son:

1. La sección VI.B.2 presenta el listado de las tareas y su respectivo estado (generada, delegada, proceso, finalizada), adicional a ello existe el botón de delegar al controlador, el cual será cambiado por: ingreso a cola de tareas.
2. La sección X.C.6 que contiene el código de enviar al PLC (Sub EnviarPLC_Click ()) deberá únicamente modificar la variable de estado delegado por en "cola de trabajo", y no escribir en el buffer del puerto serial la tarea.
3. La sección X.C.7 que contiene el código de la administración de la comunicación entre el controlador y la

interface deberá agregarse el código, cuando el controlador detecte una tarea terminada, la interface tomará la tarea designada en cola de trabajo según la prioridad y el orden en que fueron desarrolladas y enviarla a controlador para su respectivo desarrollo.

**UNIVERSIDAD DEL VALLE DE GUATEMALA**

Facultad de Ciencias y Humanidades

Departamento de Ingeniería Electrónica.

# Interface humano-máquina para control de un robot.

Mynor Antonio Joachín Godínez

Guatemala

1998.

Vo. Bo.:

(f) _____.
Dr.- Ing. Manuel Antonio López Valdés.
Asesor.

Terna examinadora:

(f) _____.
Dr.- Ing. Manuel Antonio López Valdés.

(f) _____.
Ingeniero Luis Furlán Collver.

(f) _____.
Ingeniero Daniel Sandoval Monzón.

Fecha de aprobación: 06 de Agosto de 1,998.