INSTITUTO TECNOLOGICO DE COSTA RICA
VICERRECTORÍA DE INVESTIGACIÓN Y EXTENSIÓN
DIRECCIÓN DE PROYECTOS

Actividad de Fortalecimiento:

"Diseño e implementación de los algoritmos, para controlar dos celdas flexibles de manufactura, usando el método aproximación basada en matrices"

Documento I : Informe técnico.

Asesor Oficial de proyectos
Master. Rolvin Salas Quesada

Investigador:
Ing. Luis Diego Murillo Soto. M.C.

Enero 2012

Tabla de contenido

_Toc315641071
1 Resumen 5
2 Palabras Claves. 5
3 Introducción. 5
3.1 Clasificación de sistemas FMS 6
3.2 Problemática de los FMS. 7
3.3 Descripción de la celda teórica. 8
3.4 Descripción de la celda en laboratorios de sistemas modernos. 9
4 Objetivos (General y Específicos) 10
4.1 Cumplimiento de objetivos 10
5 Revisión Literaria. 11
5.1 Aproximación Basada en Matrices 11
5.2 Celda multi re-entrantes: consideraciones generales. 13
5.3 Redes de Petri. 14
5.4 Bloqueo en celdas de manufactura. 15
5.5 Políticas de despacho para el controlador 17
5.5.1 LBFS 18
5.5.2 FBFS. 18
6 Equipos y software 19
6.1 Equipo de la celda de manufactura de laboratorio. 19
6.2 Software de simulación 20
6.3 Software para la implementación de laboratorio. 21
7 Analisis de las celdas de manufactura 22
7.1 Pasos de análisis y elaboración del controlador para la celda teórica 22
7.1.1 Análisis de la celda teórica y la matrices del controlador. 23
7.1.2 Bloqueo de celdas de manufactura. 26
7.1.3 Propuesta para la asignación de recursos en la celda de manufactura 27
7.1.4 Algoritmo de control. 29
7.2 Elaboración del controlador para la celda de laboratorio. 32
7.2.1 Árbol de ensamble de la celda de manufactura. 32
7.2.2 Red de Petri de la celda de manufactura. 33
7.2.3 Obtención de los lugares de conflicto del sistema 34
7.2.4 Obtención de las esperas circulares y sifones críticos del sistema 34
8 Simulaciones de las celdas de manufactura 35
8.1 Implementación del controlador para la celda teórica en Petri.Net 35
8.2 Simulación de la celda teórica 36

8.2.1 Simulaciones realizadas de la celda sin controlador. 36
8.2.2 Simulaciones realizadas de la celda con el algoritmo de control. 37
8.3 Pruebas del controlador para la celda de laboratorio..... 39
8.3.1 Simulación de la celda de laboratorio sin reglas de control 39
8.3.2 Diseño y simulación del algoritmo de control LBFS 40
8.3.3 Diseño del algoritmo de control FBFS..... 41
9 Implementación del modelo matricial en el PLC 42
9.1 Estructura del control del proceso y de la comunicación entre las máquinas 42
9.2 Lista de conexiones entre las máquinas 43
9.3 Programa del Controlador Lógico Programable..... 44
9.4 Programa del controlador del brazo robótico 44
10 Resultados y Discusión 45
10.1 Celda de manufactura teórica 45
10.2 Celda de manufactura de laboratorio..... 45
10.2.1 Comportamiento de la celda de manufactura de laboratorio..... 47
11 Conclusiones. 47
12 Aportes 48
13 Referencia bibliográfica 48
14 Apendice 1: Programa PLC CELDA 50
15 Apendice 2: Programa Robot Pegasus 72

Tabla de figuras.

Figura 1: Clasificación de FMS según el volumen y el número de partes que produce (MS Visio 2007)..... 7
Figura 2: Layout de una celda de manufactura planteada en [1]. (MS Visio 2007). 8
Figura 3: Layout de la celda de manufactura con su fotografía 9
Figura 4. Relación entre el autómata programable y la celda flexible de manufactura (MS Visio 2007)..... 12
Figura 5. Red de petri, con cuatro lugares, cuatro transiciones, cinco arcos dirigidos de peso uno, y cuatro arcos dirigidos de peso dos. Visio 2007. 14
Figura 6. Ejemplo de un proceso sencillo de ensamble, tomado de [5] 18
Figura 7. Equipo de la celda de manufactura. a) Controlador lógico programable PLC CompactLogix 5330. b) Robot Pegasus. c) : Driver del robot con puertos de E/S. d) Máquina CNC de la celda. e) Mesa de ensamble para la caja de engranes. f) A almacén (buffer) para el acrílico maquinado. g) Banda transportadora de salida para producto terminado, funciona también como entrada de tapas de acrílico. h) Entrada de las cajas de engranes y de los piñones plásticos. 20
Figura 8. Interface gráfica de usuario principal del software Petri.Net. 21
Figura 9. a) GUI del software Pegasus para programar el control del robot. b) GUI del RSLogix5000 para programar el controlador de la celda de manufactura. 22
Figura 10. Flujo de partes dentro de la celda. Se observa que la celda descrita en [1], es del tipo FMRF ya que una piezas entran al menos 2 veces en un recurso mismo recurso y existen dos rutas de tareas para la pieza tipo C. La celda consta de 5 recursos, un buffer y realiza 20 tareas. (MS Visio 2007) 23
Figura 11. Arboles de ensamble de la celda de manufactura. Las secuencias A), B), C) describen la el orden de las tareas para las piezas A, B, C respectivamente. (MS Visio 2007). 25
Figura 12. Red de Petri sin controlador. Los círculos con marcado uno representan los recursos del sistema y los círculos sin marcado representan las tareas. (Editor Petri.Net)..... 26
Figura 13. Red de Petri con los nodos lugar de control. Estos nodos de control se representan como círculos dobles y tienen un arco dirigido a un nodo transición. (Editor Petri.Net)..... 32
Figura 14. Árbol de ensamble de la celda de manufactura, para la fabricación de una caja reductora. (MS Visio 2007). 33
Figura 15. Red de Petri del laboratorio de manufactura sin lugares de control. (Editor Petri.Net).... 33
Figura 16. RdP con los lugares de control, en las transiciones donde en conflicto. (Editor Petri.Net). 34
Figura 17. Los recursos R, B, M, que conforman las esperas circulares del proceso de ensamble. .. 35
Figura 18. Editor de reglas del programa Petri.Net, donde se implementó las reglas del control descritas en e.c (16). 35
Figura 19. Oscilograma de las colas de entrada y salida de las piezas A, B, C, Así como, la utilización de las máquinas y buffer de la celda. Para las máquinas el valor de cero o negativo indica maquinaria en uso. 36
Figura 20. Resultado numérico parcial de la simulación de la RdP sin controlar. El software indica los pasos específicos en que el sistema se bloquea,la fila se marca la fila con la letra L. Si existe conflicto de recursos, la fila se marca la letra C. 37
Figura 21. Oscilograma para la RdP controlada con el algoritmo propuesto. Se observa que los conflictos y bloqueos se han eliminado. 38
Figura 22. Resultado parciales de las últimas 32 simulaciones. Se observó que en mil pasos de simulación el sistema no presento conflictos ni bloqueos. 39
Figura 23 Tareas bloqueadas a partir de la novena iteración con entradas periódicas cada 3 ciclos.. 40
Figura 24: Comportamiento de algunas de las tareas con la política de LBFS 40
Figura 25. Comportamiento de los recursos con la política de LBFS 41
Figura 26. Comportamiento de algunas tareas con la política de FBFS..... 41
Figura 27. Comportamiento de los recursos con la política de FBFS 42
Figura 28. Conexiones entre las máquinas de la celda. 43

1 RESUMEN

El presente proyecto se diseñó e implementó dos algoritmos para controlar dos celdas de manufactura a partir de la metodología llamada aproximación basada en matrices. Dicho método es poco conocido y puede ser utilizado en el desarrollo de algoritmos para controladores programables (PLC), como se demostrará en el presente trabajo. Este proyecto de fortalecimiento permitió adquirir conocimiento necesario para aplicar el método, este ¿*know how*? adquirido, permitirá posteriormente realizar implementaciones en la industria.

Los resultados de este proyecto buscan aprovechar y aplicar las nuevas metodologías y mejores herramientas que permitan evaluar alternativas de bajo costo para volúmenes de producción relativamente moderados, con una alta mezcla de productos, que típicamente se presentan en los procesos productivos actuales de las pequeñas y medianas empresas manufactureras

El método de aproximación basada en matrices, se aplicará a dos problemas: el primero a una celda de manufactura teórica descrita en por Peng, 2003 [**1**], y el otro caso surge de la celda manufactura que se encuentra en el laboratorio de sistemas modernos de manufactura, de la Escuela de Ingeniería en Producción Industrial.

En cuanto al diseño del controlador en la celda de manufactura práctica, se obtuvieron los resultados esperados, se logró una implementación del controlador matricial en un PLC RSLogic 5330, se implementó dos políticas de despacho diferentes LBFS y FBFS. La implementación se realizó en una celda de manufactura flexible con un brazo robótico, una máquina CNC, una mesa de ensamble y cuatro entradas de piezas diferentes.

El diseño del controlador para la celda de manufactura teórica descrita en [**1**], se encontró que las políticas de despacho para el método matricial son aplicables a sistemas multi reentrantes (MFR), pero no así a sistemas multi- reentrantes de libre escogencia (FMRF), ni a sistemas híbridos que son la combinación de MRF y FMRF. En el proceso de análisis de la celda teórica, se determinó que esta es un tipo de celda híbrida, por lo que la forma en que se construyen las políticas descritas por Bogdan, 2006 [**2**], no son aplicable en forma directa. Por tal razón, se investigó cómo desarrollar una política adecuada, y se propuso los postulados de cómo crear la política; finalmente se probó la política de control de despacho mediante simulaciones empleando el software Petri.NET, arrojando que la política propuesta controla la celda, evitando los conflictos y los bloqueos.

2 PALABRAS CLAVES.

Celda de manufactura flexible (FMS), Sistemas multi reentrantes (MRF), Sistemas de libre escogencia multi reentrantes (FMRF), Bloqueos (Deadlocks), Políticas de despacho.

3 INTRODUCCIÓN.

El concepto inicial de un Sistema Flexible de Manufactura (siglas en ingles FMS) surgió en la fábrica de compresores Ingersoll-Rand en 1970 [**3**]. Un FMS se compone en forma primaria de un grupo de estaciones de trabajo, interconectada mediante sistemas automatizados de manejo y almacenamiento de materiales, todo esto controlado por un sistema computarizado de control. Cada elemento primario del FMS se explica a continuación:

- Estaciones de procesamiento, usualmente máquinas computarizadas de control numérico (CNC), que realizan operaciones de maquinado sobre familias de partes. Sin embargo, sistemas

automatizados de inspección, ensamblado de piezas y robots, se pueden incorporar como parte de este concepto.

- Sistemas automatizados de manejo y almacenamiento de materiales, estos sistemas permiten el direccionamiento flexible de piezas.
- Sistema computarizado de control, es el encargado de coordinar las actividades entre las estaciones de procesamiento y los sistemas de manejo de materiales. Además de monitorear el estatus de los trabajos, re direccionamiento de partes y materiales, así como la selección de la maquinaria.

El factor humano también debe ser considerado, debido a que un sistema FMS no es ciento por ciento autónomo, el sistema requerirá mantenimientos, reprogramación de máquinas CNC, así como la operación del sistema de control supervisorio.

3.1 Clasificación de sistemas FMS

Los sistemas de manufactura flexible son clasificados según el concepto de flexibilidad. De acuerdo a [**2**], la flexibilidad de un sistema de manufactura se puede valorar en los siguientes aspectos:

- Flexibilidad de máquinas: facilidad para realizar cambios para producir un conjunto de tipos de partes.
- Flexibilidad de proceso: habilidad de producir un conjunto de partes en distintas formas.
- Flexibilidad de producto: habilidad de cambiar a nuevos productos en forma económica y rápida.
- Flexibilidad de direccionamiento: habilidad de manejo de paros y fallas, con la finalidad de continuar produciendo un conjunto de tipos de partes.
- Flexibilidad de volumen: habilidad de operar a distintos volúmenes de producción.
- Expansión de flexibilidad: habilidad de expandir el sistema en forma modular y de manera sencilla.
- Flexibilidad de operaciones: habilidad de intercambiar el ordenamiento de algunas operaciones para cada tipo de producto.
- Flexibilidad de producción: total de partes que un sistema de manufactura flexible puede producir.

La tabla uno muestra la clasificación de los sistemas de manufactura .Por otra parte, la figura 1, es una ilustración de cómo se superponen los FMS en función de la cantidad de partes a producir y su volumen.

TABLA N° 1: CLASIFICACIÓN DE SISTEMA DE MANUFACTURA.

Sistema de manufactura	**Nivel de flexibilidad**	**Número de partes dentro de una familia**	**Tamaño promedio del Lote**
Transferencia en línea	Baja	1-2	Arriba de 7000
FMS dedicados	Media	3-10	1000-10000
FMS secuenciales	Media	4-50	50-2000
Celdas de Manufactura	Media	30-500	20-500
Máquinas CN (Job Shop)	Alta	Arriba de 200	1-50

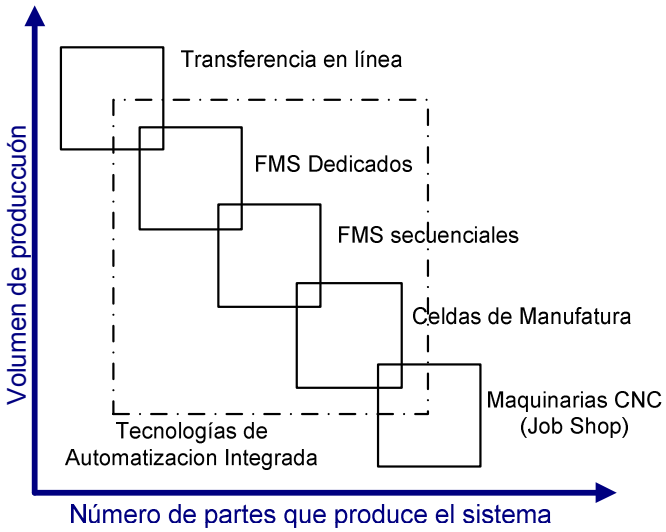


Figura 1: Clasificación de FMS según el volumen y el número de partes que produce (MS Visio 2007).

3.2 Problemática de los FMS.

La tecnología empleada para coordinar y controlar los componentes de un FMS, así como construir un sistema de trabajo eficiente y de alto desempeño, usualmente se basa en Controladores Lógicos Programables (PLC) y/o computadores industriales. Con este hardware, la ingeniería del país debería programar algoritmos de control supervisorio para dirigir el itinerario del proceso, las tareas de entrega y flujo de materiales, así como la solución de los conflictos y bloqueos en el sistema, sin embargo esta área es deficiente. El bloqueo es la asignación de recursos a tareas especificas, que generan una situación del sistema del cual no se puede salir o evolucionar a la ejecución de nuevas tareas.

En un contexto global, la elaboración de algoritmos de control efectivo, en el establecimiento del itinerario de operaciones en manufactura, representa un problema central de investigación en el área de sistemas dinámicos de eventos discretos (DEDS). La necesidad de comprender un sistema DEDS, modelarlo, analizarlo, controlarlo y simularlo, ha conducido a la investigación de mejores metodologías y herramientas. El método de aproximación basada en matrices es uno de estos esfuerzos que busca obtener una solución definitiva al problema.

Ahora bien, surge la pregunta, ¿Todo este avance tecnológico en la automatización de aplicaciones industriales en manufactura puede ser dirigido hacia la pequeña y mediana empresa para aprovechar las nuevas metodologías y mejores herramientas?

Desde un punto de vista de sociedad global, una manufactura eficiente y efectiva resulta de importancia para un desarrollo sostenible. La eficiencia en el uso de los recursos radica en una cuidadosa planeación de la tecnología de producción, procesos productivos altamente automatizados y equipos configurados para propósitos especiales o de necesidades particulares. Esto ha resultado cierto para procesos productivos de alto volumen, pero ¿Es posible hacerlo extensivo en parte hacia la mediana y pequeña empresa, dado el estado actual de la tecnología? ¿Es posible aplicar herramientas que permitan evaluar alternativas de bajo costo para volúmenes de producción relativamente moderados con una mezcla de distintos productos?

En este sentido, el presente proyecto buscó generar conocimiento de implementación, es decir el *¿Know how?* de cómo diseñar el algoritmo de control para dos Celdas Flexibles de Manufactura y como implementar dichos algoritmos en un autómata programable. Los artículos y libros que versan sobre el método, lo hacen a nivel teórico, plantean las ecuaciones de las FMS, las resuelven y posteriormente muestran el resultado de las ecuaciones, que describe el algoritmo de control, es importante mencionar que no indican ningún procedimiento de cómo implementar las ecuaciones en un PLC. Además, las implementaciones del método que aparecen en Mirelles, 2001 [**4**] y en Bogdan, 2002 [**5**], son para sistemas del tipo multi reentrantes (MRF), por lo que la teoría se puede aplicar en forma directa, sin embargo en sistemas FMRF, las experiencias son nulas, solo se ha planteado a nivel teórico como resolver el caso de sistemas FMRF, ver Mirelles [**6**], sin ejemplo práctico, y más actualmente en el 2008, Badall [**7**], ha trabajo con sistemas de distribuido de sensores tipo FMRF, pero no sistemas de manufactura por lo que no hay experiencia concreta.

Actualmente los algoritmos de control de los sistemas FMS que se importan al país, son "cajas negras" para los ingenieros y académicos costarricenses, dado que las carreras de ingeniería costarricense, los métodos para diseñar algoritmos de control para celdas de manufactura flexible son pobres. Este proyecto fortalece en forma directa la academia e ingeniería del país, por cuento se logró implementar y desarrollar el control supervisor (DEC) de un Sistema de Manufactura Flexible (FMS) para dos celdas de manufactura.

3.3 Descripción de la celda teórica.

En el artículo descrito llamado "*Sensor based stage Petri net modelling of PLC logic programs for discrete event control design*", se plantea la distribución física y secuencia de funcionamiento de una celda de manufactura. Dicha celda consta de dos bandas transportadoras, una máquina de control numérico CNC, un centro de manufactura vertical VMC, un robot, un *buffer* o almacén de tamaño uno, además del controlador lógico programable PLC. La celda de manufactura flexible FMC, a su vez cuenta con 14 sensores que monitorean eventos de inicio o finalización de actividades. En la figura 2 se muestra una representación de la distribución física del la celda de manufactura.

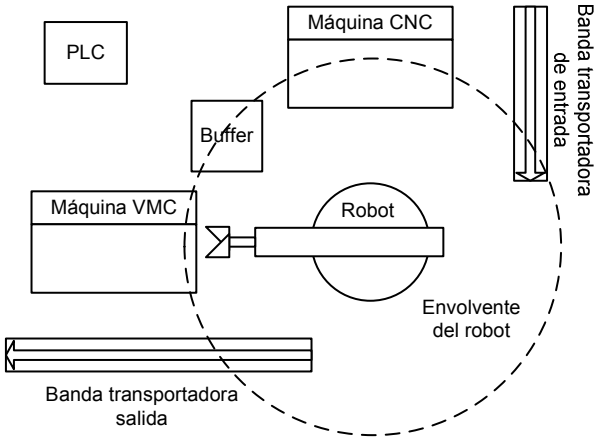


Figura 2: Layout de una celda de manufactura planteada en [1]. (MS Visio 2007).

La celda de manufactura descrita debe procesar tres piezas distintas llamadas {A,B,C}. Las piezas arriban a la banda transportada en forma individual hasta el final de la banda transportadora. Por ejemplo si la pieza A llega al final de la banda, el controlador de la celda (PLC) envía una señal al robot para que recoja la pieza y la cargue a la máquina CNC (se ejecuta el programa de robot

$RP_{AC\text{-}CNC}$). Una vez montada la pieza A en la máquina CNC el robot retorna a su posición origen (*home*), posteriormente la puerta se cierra y se inicia la ejecución del código numérico para la pieza A llamado NC_A . Una vez que el maquinado esta completo la puerta se abre, en este instante y si el robot está disponible, se ejecuta el programa de descarga de la pieza A hacia la banda transportadora de salida ($RP_{A\text{-}TB2}$).

Las piezas B arriban por la banda transportadora y el controlador debe indicarle al robot que tome la pieza ($RP_{B\text{-}VMC}$) y la coloque en el centro vertical de maquinado (VMC). Una vez colocado la pieza el robot retorna a origen, la máquina VMC cierra la puerta e inicia la ejecución del programa numérico NC_B. Una vez finalizado el maquinado se abre la puerta y si el robot está disponible se toma la pieza B ($RP_{BC\text{-}TB2}$) y se descarga en la banda transportadora de salida, después de esto el robot retorna a la posición de origen.

Finalmente las piezas tipos C reciben un tratamiento similar a las piezas A y B, sólo que esta vez estas piezas pasarán tanto por la máquinas CNC y VMC. Si existe una pieza tipo C lista para procesamiento, y tanto el robot como la máquina CNC están disponibles, entonces el controlador le indica al robot coloque la pieza C en la CNC mediante el programa $RP_{C\text{-}CNC.}$ Una vez que el trabajo está listo y tanto el robot como la máquina VMC estén libres, entonces se carga la pieza C en VMC a usando el programa $RP_{C\text{-}VMC}$. En caso que la máquina VMC este en operación se almacenará la pieza en el almacén (buffer) en espera que el recurso se libere. Suponiendo que la máquina VMC haya finalizado la operación sobre la pieza C, el controlador le indica al robot que descargue la pieza en la banda mediante el programa $RP_{BC\text{-}TB2}$.

3.4 Descripción de la celda en laboratorios de sistemas modernos.

El proceso de la figura 3, realiza el trabajó de ensamble de una caja de engranes de prueba. La caja consiste de 4 partes, la base de la caja, dos engranes y la tapa del conjunto.

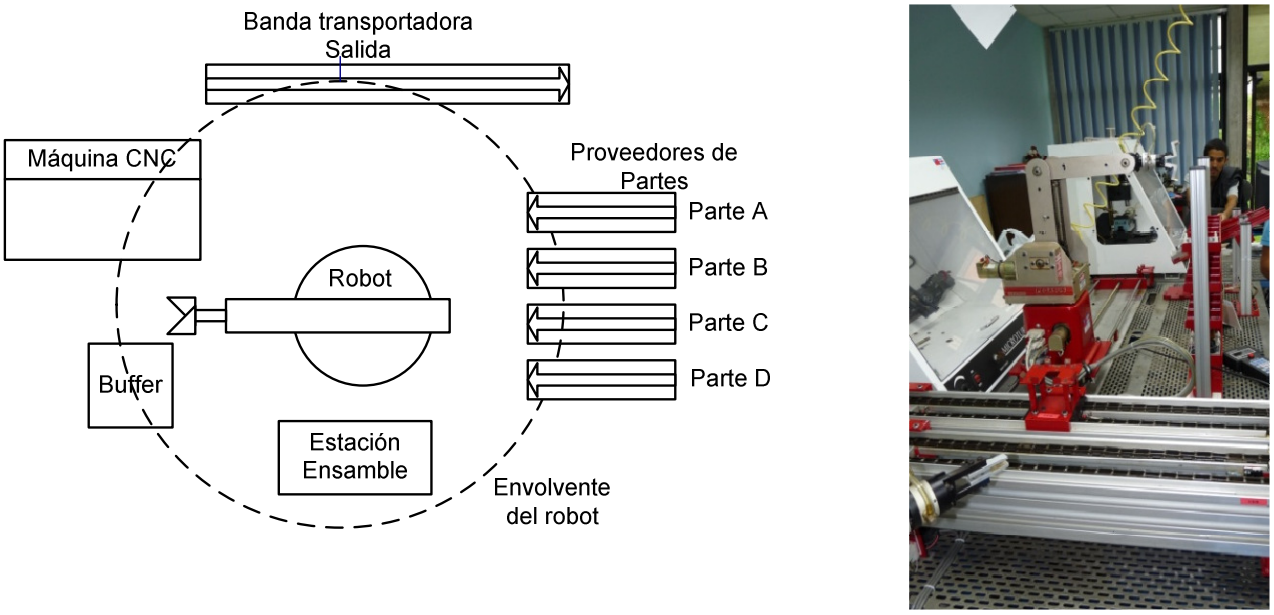


Figura 3: Layout de la celda de manufactura con su fotografía

El proceso consiste en tomar un acrílico de la banda transportadora y ponerlo en la CNC para ser maquinado, cuando esté listo ponerlo en un buffer para ser utilizado después. Además hay que colocar la caja en la mesa de ensamble y después colocarle los dos engranes. Finalmente se toma el acrílico del buffer para colocarlo en el sub-ensamble y se lleva el producto terminado hacia la banda transportadora para su salida.

El proceso no necesariamente se realizará en forma secuencial, dependiendo de la política de control elegida y de las entradas que reciba el sistema. La celda a automatizar es del tipo MRF y lo que se pretende es implementar un controlador basado en el modelo matricial de un proceso multi-reentrante usando políticas LBFS y FBFS, estos resultados se comparan contra una programación secuencial del mismo proceso.

4 OBJETIVOS (GENERAL Y ESPECÍFICOS)

Objetivo General:

Diseñar e implementar dos algoritmos de control para dos FMS, usando el método aproximación basada en matrices.

Objetivos Específicos:

1. Desarrollar un algoritmo de control para una FMS teórica, descrita en [**1**].
2. Desarrollar un algoritmo de control para una FMS de laboratorio.
3. Determinar la dinámica del sistema FMS mediante simulación.
4. Implementar los dos algoritmos de control para la FMS en el PLC S-200 o S-300.
5. Establecer una forma alternativa de programación para algoritmos matriciales, distinto a como se plantea en la literatura [**2**], [**5**], [**8**] .

4.1 Cumplimiento de objetivos

Se considera que todos los objetivos planteados se han cumplido a cabalidad, sin embargo algunos de los objetivos específicos han sufrido variaciones. Por ejemplo el objetivo 4, plantea implementar el algoritmo de control en PLC S-200 o S-300, sin embargo el PLC utilizado fue el RS LogiX 5330, este cambio se debió a que la celda de manufactura seleccionada en el laboratorio de sistemas de manufactura poseía este autómata y no era factible realizar el cambio. Sin embargo, la funcionalidad y el lenguaje de programación de ambas computadores industriales es semejante, ya que son sistemas competidores, los primeros marca SIEMENS y el último marca Allen-Bradley.

Este proyecto no solo cumple los objetivos, los excede. Esto porque propone reglas teóricas de cómo controlar sistemas de manufactura híbridos entre sistemas MRF y FMRF. Esto debido a que en el desarrollo del proyecto, se determinó que el método matricial es de implementación directa a sistemas MRF, pero no a los híbridos, como el que se plantea en [**1**].

5 REVISIÓN LITERARIA.

5.1 Aproximación Basada en Matrices

La aproximación basada en matrices, se puede entender como un conjunto de reglas lógicas que plasman la dinámica del sistema discreto, de tal forma que se logra describir la relación entre los distintos recursos, las operaciones que realizan, las señales de control que generan y el control de las partes dentro del sistema de manufactura flexible (FMS). Este conjunto de reglas lógicas para la FMS, son establecidas en forma de matrices booleanas, que se relacionan en forma algebraica y describen en forma precisa la naturaleza de un sistema de eventos discretos. El método se basa en la obtención de las ecuaciones matemáticas del sistema, que posteriormente se utiliza para construir el algoritmo de control que se implementará por ejemplo en un autómata programable.

Las ecuaciones para el desarrollo del algoritmo del controlador del FMS, se muestran a continuación y se describen ampliamente en [**2**], [**9**], [**10**]:

Matriz de estado del sistema

$$\overline{x} = \boldsymbol{F_v}\Delta\overline{v_c}\nabla\boldsymbol{F_r}\Delta\overline{r_c}\nabla\boldsymbol{F_u}\Delta\overline{u}\nabla\boldsymbol{F_D}\Delta\overline{u_D} \quad (1)$$

Ecuación de inicio de trabajos:

$$v_s = \boldsymbol{S_v}\Delta x \quad (2)$$

Ecuación de liberación de recursos:

$$r_s = \boldsymbol{S_r}\Delta x \quad (3)$$

Ecuación de productos salientes de la celda:

$$y = \boldsymbol{S_y}\Delta x \quad (4)$$

Donde cada matriz significa lo siguiente:

- $\boldsymbol{F_v}$ matriz que establece la secuencia de trabajo, esta matriz muestra el orden de las tareas del proceso de ensamble.
- $\boldsymbol{F_r}$ matriz que establece los requerimientos de recursos, muestra los recursos necesarios para llevar a cabo cada tarea, por tanto relaciona las tareas ejecutadas por cada recurso.
- $\boldsymbol{S_v}$ matriz de inicio de tareas, indica para cada tarea cuáles son sus tareas previas.
- $\boldsymbol{S_r}$ matriz de liberación de recursos, indica cuales tareas una vez que finalizan liberan un determinado recurso
- $\boldsymbol{F_u}$ matriz que establece la lógica de partes entrantes al sistema.
- $\boldsymbol{S_y}$ matriz que declara la lógica de salida para los productos.
- $\boldsymbol{F_D}$ matriz de control de despacho.

Los vectores se asocian al estado de FMS y a los sensores dentro de la celda, de tal forma que:

- *x* es el vector que describe las reglas lógicas que están activas.
- *v* es el vector que indica cuales tareas se han terminado.
- *r* es el vector que indica cuales recursos libres.
- *u* es el vector que indica el producto entrante en la celda FMS.
- u_D es el vector que establece las políticas de cómo despachar una parte o subproducto.

Finalmente, la simbología usada tiene el siguiente significado:

Δ es el símbolo para la conectiva lógica AND entre dos matrices booleanas

∇ es el símbolo para la conectiva lógica OR entre dos matrices booleanas.

Los dos operadores matriciales se desarrollan de la siguiente forma, si A y B son matrices boleadas, de tamaño n x m y m x n respectivamente. La operación C= AΔB, se resuelve para la celda $c_{ij} = (a_{i1} \wedge b_{1j}) \vee (a_{i2} \wedge b_{2j}) \vee \cdots \vee (a_{im} \wedge b_{mj})$. Por otra parte la operación C= A∇B, se resuelve para la celda $c_{ij} = (a_{ij} \vee b_{ij})$. Donde el símbolo ∧ denota la conectiva lógica Y y el símbolo ∨ representa la conectiva lógica O. La barra superior sobre un vector denota la negación de de todas las variables del vector.

La figura 4, muestra un esquema entre el controlador y la celda de manufactura . El controlador debe ser programado de acuerdo a las ecuaciones 1,2,3,4, con la finalidad de obtener un algoritmo que permita administrar adecuadamente las tareas y recursos de la celda, con la finalidad de evitar las siguientes dos condiciones: conflictos por un mismo recurso o bloqueo de recursos en el sistema.

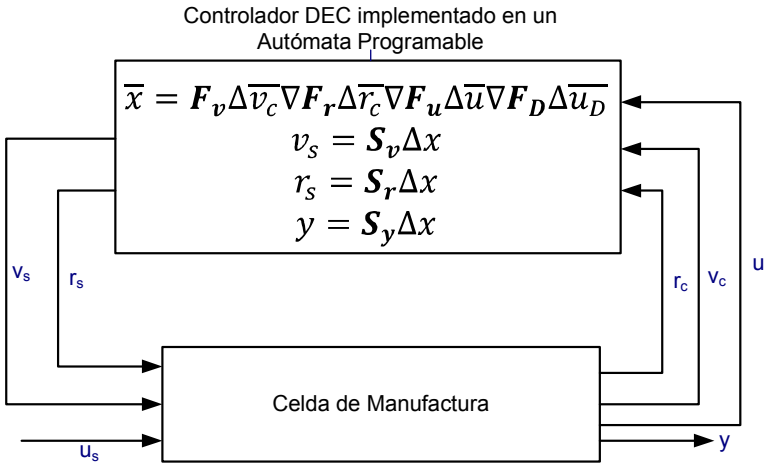


Figura 4. Relación entre el autómata programable y la celda flexible de manufactura (MS Visio 2007).

Formalmente en Huang, 2001 [**9**] y en Lewis y otros (1998) [**10**], definen las matrices utilizadas por el método, e indican que la matriz $\mathbf{F_v}$ y $\mathbf{F_r}$ son utilizadas en la planificación de producción desde 1962 y 1973 respectivamente, por lo que no representan una novedad alguna. Sin embargo la utilización de estas matrices en el método les da un carácter más formal. La matriz $\mathbf{F_v}$ describe la secuencia de tareas de la celda. Se dice que para celdas tipo MRF, si hay un uno en la posición (i,j) significa que la tarea j es requisito de una tarea i, siendo j columnas y i filas. En general y en el contexto del método, se dice que $\mathbf{F_v}(i,j)= 1$ si la tarea j contribuye a la construcción del i-ésimo componente del vector lógico de estados **x**. Cuando la celda es de tipo FMRF, existirá al menos una tarea que contribuye a la construcción de dos o más variables del vector **x**. Para cualquier otro caso $\mathbf{F_v}(i,j)= 0$. La matriz $\mathbf{F_r}$ define la secuencia de los recursos de la celda, la matriz $\mathbf{F_r}(i,j)= 1$ si el recurso j contribuye a la construcción i-ésimo componente del vector de estados, y $\mathbf{F_r}(i,j)= 0$ para cualquier otro caso. El significado $\mathbf{F_r}$ establece cuales recursos j son requeridos por las tareas i.

La matriz $\mathbf{S_v}$ se conoce como arreglo de inicio de trabajos, esta matriz relaciona el vector lógico de estados **x** y el conjunto de trabajos, se dice que $\mathbf{S_v}(i,j)=1$ si el j-ésimo componente del vector de estados es requisito para el inicio de la tarea i, para cualquier otra combinación $\mathbf{S_v}(i,j)=0$. La matriz $\mathbf{S_r}$ relaciona el vector de estados **x** y el conjunto de recursos, si $\mathbf{S_r}(i,j)=1$el j-ésimo componente del vector de estados es requisito para liberar el recurso i, si $\mathbf{S_r}(i,j)=0$ no existe relación entre el vector y el recurso. A continuación se muestran las matrices para la celda.

5.2 Celda multi re-entrantes: consideraciones generales.

Para efectos del método matricial es necesario definir recursos, tareas y partes producidas, la notación usada aparece descritas en [**2**], [**11**], [**12**]. El símbolo **Π** representará el conjunto de partes procesadas en la FMS, para el ejemplo que se desarrollará **Π**={A, B, C}, cada parte k es caracterizada por una secuencia de operaciones $J^k=\{J_1^k, J_2^k, J_3^k, ..., J_L^k\}$. El conjunto de recursos r_i de la celda es llamado *R*, a su vez un ordenamiento particular de recursos r_i son los encargados de ejecutar las operaciones de la secuencia J^k, los recursos utilizados en esa secuencia de trabajos particular se representan como R^k, a su vez R^k deben ser un subconjunto de *R*, es decir $R^k \subset R$. De esta forma $R = \bigcup_{k\in\Pi} R^k$ y $J = \bigcup_{k\in\Pi} J^k$ representan todos los recurso y tareas de la celda de manufactura.

Un recurso compartido r_i es aquel que puede ser utilizado para desempeñar más de una tarea distinta sobre la pieza k, es decir: $r_i \subset R^k$, y $J_i^k \cap J_j^k = \emptyset$. En el caso de un recurso secuencialmente compartido las tareas realizadas por un recurso compartido r_s son más de una, es decir $|J(r_s)|>1$ si $r_s \epsilon R_s$, donde R_s es el conjunto de recursos compartidos. Dado que los recursos compartidos son los elementos que permiten realizar tareas distintas, se dice que el flujo de tareas de la parte *k* a procesar está re-entrando al recurso. Las celdas de manufactura de este tipo son conocidas como MRF (*Multiple Re-entrant Flowlines*) dado que cumplen esta característica de poseer recursos compartidos, un análisis formal de un sistema MRF se puede encontrar en Huang 1996 [**13**], así como su programación para propósitos de simulación con MATLAB en Taconni, 1997 [**14**].

Las celdas de manufactura MRF se caracterizan porque para una secuencia causal de tareas $J^{k,}$ cada una es realizada por el recurso r_i, pero existen recursos R_s que desempeñan varias tareas. En las celdas MRF no existe procesos de ensamble de partes, ni la escogencia de rutas alternas de tareas, es decir, para desempeñar una tarea J_i^k se asigna un único recurso r_i, sin embargo si $r_i \epsilon R_s$ entonces este recurso puede desempeñar otra tarea llámese J_s^k . En este tipo de celda no existen secuencias de tareas en paralelos para obtener el producto final, sólo existe la secuencia de tareas predefinida.

Existe un caso más general de celdas de manufactura llamadas FMRF (Free choice *Multiple Re-entrant Flowlines*), estas celdas poseen las siguientes dos características según Mirelles 2002 [**6**] y Badall, 2007 [**15**].

- la celda posee dos o más recursos r_i, que pueden desempeñar la tarea J_i^k, y cada recurso r_i puede desempeñar varias tareas, ya que $r_i \epsilon R_s$.
- La celda puede presentar múltiples rutas de trabajos, estas rutas no son determinísticas, y dependen del estado de ocupación de los recursos en la celda.

Independiente del tipo de celda utilizada, el diseño del controlador para una celda de manufactura, asume que la celda cumple con las siguientes características funcionales:

i. Tareas sin derecho a prioridad, una vez asignado un recurso r_i a la tarea J_i^k, el recurso no puede ser removido a otra tarea J_j^k, hasta que la tarea J_i^k haya finalizado.

ii. Exclusión mutua, un recurso r_i solo puede ser usado sólo para una tarea J_i^k a la vez.
iii. Mantener mientras espera, un proceso mantiene los recursos, hasta que pueda tener todos los recursos necesarios para ejecutar la tarea.
iv. No se considera fallas de máquinas en la celda.

Las características funcionales anteriores, i),ii),iii) de la celda de manufactura, son las mismas condiciones que deben existir para que exista bloqueos en un sistema operativo computacionales, esto se deduce al comparar las condiciones de bloqueo que aparecen Silberschatz, pag 209, [**16**]. La condición que hace falta para que exista el bloqueo de recursos, es la existencia de esperas circulares de recursos en el sistema, esto a su vez implica en el caso de celdas MRF y FMRF, que una adecuada administración de los recursos compartidos impide el bloqueo. Es importante mencionar que las técnicas de análisis para sistemas MRF no aplican en FMRF, desacuerdo al trabajo de Ballal, 2007 [**15**], y la mayoría de trabajos en el tema, como se verá más adelante, aplican el método basado en matrices a celdas de manufactura tipo MRF de tamaño pequeño.

5.3 Redes de Petri.

Las Redes de Petri clásicas se conciben como un grafo dirigido que posee dos tipos de nodos principales: los *lugares* representados por círculos y las *transiciones* representadas por barras rectangulares, ver figura 5. Entre los nodos se ubican los arcos dirigidos, los cuales se encargan de unir las transiciones con los lugares y viceversa. Cada arco dirigido posee un número que indica el peso del mismo, este peso determina la cantidad de *marcas* que consume de un lugar o deposita a un lugar, siempre y cuando se haya disparado una transición habilitada. Los arcos dirigidos sin número se entiende que consumen o depositan una marca. Las marcas o "tokens" se representan en forma gráfica como puntos negros que se ubican dentro de cada nodo lugar.

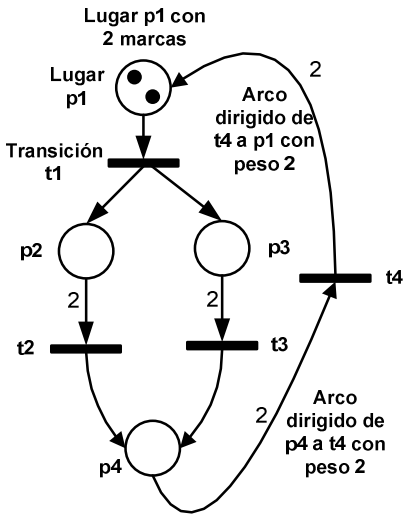


Figura 5. Red de petri, con cuatro lugares, cuatro transiciones, cinco arcos dirigidos de peso uno, y cuatro arcos dirigidos de peso dos. Visio 2007.

Formalmente, una Red de Petri se define como una quintupla,

$$RdP = (P,T,F,W,M_0) \quad (5)$$

Donde:

$P=\{p_1,p_2,\ldots,p_m\}$ es un conjunto finito de lugares.

$T=\{ t_1,t_2,\ldots,t_n \}$ es un conjunto finito de transiciones.

$F \subseteq (P \times T) \cup (T \times P)$ es un conjunto de arcos dirigidos.

$W:F \rightarrow \{1, 2, 3, \ldots\}$ es una función de pesos de los arcos.

$M_0:P \rightarrow \{1, 2, 3, \ldots\}$ es el marcado inicial de la red.

$P \cap T = \emptyset$ y $P \cup T \neq \emptyset$.

El marcado inicial de una RdP son las marcas que posee cada lugar de la red en su inicio. Una red de Petri con la estructura (P, T, F, W) sin especificar su marcado inicial es denotada por N. Por otro lado, una PN con un marcado inicial dado es denotado por $PN=(N, M_0)$

El comportamiento de los sistemas puede ser descrito en términos de sus estados y sus cambios. En las redes de Petri el estado del sistema, o mejor dicho, el marcado de la PN, cambia de acuerdo a las siguientes reglas de disparo o transición:

- Se dice que una transición es habilitada si cada lugar de entrada p de t es marcada con al menos w(p,t) marcas, donde w(p,t) es el peso del arco de p a t.
- Una transición habilitada puede o no ser disparada (esto depende solamente del carácter no determinístico del evento).
- El disparo de una transición t habilitada remueve w(p,t) marcas de cada lugar de entrada p de t y agrega w(t,p) marcas a cada lugar de salida p de t, donde w(t,p) es el peso de los arcos de t a p.

Para un completo tutorial sobre las propiedades estáticas y dinámicas de las RdP, ver Murata, 1989 [**17**].

Si bien el modelo matricial representa al sistema adecuadamente, no es necesario realizar las operaciones indicadas anteriormente para calcular el vector de estados. El vector resultante corresponde a cada transición de la Red de Petri, donde se escriben las tareas y recursos necesarios para cada transición. Por lo que tanto el método matricial como la RdP son representaciones matemáticas equivalentes.

5.4 Bloqueo en celdas de manufactura.

Para celdas de manufactura tipo MRF y FMRF la estrategia de control de los recursos se basa en evitar el bloqueo del sistema, buscando la mayor utilización de los recursos y estableciendo una cantidad máxima de tareas a realizar. Huang en 1996 [**13**], se establece el concepto de sifones críticos, y que limitando las tareas de ese sifón, se evita el bloqueo de ese sifón. El concepto de sifón se refiere a los recursos que se encuentra en espera circular (se define adelante) y a las tarea que desempeñan esos recursos que hacen que se bloque el sistema. En dicho trabajo se establece los teoremas uno y dos del presente trabajo, donde se definen los criterios estructurales que se debe cumplir para que la celda de manufactura no entre en bloqueos. Además se analizan distintas política de asignación de recursos para el no bloqueo, entre las cuales se encuentran las llamadas: LBFS (last buffer first served), FBFS (fist buffer first served), MAXWIP (max work in process), etc.

***Teorema 1**: Definición y construcción del Sifón Crítico*

$$Sc = C \cap J(C)_+ = C + \sum_{i=0}^{n} \{\bullet\{\bullet r_{si}\}_+ \cap J\} \tag{6}$$

donde:

Sc es el conjunto de recursos y tareas de la celda que forma el sifón crítico.

C son las los recursos que componen una espera circular.

J tareas de la celda de manufactura.

$J(C)_+$ *tareas que realizan los recursos C, cuyos arcos de salida no apuntan a nodos ubicados dentro de la espera circular C.*

r_{si} *son los recursos compartidos dentro de C.*

$\{\bullet r_{si}\}_+$ *son el conjunto de transiciones cuyos arcos apuntan a* r_{si} *y no tienen arcos de entrada de otros recursos dentro de la espera circular C.*

$\bullet\{\bullet r_{si}\}_+$ *son el conjunto de tareas cuyos arcos de salida apuntan a las transiciones definidas por* $\{\bullet r_{si}\}_+$.

La notación de puntos proviene del contexto de Redes de Petri (RdP), por ejemplo sea P un nodo tipo lugar, la notación con puntos ●P, sirve para denotar los nodos anteriores a P, por lo que se brindan los nodos tipo transición. La notación ●●P denota los nodos tras anteriores. De igual forma aplica para los nodos siguientes al actual P● y a los post siguientes P●●. El nodo P puede ser tanto tipo lugar, como del tipo transición. Esta notación se pueden ver en mayor detalle en Murata 1989, [**17**].

Se entiende como espera circular (CW), al conjunto de recursos dentro la celda R que están en una cadena de relaciones de espera, y esta cadena de relaciones es circular. La relación de espera entre dos recursos se denota como ($r_i \rightarrow r_j$) y significa que la tarea r_i no inicia hasta que se libere el recurso r_j. Una espera circular simple CW se denota como ($r_i \rightarrow r_j \rightarrow r_k \rightarrow r_p \rightarrow r_i$), donde los recursos necesarios para la espera deben ser al menos dos. Para este ejemplo el recurso i espera al recurso j, que a su vez espera al recurso k, este a su vez espera al recurso p, y el recurso p espera al recurso i, cuando los recursos conforma un anillo de relaciones de espera, estas se conocen como CW simples.

Si una celda de manufactura tipo MRF o FMRF presentan CW simples se le conoce como sistemas regulares. Si una espera circular está contenida en otra espera circular, los sistemas se le llaman irregulares, ya que pueden presentar el fenómeno de bloqueo en segundo nivel (*second*-level deadlock, SLD). Para sistemas MRF regulares, el bloqueo del sistema está asociado a las esperas circulares, la forma en que se puede evitar los bloqueos en sistemas MFR es la siguiente:

***Teorema 2**: Teorema principal para la evitar bloqueos. Sea C una espera circular llamada y Sc su sifón crítico. Entonces C está es bloqueo circular (CB) si y sólo si el sifón crítico Sc está vacío.*

La obtención de las esperas circular se realiza mediante el algebra de cadenas, esta algebra se explica en los siguientes dos libros Chang [**18**] y en Bogdan [**2**]. La espera circulares se caracterizan porque poseen al menos un recurso compartido, sin embargo es posible obtener las esperas circulares CW a partir del modelado en Red de Petri de la celda. La obtención de las esperas circulares puede realizarse directamente de la red siguiendo en forma inversa los arcos dirigidos entre recursos, partiendo del recurso compartido y regresando a este, con este procedimiento es

posible obtener las esperas circulares simples CWs de la celda. Para las CW compuestas, en sistemas MRF se definen como la combinación de las CW simples a través del recurso compartido, el cual es el elemento de unión entre CWs. A estas nuevas esperas circulares se les conoce como esperas circulas cíclicas (CCW) ya que presenta el fenómeno de SLD. La matriz CW_r representa todas las CW de la celda, cada fila muestra los recursos que están dentro de una espera circular. De esta manera para la matriz $\mathbf{CW_r}$, cada uno significa que el recurso j esta dentro de la i-ésima espera circular, C_i.

Por otra parte, en Ballal, 2007 [**15**], se demuestra porque los criterios anteriores son insuficientes para establecer políticas de control a celdas tipo FMRF, por lo se desarrolla la siguientes enunciados.

Sea C_s el conjunto de todas las esperas circulares simples CWs dentro de la FMRF. Sea p un nodo lugar de decisión con recurso r_p=R(p). El conjunto de todas las CWs dentro C_s que contienen el recurso de decisión r_p se define como:

$$C_p = \{C \in C_s | r_p \in C\} \tag{7}$$

Dada una CW C, el conjunto de los lugares de decisión se definen como:

$$J_{dec}(C) = \{p \in J(C) | \ |p\bullet| \ > 1\} \tag{8}$$

Se define C_p(C) como:

$$C_p(C) = \{C_p | \ p \in J_{dec}(C)\} \tag{9}$$

***Teorema 3**: Dada el conjunto de recursos $C_p(C)$ y sean las tarea $J(C_p(C))_+ \cap J(C_p(C))_o = \emptyset$. Entonces $C_p(C)$ esta en CB si y sólo si el sifón Sc es vacío.*

***Teorema 4**: Si las tareas de la celda presentan la relación $J(C_p(C))_+ \cap J(C_p(C))_o \neq \emptyset$. Entonces $C_p(C)$ nunca estará en bloqueo circular (CB).*

***Teorema 5**: Si $J(C_p(C))_+ \cap J(C_p(C))_o = \emptyset$, entonces la espera circular C esta en CB si y sólo si $C_p(C)$ está en CB.*

Con los teoremas anteriores se establece el siguiente lema:

***Lema 1**:En una FMRF regular, no hay CB en el sistema si y sólo si, para cada C que existe en CW_r , el marcado $m(J(C_p(C)))_0$ es menos que el marcado inicial de $C_p(C)$.*

Note que le lema 1, aplica para sistemas regulares, es decir que no poseen esperas circulares cíclicas (CCW). En Ballal 2008 [**7**], en su tesis doctoral se amplía el concepto a sistemas FMRF irregulares, y se dice que cuando un recurso llave está presente, se debe limitar el trabajo en proceso (WIP, work in process), el subsistema crítico $m(J(C_p(C)))_0$ se debe limitar a $m(C_p(C))-2$. Ballal afirma que el bloqueo se evita limitando el WIP en todos los subsistemas críticos.

5.5 Políticas de despacho para el controlador

Los lugares de control que se denotan como Ud, se colocan antes de todas las transiciones que presentan conflictos con los recursos. Es decir, donde más de una tarea requiere utilizar el mismo recurso. Esto se traduce en que se deben colocar lugares de control antes de todas las transiciones que utilizan un recurso compartido.

En cada corrida del proceso solo se le asigna un "token" a uno de los lugares de control, asegurándose que solo una de las transiciones con recurso compartido será activada. La forma de

asignar los tokens depende de la política de control que se elija, en este trabajó se explicarán las 2 políticas de control implementadas.

5.5.1 LBFS

Última cola, primera en despachar (Last Buffer First Served), consiste en que cada vez que haya un conflicto con un recurso debe primero realizar las operaciones que se encarguen de sacar producto terminado, garantizando de esta manera que se evitaran los bloqueos en el sistema. Por ejemplo, imaginando la siguiente Red de Petri con los lugares de control Ud1, Ud2 y Ud3:

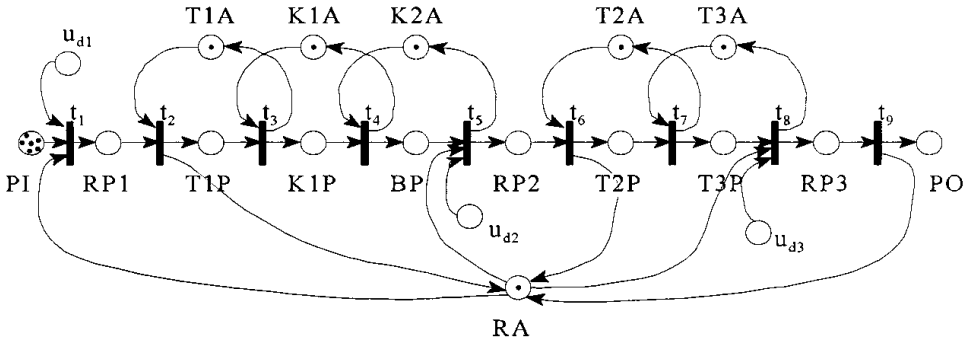


Figura 6. Ejemplo de un proceso sencillo de ensamble, tomado de [5]

La política LBFS indica que en caso de conflicto entre t8, t5 y t1 las transiciones deben dispararse primero las más cercanas a la salida. Es decir, primero activar Ud3, luego Ud2 y al final, cuando no haya conflicto, activar Ud1.

Tiene la ventaja de que garantiza que no habrá bloqueos al darle prioridad a las operaciones al final, y es una política conservadora, además de ser fácil de implementar. Sin embargo, tiene la desventaja de que termina en un proceso prácticamente secuencial y poco óptimo.

5.5.2 FBFS.

Primera cola, primero en despachar (First Buffer First Served). En caso de conflicto le da prioridad a las operaciones en la entrada de piezas, siempre y cuando la realización de esas operaciones no causará un bloqueo. La posibilidad del bloqueo se ve hacia adelante mediante la regla que dice que el número de marcas en las tareas de un sifón crítico debe ser menor al número de marcas en dicha espera circular.

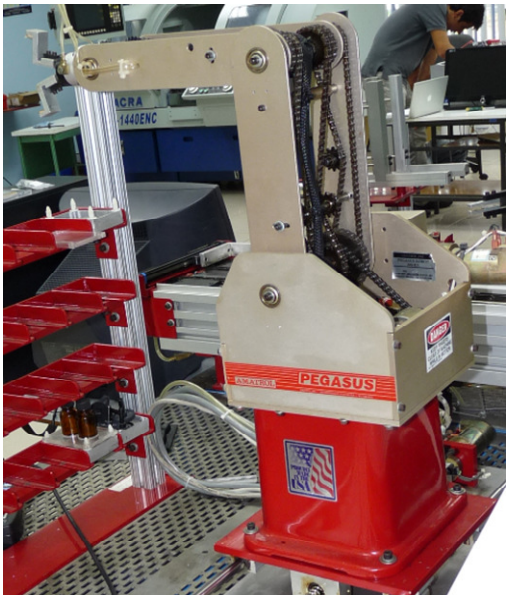
En el ejemplo de la figura 6 esta política, en caso de conflicto, activaría Ud1 siempre y cuando se esté realizando hasta 2 tareas en la espera circular después de X1, Ud2 se activa cuando se está realizando hasta 1 tarea en la espera circular de X5, y de lo contrario se activará Ud3.

6 EQUIPOS Y SOFTWARE

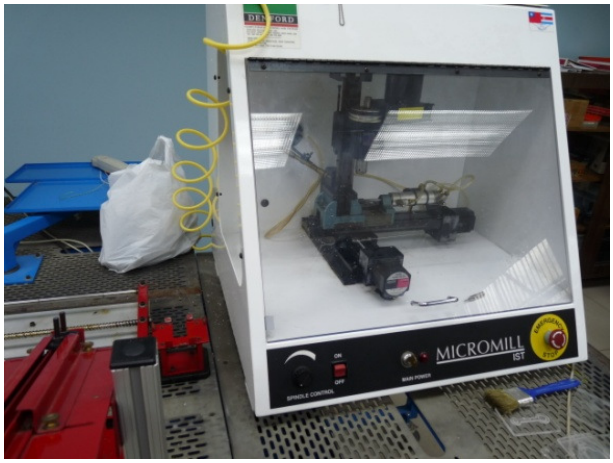
6.1 Equipo de la celda de manufactura de laboratorio.

El proceso se controlará desde un PLC CompactLogix 5330, que cuenta con dos módulos de entradas digitales, 2 módulos de salidas digitales, un módulo de Controlnet y un módulo de comunicación por Ethernet. El PLC se muestra a continuación, Se cuenta con un robot Amatrol Pegasus, que se controla desde una estación de trabajo. La celda de manufactura cuenta con una máquina CNC Micromill IST y una estación de ensamble. También se cuenta con las entradas de cuatro piezas diferentes: El acrílico, la caja de engranes, así como ambos engranes. Finalmente, se cuenta con un buffer para almacenar un acrílico maquinado. Ver fig.7 para más detalle.

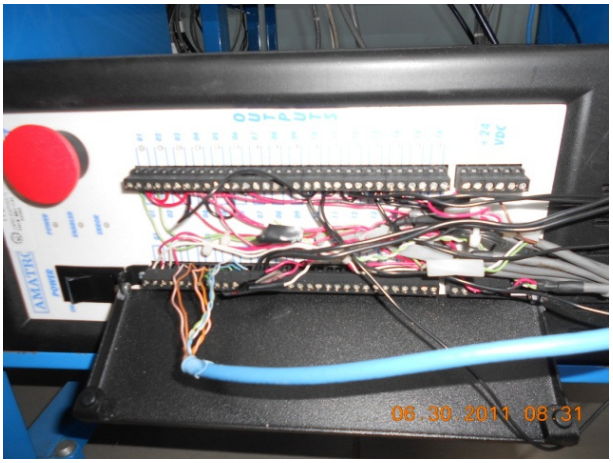
a)



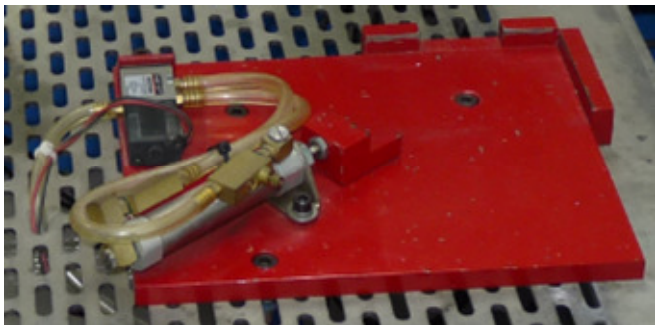
b)



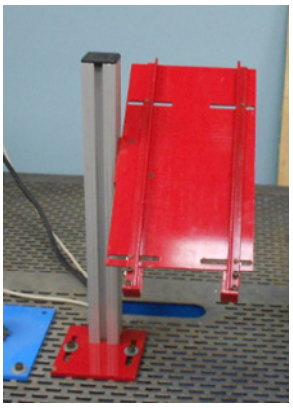
c)



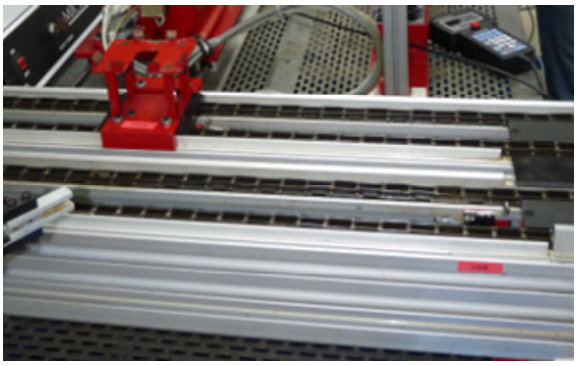
d)



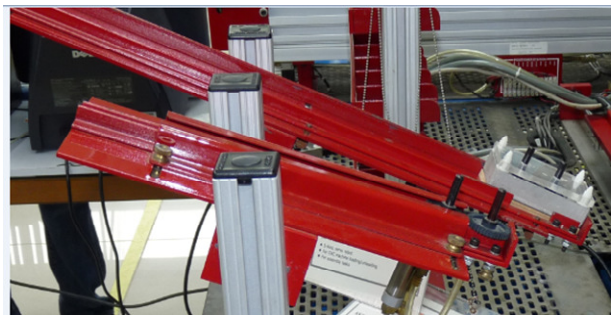
e)



f)



g)



h)

Figura 7. Equipo de la celda de manufactura. a) Controlador lógico programable PLC CompactLogix 5330. b) Robot Pegasus. c) : Driver del robot con puertos de E/S. d) Máquina CNC de la celda. e) Mesa de ensamble para la caja de engranes. f) A almacén (buffer) para el acrílico maquinado. g) Banda transportadora de salida para producto terminado, funciona también como entrada de tapas de acrílico. h) Entrada de las cajas de engranes y de los piñones plásticos.

6.2 Software de simulación

La simulación de los algoritmos se realizó con el software Petri.Net. Este paquete permite simular sistemas de eventos discretos a través de RdP binarias y temporizadas. El software fue desarrollado por el Departamento de Control y Ingeniería de Computación, de la Faculta de Electrotecnia y Computación de la Universidad de Zagreb, más detalle del paquete aparece en el libro [**2**].

La pantalla principal se compone de tres etiquetas llamadas, *PetriNet Editor, Description* y *Response.* En el editor es el ambiente de trabajo donde se configura la RdP, en *Description* se dan las matrices que describen las esperas circulares para sistemas MRF (no para FMRF) y en la etiqueta *Response* se brinda el oscilograma o diagrama de tiempo para cada nodo lugar dentro de la red, porcentajes de utilización y una tabla donde se muestra el estado de la red para cada paso de simulación. Adicionalmente, en la pantalla principal existen tres marcos de herramientas llamados: *Toolbox, Properties, Rules Editor* . El *Toolbox* posee todos los componentes de la red, transiciones, nodos de lugar para representar recursos, operaciones y control, arcos dirigidos, bloques de subsystemas, etc. En *Properties* aparecen las características de cada objeto señalado dentro del *Petri.Net Editor*. Finalmente en *Rules Editor,* se escribe el algoritmo de control para la red, es importante señalar que el editor de reglas solamente soportar reglas escritas con la estructura de

control IF-THEN, por lo que no soporta la estructura de control el ELSE IF. La figura 8 muestra la vista de la interface principal del software.

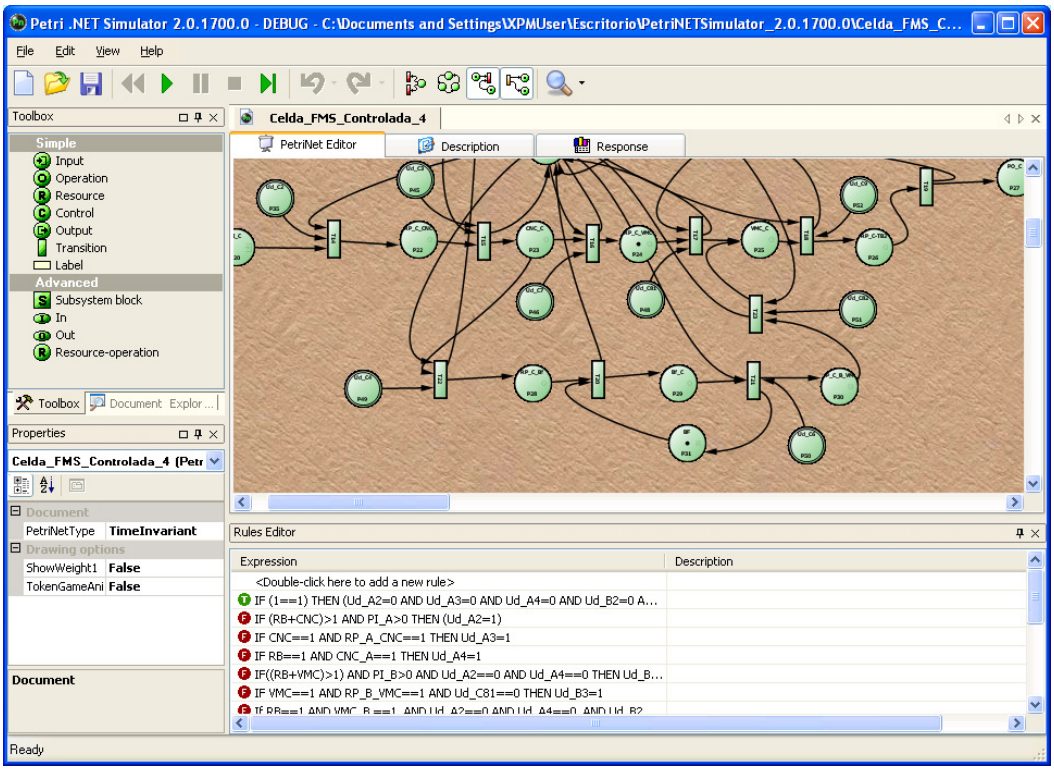
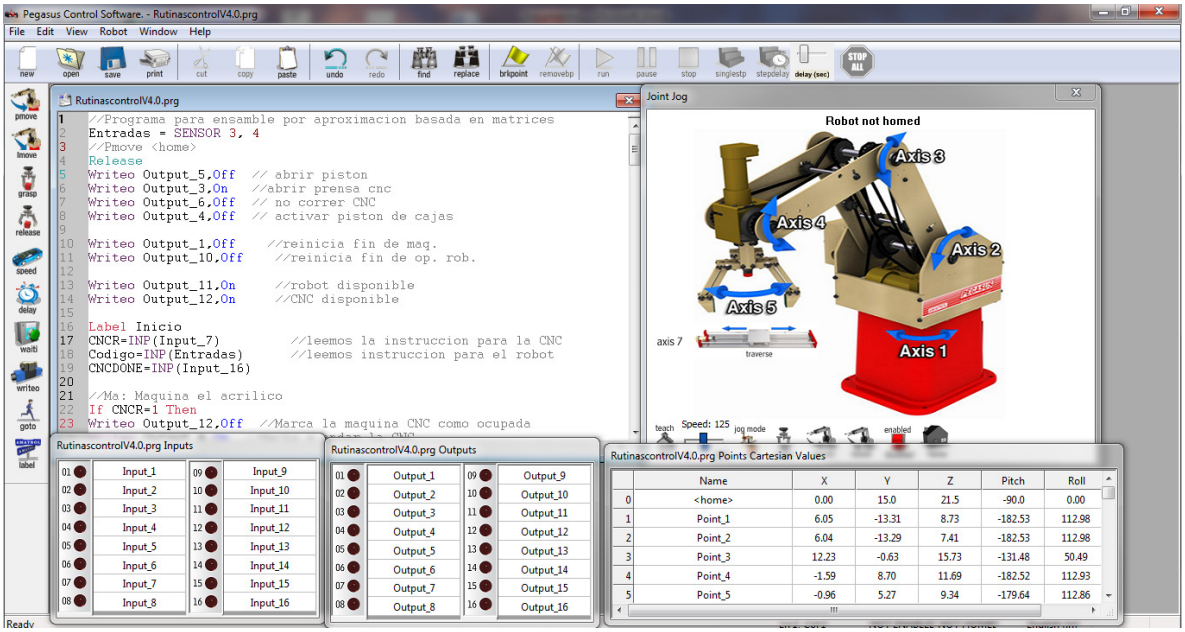


Figura 8. Interface gráfica de usuario principal del software Petri.Net.

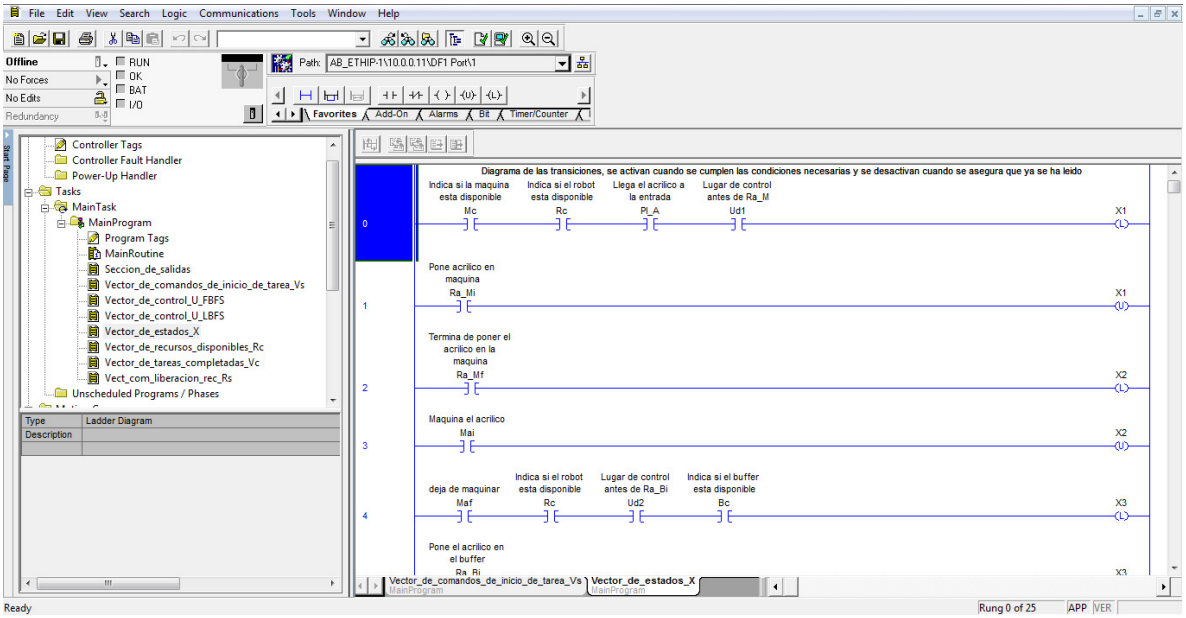
6.3 Software para la implementación de laboratorio.

Para programar el controlador (PLC), basado en la aproximación basada en matrices, en la celda de manufactura del laboratorio. Se programó el PLC RS5330 con el software *RSLogix5000 Pro*. Versión 16.00, y el robot se programó con *Pegasus II Control Software*, Versión 1.1.4.

El software para el PLC se programó en forma modular, donde cada modulo representa una matriz de la ecuación de estados e.c (1). A lo interno de cada modulo se programa en lenguaje escalera LD. Para el controlador del robot este se programa en lenguaje estructurado. La figura 9 muestra las interfaces gráficas de usuarios de ambos software. No se profundiza en la descripción de estos paquetes, ya que son de uso general a nivel de ingeniería de automatización.



a)



b)

Figura 9. a) GUI del software Pegasus para programar el control del robot. b) GUI del RSLogix5000 para programar el controlador de la celda de manufactura.

7 ANALISIS DE LAS CELDAS DE MANUFACTURA

7.1 Pasos de análisis y elaboración del controlador para la celda teórica

Para obtener controlador de la celda teórica, se realizaron los siguientes pasos:

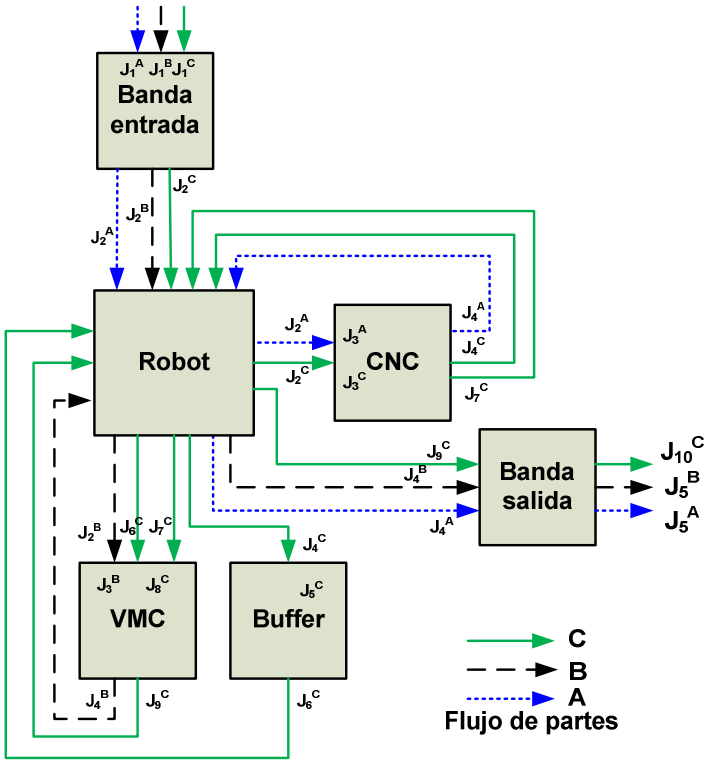
- Análisis de la celda teórica y obtención de matrices.
- Estudio del bloqueos en celdas de manufactura.
- Propuesta de política para la asignación de recursos.
- Propuesta del algoritmo de control.

.

7.1.1 Análisis de la celda teórica y las matrices del controlador.

La obtención de las matrices $\mathbf{F_v}$, $\mathbf{S_v}$, $\mathbf{F_r}$, $\mathbf{S_r}$, $\mathbf{F_d}$ se inicia con la declaración de las tareas y recurso de la celda. Primero se debe entender el árbol de ensamble por cada pieza producida, en este caso las partes **Π** que ingresan a la celda es igual a las piezas producidas**,** por lo se obtendrán tres arboles sin ramas, que representan la secuencia de tareas sobre la pieza.

Normalmente el árbol de ensamble se construye a partir del BOM (*Bill of materials*) definido por los departamentos de producción, sin embargo dado que el problema descrito, no establece la secuencia mediante un BOM, se procede primeramente a construir un diagrama del flujo de las piezas a los largo del celda de manufactura. Este tipo de diagrama permite determinar todas las tareas realizadas por la celda usando como punto de partida, el *layout* y las tareas dadas en el problema.



..

Figura 10. Flujo de partes dentro de la celda. Se observa que la celda descrita en [1], es del tipo FMRF ya que una piezas entran al menos 2 veces en un recurso mismo recurso y existen dos rutas de tareas para la pieza tipo C. La celda consta de 5 recursos, un buffer y realiza 20 tareas. (MS Visio 2007)

La figura 10 muestra el flujo de tareas de la celda. En el diagrama se constata que la celda presenta dos rutas no determinanticas para las piezas tipo C, por tanto la celda es del tipo FMRF. Es importante mencionar que el método aproximación basada en matrices, fue formulado para celdas del tipo MRF, dado que son casos donde es más sencillo determinar el bloqueo del sistema.

En la revisión literaria realizada no se encontró referencias a ejemplos prácticos donde se solucione una celda del tipo FMFR. Todos los casos resueltos en libros y artículos corresponden a celdas MFR: Bogdan y otros (2006) [**2**], Zhou (2004) [**12**], Mirelles y otros (2001) [**4**], Bogdan y otros (2002) [**5**], Bogdan y otros (2004) [**8**]. La teoría de sistemas FMFR se muestra en el trabajo de Ballal, (2007) [**15**] y en su tesis de doctorado Ballal 2008 [**7**], no obstante este trabajo se enfoca en el desarrollo de sistemas cooperativos distribuidos, específicamente redes de robots.

De la figura 11 se puede establecer las siguientes sucesiones de tareas. Para la pieza A esta dada por: J^A={ $TB1_A$, $RP_{A\text{-}CNC}$, CNC_A, $RP_{A\text{-}TB2}$, $TB2_A$}, para la pieza B está dada por J^B={ $TB1_B$, $RP_{B\text{-}VMC}$, VMC_B, $RP_{B\text{-}TB2}$, $TB2_B$}, finalmente para la pieza C la secuencia está dada por J^C={ $TB1_C$, $RP_{C\text{-}CNC}$, CNC_C, $RP_{C\text{-}Bf}$, Bf_C, $RP_{C\text{-}B\text{-}VMC}$, $RP_{C\text{-}VMC}$, VMC_C, $RP_{C\text{-}TB2}$, $TB2_C$ }. Los recursos de la celda son R={TB1, R, CNC, Bf, VMC, TB2} y los recursos compartidos R_s ={TB1, R, CNC, VMC, TB2}. La tabla uno muestra el significado de la nomenclatura para las tareas.

TABLA 2: TAREA A DESEMPEÑAR POR LA CELDA DE MANUFACTURA

	Notación	**Tareas**
J_1^A	$TB1_A$	Banda 1 Transporta la pieza A
J_2^A	$RP_{A\text{-}CNC}$	Robot carga piezas A en la maquina CNC
J_3^A	CNC_A	CNC maquina la pieza A
J_4^A	$RP_{A\text{-}TB2}$	Robot descarga piezas A en la banda transportadora 2
J_5^A	$TB2_A$	Banda 2 transporta la pieza A
J_1^B	$TB1_B$	Banda 1 Transporta la pieza B
J_2^B	$RP_{B\text{-}VMC}$	Robot carga piezas A en la maquina VMC
J_3^B	VMC_B	VMC maquina la pieza B
J_4^B	$RP_{B\text{-}TB2}$	Robot descarga piezas B en la banda transportadora 2
J_5^B	$TB2_B$	Banda 2 transporta la pieza B
J_1^C	$TB1_C$	Banda 1 Transporta la pieza C
J_2^C	$RP_{C\text{-}CNC}$	Robot carga piezas C en la maquina CNC
J_3^C	CNC_C	CNC maquina la pieza C
J_4^C	$RP_{C\text{-}Bf}$	Robot descarga piezas C y carga en *Buffer*
J_5^C	Bf_C	Almacena pieza C en *Buffer*
J_6^C	$RP_{C\text{-}B\text{-}VMC}$	Robot descarga buffer y carga piezas C en la maquina VMC
J_7^C	$RP_{C\text{-}VMC}$	Robot carga piezas C en la maquina VMC
J_8^C	VMC_C	VMC maquina la pieza C
J_9^C	$RP_{C\text{-}TB2}$	Robot descarga piezas C en la banda transportadora 2
J_{10}^C	$TB2_C$	Banda 2 Transporta la pieza C

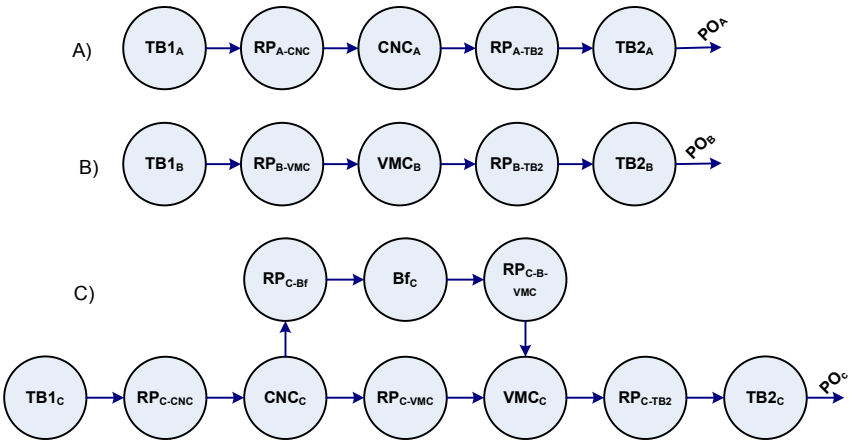


Figura 11. Arboles de ensamble de la celda de manufactura. Las secuencias A), B), C) describen la el orden de las tareas para las piezas A, B, C respectivamente. (MS Visio 2007).

El significado de las siguientes matrices se detalla en la sección 5.1 del presente documento.

Fr =

P2 TB1_A	P3 RP_A-CNC	P4 CNC_A	P5 RP_A-TB2	P6 TB2_A	P9 TB1_B	P10 RP_B-VMC	P11 VMC_B	P12 RB_B-TB2	P13 TB2_B	P21 TB1_C	P22 RB_C-CNC	P23 CNC_C	P24 RB_C-VMC	P25 VMC_C	P26 RB_C-TB2	P28 RP_C-Bf	P29 Bf_C	P30 RP_C-B-VMC	P32 TB2_C	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	T1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	T2
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	T3
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	T4
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	T5
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	T6
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	T7
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	T8
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	T9
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	T10
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	T11
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	T12
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	T13
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	T14
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	T15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	T16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	T17
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	T18
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	T19
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	T20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	T21
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	T22
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	T23
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	T24

Fr =

P15 TB1	P16 RB	P17 CNC	P18 TB2	P19 VMC	P31 BF	
1	0	0	0	0	0	T1
0	1	0	0	0	0	T2
0	0	1	0	0	0	T3
0	1	0	0	0	0	T4
0	0	0	1	0	0	T5
0	0	0	0	0	0	T6
1	0	0	0	0	0	T7
0	1	0	0	0	0	T8
0	0	0	0	1	0	T9
0	1	0	0	0	0	T10
0	0	0	1	0	0	T11
0	0	0	0	0	0	T12
1	0	0	0	0	0	T13
0	1	0	0	0	0	T14
0	0	1	0	0	0	T15
0	1	0	0	0	0	T16
0	0	0	0	1	0	T17
0	1	0	0	0	0	T18
0	0	0	1	0	0	T19
0	0	0	0	0	1	T20
0	1	0	0	0	0	T21
0	1	0	0	0	0	T22
0	0	0	0	1	0	T23
0	0	0	0	0	0	T24

Fw =

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24		
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TB1_A	P2
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RP_A-CNC	P3
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CNC_A	P4
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RP_A-TB2	P5
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TB2_A	P6
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TB1_B	P9
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RP_B-VMC	P10
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	VMC_B	P11
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RB_B-TB2	P12
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TB2_B	P13
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	TB1_C	P21
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	RB_C-CNC	P22
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	CNC_C	P23
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	RB_C-VMC	P24
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	VMC_C	P25
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	RB_C-TB2	P26
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	RP_C-Bf	P28
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	Bf_C	P29
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	RP_C-B-VMC	P30
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	TB2_C	P32

Sr =

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24		
0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	TB1	P15
0	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0	1	1	0	0	1	0	RB	P16
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	CNC	P17
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	TB2	P18
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	VMC	P19
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	BF	P31

7.1.2 Bloqueo de celdas de manufactura.

Para la celda propuesta por Peng [**1**], se puede decir que es del tipo FMRF ya que existe una tarea que habilita dos componentes del vector de estados, que se traduce en la libre escogencia de al menos dos tareas posteriores. Este conflicto tanto en la matriz **Fv,** columna P23, como en la Red de Petri de la figura 12, más información sobre el modelado de celdas de manufactura, ver Silva [**19**] y Zhou [**20**]. Se observa que la tarea CNC_c, posee dos arcos de salida, es decir las tarea posteriores son dos, cumpliéndose que $p \in J, |p\bullet| > 1$. Es importante señalar que la celda en análisis, posee una máquina CNC que posee dos connotaciones distintas, como recurso compartido r_s y como recurso de decisión r_p, es decir el recurso $CNC=r_s=r_p=R(CNC_c)$. Lo anterior es suma relevancia para el estudio, ya que en la revisión literaria no se encontró ningún caso donde se analice una celda FMRF irregular con un recurso decisión, que a su vez es un recurso compartido. ***Es decir, no se han encontrado casos de análisis para sistemas híbridos para FMRF y MRF, como el que se plantea.***

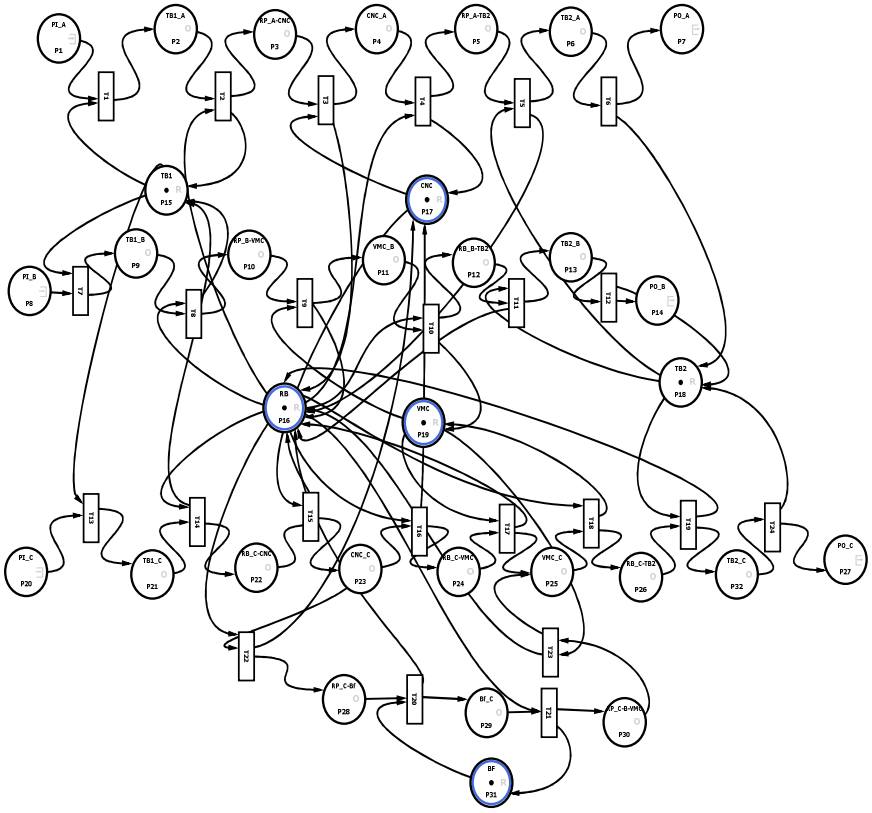


Figura 12. Red de Petri sin controlador. Los círculos con marcado uno representan los recursos del sistema y los círculos sin marcado representan las tareas. (Editor Petri.Net)

Las esperas circulares **CW,** se obtuvieron analizando la Red de Petri de la figura anterior. En (10) se muestra la matriz resultante. El significado de la matriz **CWr,** es de relevancia para la construcción del algoritmo que se propondrá en el siguiente apartado, esta matriz representa el conjunto de esperas circulares simples y compuestas de la celda. Ver mayor detalle en sección 5.4.

$$\boldsymbol{CWr} = \begin{vmatrix} TB1 & TB2 & RB & CNC & VMC & Bf & \\ 0 & 0 & 1 & 1 & 0 & 0 & C_1 \\ 0 & 0 & 1 & 0 & 1 & 0 & C_2 \\ 0 & 0 & 1 & 1 & 1 & 0 & C_3 \\ 0 & 0 & 1 & 0 & 0 & 1 & C_4 \\ 0 & 0 & 1 & 1 & 0 & 1 & C_5 \\ 0 & 0 & 1 & 0 & 1 & 1 & C_6 \\ 0 & 0 & 1 & 1 & 1 & 1 & C_7 \end{vmatrix} \tag{10}$$

7.1.3 Propuesta para la asignación de recursos en la celda de manufactura.

Retomando el método Aproximación Basada en Matrices definida en la ecuación (1), es una ecuación que define para un sistema de eventos discretos representado mediante Redes de Petri, el vector de transiciones de la red. La construcción de la ecuación (1) depende básicamente de cuatro elementos:

a) Los trabajos y su secuencia a ejecutar se representa por la matriz $\mathbf{F_v}$.
b) Los recursos que se requieren para una tarea determinada, matriz $\mathbf{F_r}$.
c) Los vectores de trabajos terminados y recursos libres, v y r respectivamente, que sería señales de sensores dentro de la celda de manufactura.
d) La ley de control define como se controla el sistema, está ley se construye con la matriz $\mathbf{F_d}$ y las reglas de control. u_D. La matiz $\mathbf{F_d}$ define las transiciones de la RdP que se habilitarán con la respectiva regla, y las reglas de control u_D, definen el momento particular de cuando se habilita la transición respectiva.

Lo anterior significa que los elementos que las matrices $\mathbf{F_v}$ y $\mathbf{F_r}$ dependen de la estructura de la RdP y que son nodos lugar que representan tarea y recursos respectivamente, y preceden las transiciones. Los vectores v y r que son señales de la celda de manufactura indicándole al controlador cuando las tareas ha finalizado o cuando los recursos está disponibles, estos vectores solamente depende del tamaño y topología de la red, por lo que es fácil su definición. Los que si resulta complejo es definir una adecuada ley de control que permita optimizar y maximizar la cantidad de trabajos sin que estos bloquen el sistema.

La RdP de la figura 12, muestra la celda de manufactura sin controlador. ***Dado que la celda no es del MRF, ni FMRF puras, y al no encontrarse en literatura nada similar, se propone la siguiente solución del algoritmo de asignación de recursos, a partir de los siguiente lineamientos***:

1. Por cada nodo recurso que sus transiciones posteriores estén en conflicto**,** se establecerán lugares de control que tenga arcos apuntando a la transición respectiva. El nodo lugar de control será llamado u_{D_ki}, donde k es el tipo de parte y el indice i representa el número consecutivo de la tarea asociada a la parte k. Entonces las transiciones **T** en conflicto se pueden establece como:

$$\exists u_{D\_ki}\, \forall x_c \left[\, u_{D\_ki} = \bullet x_c \,\right] \tag{11}$$
$$x_c = \{x \in T \mid \ \bullet x \cap R \neq \emptyset \ \wedge \ |(\bullet x)\bullet| \ > 1\}$$

2. Ahora se analiza los casos de los bloqueos del sistema a través de las esperas circulares CW, para esto se sigue la idea de Badall, 2008, en donde para cada espera circular simple C_i, la cantidad máxima de trabajos que puede soportar una celda MRF es $J(C)_0 = m(C)-1$ y para una celda FMRF los trabajos de la espera circular $J(C_p(C))_0 = m(C_p(C))-2$. Sin embargo encontrar los sifones de tareas $J(C)_0$ y $J(C_p(C))_0$ es laborioso, y se requiere de complejos algoritmos de búsqueda, por lo que se propone el siguiente enunciado.

Para celdas de manufactura donde los recursos desempeñan una única tarea a la vez en un espacio de tiempo dado, los trabajos a ejecutar para cada espera circular simple C_i debe cumplir con:

El marcado de una espera circular simple C_i debe siempre ser mayor a uno para que no exista bloqueo . (12a)

$$m(C_i) > 1$$

Si el marcado de C_i llega a ser uno, entonces el único recurso disponible será el compartido de la espera circular. (12b)

$$m(C) = m(r_{Si}) = 1$$

Y sólo se utilizará en la ejecución de tareas que liberan recursos dentro de la espera circular, haciendo que el marcado de C cumpla (11a) una vez finalizada la tarea $J(r_{Si})$.

Lo anterior se deduce de la ecuación $J(C)_0 = m(C)-1$. Esta ecuación indica que la cantidad de tareas del sifón que se ejecutan dentro de la espera circular *C*, no debe superar la cantidad de recursos disponibles. Si se plantea de la forma $m(C) = J(C)_0+1$, significa que la cantidad de recursos disponibles dentro C, debe ser al menos la cantidad de tareas a ejecutar dentro del sifón más uno, con el fin de evitar bloqueos. Sin embargo esto implica conocer las tareas a ejecutar por cada espera circular. Ahora si se establece la cantidad mínima de recursos disponibles dentro de una espera circular para que no existan bloqueos, se evita conocer el conteo de los trabajos en ejecución, por lo que se obtienen e.c (12a, 12b). **El cambio propuesto consiste en definir la cantidad mínima de recursos disponibles dentro de una espera circular, y no la cantidad máxima de tareas sifón que se puede ejecutar**.

De manera análoga para sistemas FMRF, con esperas circulares compuesta se utiliza el criterio de Badall. Los criterios modificados se utilizarán si (12a) y (12b) no son suficientes para resolver una situación de bloqueo:

El marcado de una espera circular compuesta $C = C_i \cup C_j$, debe ser siempre mayor a dos para que no exista bloqueo (12c)

$$m(C) > 2$$

Si el marcado de C llega a ser dos, se entonces los recursos disponible deberán pertenecer a una misma espera circular simple, es decir. (12d)

$$m(C_i) > 1 \wedge m(r_{ns\_Cj}) = 0$$

Y sólo se utilizará en la ejecución de tareas que utilice los recursos dentro de la espera circular disponible.

3. Establecer prioridad entre piezas a A, B, C. Dado que el problema planteado por Peng, no plantea nada al respecto, se establece que la celda de manufactura dará prioridad a la ejecución de las tareas de las piezas tipo A, luego a las tareas de las piezas B, y por último a las C. Esta asignación de prioridad, permite resolver conflictos, cuando dos o más tareas de piezas distintas compiten por el mismo recurso. Esto se resuelve asignando el recurso a la tarea que demanda la pieza con la prioridad más alta. A continuación se muestra el seudocódigo, por ejemplo $J_2^A(r_1)$ significa la tarea 2 sobre la pieza A, requiere el recurso r_1. Por otra parte $m\left(\bullet\bullet\left(J_2^A(r_1)\right)\right) = 1$, significa que la cantidad de *tokens* de la tarea anterior a $J_2^A(r_1)$ debe ser uno. Además, en el seudocódigo se está incorporando el criterio del punto

anterior, ya que el recurso r_i es parte de la espera circular C_j, por lo que se debe contemplar la cota mínima de recursos disponibles.

$$
\begin{aligned}
&IF\ \left(m(C_j)>1\ \wedge\ \left(m\left(\bullet\bullet\left(J_l^A(r_i)\right)\right)=1\ \wedge\ m\left(\bullet\bullet\left(J_m^B(r_i)\right)\right)=1\ \wedge\ m\left(\bullet\bullet\left(J_n^C(r_i)\right)\right)=1\right)\right)\\
&\quad THEN\ u_{D\_kl}=1\quad \Rightarrow\ J_l^A(r_i)=ejecute\\
&ELSE\ IF\ \left(m(C_j)>1\ \wedge\ \left(m\left(\bullet\bullet\left(J_m^B(r_i)\right)\right)=1\ \wedge\ m\left(\bullet\bullet\left(J_n^C(r_i)\right)\right)=1\right)\right)\\
&\quad THEN\ u_{D\_km}=1\ \Rightarrow\ J_m^B(r_i)=ejecute\\
&ELSE\ IF\ \left(m(C_j)>1\ \wedge\left(m\left(\bullet\bullet\left(J_n^C(r_i)\right)\right)=1\right)\right)\\
&\quad THEN\ u_{D\_kn}=1\ \Rightarrow\ J_n^C(r_i)=ejecute\\
&ENDIF.
\end{aligned}
\tag{13}
$$

Donde:

$$r_i \in C_j$$

$$\left(\bullet\bullet\left(J_l^{\Pi}(r_i)\right)\right)\cap J\neq\emptyset,\ \ \Pi=\{A,B,C\}$$

$$m(C_j)=m(r_i)+m(C_j-r_i),\ m(r_i)=1$$

4. Si existen tareas $J_l^A(r_k)$, $J_m^A(r_k)$, sobre la misma pieza A, que requieren el mismo recurso $r_i \in R_s$, donde la tarea m es posterior a la tarea l, entonces se asignará el recurso a la tarea que cumpla (12a) o (12b).

$$
\begin{aligned}
&IF\ \left(m(C_j)>1\ \wedge\ m\left(\bullet\bullet\left(J_l^A(r_i)\right)\right)=1\ \wedge\ m\left(\bullet\bullet\left(J_m^A(r_i)\right)\right)=1\right)\\
&\quad THEN\ u_{D\_kl}=1\quad \Rightarrow\ J_l^A(r_i)=ejecute\\
&ELSE\ IF\ \left(m(r_i)=1\ \wedge\ m\left(\bullet\bullet\left(J_l^A(r_i)\right)\right)=1\ \wedge\ m\left(\bullet\bullet\left(J_m^A(r_i)\right)\right)=1\right)\\
&\quad THEN\ u_{D\_km}=1\quad \Rightarrow\ J_m^A(r_i)=ejecute\\
&ENDIF
\end{aligned}
\tag{14}
$$

7.1.4 Algoritmo de control.

El algoritmo que se propone, pretende controlar la red de petri de la figura 12, y busca que la red permanezca viva de acuerdo a Murata [**17**]. Para simplificar el análisis se eliminará de la RdP los recursos TB1 y TB2, ya que no contribuyen a los bloqueos del sistema, tal como se aprecia en la matriz de esperas **CWr**, ecuación (10). Columnas con ceros dentro de la matiz **CWr** indica que el recurso no están contenido en ninguna espera circular.

La red de la figura 13 es la nueva red reducida, y además incorpora los nodos lugar de control. Estos nodos se agregaron a la RdP, utilizando el primer lineamiento propuesto, e.c. (11). Si la tarea J_2^A debe ser controla, entonces el nodo de control se llamará u_{D_A2}. La figura 13, muestra con doble circulo todos los nodos de control u_D de la celda de manufactura

El siguiente algoritmo busca colocar marcas en los lugares de control, con la finalidad de evitar los bloqueos de la red, y buscando maximizar los productos procesados. A cada nodo de

control se asignará una regla siguiendo los lineamientos anteriores. A continuación el seudocódigo utilizando los 4 lineamientos propuestos:

$$
\begin{aligned}
&WHILE\ (true)\\
&\quad u_{D\_A2}=0, u_{D\_A3}=0, u_{D\_A4}=0\\
&\quad u_{D\_B2}=0, u_{D\_B3}=0, u_{D\_B4}=0\\
&\quad u_{D_{C2}}=0, u_{D_{C3}}=0, u_{D_{C4}}=0,\\
&\quad u_{D_{C2}}=0, u_{D_{C6}}=0, u_{D_{C7}}=0,\\
&\quad u_{D\_C81}=0, u_{D\_C82}=0, u_{D\_C9}=0,\\
&\quad IF\ \left(m(C_1)>1\ \wedge\ m\left(\bullet\bullet\left(J_2^A(RB)\right)\right)>0\right)\ THEN\ u_{D\_A2}=1\\
&\quad ELSE\ IF\ \left(m(CNC)=1\ \wedge\ m\left(\bullet\bullet\left(J_3^A(CNC)\right)\right)=1\right)THEN\ u_{D\_A3}=1\\
&\quad ELSE\ IF\ \left(m(RB)=1\ \wedge\ m\left(\bullet\bullet\left(J_4^A(RB)\right)\right)=1\right)THEN\ u_{D\_A4}=1\\
&\quad ELSE\ IF\ \left(m(C_2)>1\ \wedge\ m\left(\bullet\bullet\left(J_2^B(RB)\right)\right)>0\right)THEN\ u_{D\_B2}=1\\
&\quad ELSE\ IF\ \left(m(VMC)=1\ \wedge\ m\left(\bullet\bullet\left(J_3^B(VMC)\right)\right)=1\right)THEN\ u_{D\_B3}=1\\
&\quad ELSE\ IF\ \left(m(RB)=1\ \wedge\ m\left(\bullet\bullet\left(J_4^B(RB)\right)\right)=1\right)THEN\ u_{D\_B4}=1\\
&\quad ELSE\ IF\ \left(m(C_1)>1\ \wedge\ m\left(\bullet\bullet\left(J_2^C(RB)\right)\right)>0\right)THEN\ u_{D\_C2}=1\\
&\quad ELSE\ IF\ \left(m(CNC)=1\ \wedge\ m\left(\bullet\bullet\left(J_3^C(CNC)\right)\right)=1\right)THEN\ u_{D\_C3}=1\\
&\quad ELSE\ IF\ \left(m(C_4)>1\ \wedge m\,(VMC)=0\ \wedge\ m\left(\bullet\bullet\left(J_4^C(RB)\right)\right)=1\right)THEN\ u_{D\_C4}=1\\
&\quad ELSE\ IF\ \left(m(C_2)>1\ \wedge\ m\left(\bullet\bullet\left(J_6^C(RB)\right)\right)=1\right)THEN\ u_{D\_C6}=1\\
&\quad ELSE\ IF\ \left(m(C_2)>1\ \wedge\ m\left(\bullet\bullet\left(J_6^C(RB)\right)\right)=1\right)THEN\ u_{D\_C7}=1\\
&\quad ELSE\ IF\ \left(m(VMC)=1\ \wedge\ m\left(\bullet\bullet\left(J_8^C(VMC)\right)\right)=1\right)THEN\ u_{D\_C81}=1\\
&\quad ELSE\ IF\ \left(m(VMC)=1\ \wedge m\left(\bullet\bullet\left(J_8^C(VMC)\right)\right)=1\right)THEN\ u_{D\_C82}=1\\
&\quad ELSE\ IF\ \left(m(RB)=1\ \wedge m\left(\bullet\bullet\left(J_9^C(RB)\right)\right)=1\right)THEN\ u_{D\_C9}=1\\
&\quad ENDIF\\
&ENDWHILE
\end{aligned}
\tag{15}
$$

El código anterior (15), utiliza las esperas circulares simples para manejar los bloqueos de la red, estas se sustituyen por las máquinas que las componen. Por ejemplo el marcado de la espera circular C_1 , denotado por $m(C_1)$ se sustituye por la sumatoria de los marcados del robot y de la máquina CNC, $m(RB+CNC)$. Note que la RdP, se controla llevando control de las esperas circulares simples, las cuales son: $\{C_1, C_2, C_4\}$.

Dentro del algoritmo anterior aparecen los términos $m\left(\bullet\bullet\left(J_2^A(RB)\right)\right)>0$, $m\left(\bullet\bullet\left(J_2^B(RB)\right)\right)>0$ y $m\left(\bullet\bullet\left(J_2^C(RB)\right)\right)>0$, estos se refieren a la entrada de los productos a la calda, posee un signo de mayor a cero, porque modelan una cola de entrada a la red, por lo que el marcado puede ser desde uno a infinito, ya que la cola no es acotada.

El código que se muestra en (16), simplifica la lectura del código, y a partir de este se realiza una implementación casi directa al programa Petri.Net.

$$
\begin{aligned}
&WHILE\ (true)\\
&\quad u_{D\_A2}=0, u_{D\_A3}=0, u_{D\_A4}=0\\
&\quad u_{D\_B2}=0, u_{D\_B3}=0, u_{D\_B4}=0\\
&\quad u_{D_{C2}}=0, u_{D_{C3}}=0, u_{D_{C4}}=0,\\
&\quad u_{D_{C2}}=0, u_{D_{C6}}=0, u_{D_{C7}}=0,\\
&\quad u_{D\_C81}=0, u_{D\_C82}=0, u_{D\_C9}=0,\\
&\quad IF\ \ (\,m(RB+CNC)>1\ \wedge\ m(PI_A)>0\,)\ THEN\ u_{D\_A2}=1\\
&\quad ELSE\ IF\ \ \big(m(CNC)=1\ \wedge\ m\big(J_2^A(RB)\big)=1\big)THEN\ u_{D\_A3}=1\\
&\quad ELSE\ IF\ \ \big(m(RB)=1\ \wedge\ m\big(J_3^A(CNC)\big)=1\big)THEN\ u_{D\_A4}=1\\
&\quad ELSE\ IF\ \ (m(RB+VMC)>1\ \wedge\ m(PI_B)>0)THEN\ u_{D\_B2}=1\\
&\quad ELSE\ IF\ \ \big(m(VMC)=1\ \wedge\ m\big(J_2^B(RB)\big)=1\big)THEN\ u_{D\_B3}=1\\
&\quad ELSE\ IF\ \ \big(m(RB)=1\ \wedge\ m\big(J_3^B(CNC)\big)=1\big)THEN\ u_{D\_B4}=1\\
&\quad ELSE\ IF\ \ (m(RB+CNC)>1\ \wedge\ m(PI_C)>0)THEN\ u_{D\_C2}=1\\
&\quad ELSE\ IF\ \ \big(m(CNC)=1\ \wedge\ m\big(J_2^C(RB)\big)=1\big)THEN\ u_{D\_C3}=1\\
&\quad ELSE\ IF\ \ \big(m(RB+Bf)>1\ \wedge\ m(VMC)=0\ \wedge\ m\big(J_3^C(CNC)\big)=1\big)THEN\ u_{D\_C4}=1\\
&\quad ELSE\ IF\ \ \Big(m(RB+VMC)>1\ \wedge\ m\Big(J_5^C(Bf)\Big)=1\Big)\ THEN\ u_{D\_C6}=1\\
&\quad ELSE\ IF\ \ \big(m(RB+VMC)>1\ \wedge\ m\big(J_3^C(CNC)\big)=1\big)THEN\ u_{D\_C7}=1\\
&\quad ELSE\ IF\ \ \big(m(VMC)=1\ \wedge\ m\big(J_6^C(RB)\big)=1\big)THEN\ u_{D\_C81}=1\\
&\quad ELSE\ IF\ \ \big(m(VMC)=1\ \wedge\ m\big(J_7^C(RB)\big)=1\ \big)THEN\ u_{D\_C82}=1\\
&\quad ELSE\ IF\ \ \big(m(RB)=1\ \wedge\ m\big(J_8^C(VMC)\big)=1\ \big)THEN\ u_{D\_C9}=1\\
&\quad ENDIF\\
&\qquad ENDWHILE
\end{aligned}
\tag{16}
$$

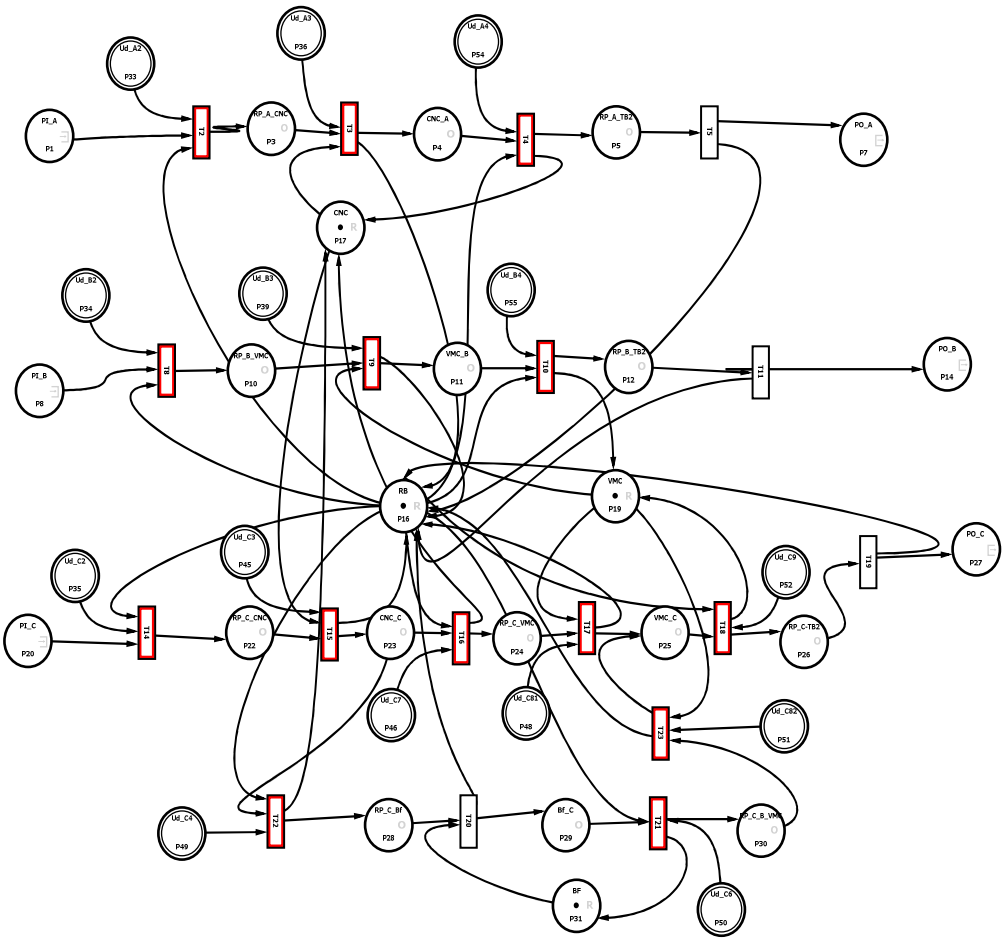


Figura 13. Red de Petri con los nodos lugar de control. Estos nodos de control se representan como círculos dobles y tienen un arco dirigido a un nodo transición. (Editor Petri.Net)

7.2 Elaboración del controlador para la celda de laboratorio.

7.2.1 Árbol de ensamble de la celda de manufactura.

El árbol de ensamble se compone por las siguientes operaciones realizadas por la celda. La tabla 3 se muestra la nomenclatura de la celda.

TABLA 3: OPERACIONES POR LA CELDA DE LABORATORIO

Notación	Tareas
Ra_M	El robot toma un acrílico de la banda y lo pone en la máquina CNC.
Ma	La máquina CNC desbasta el acrílico
Ra_B	El robot toma el acrílico recién maquinado y lo pone en el buffer
Rc_E	El robot toma una caja de la entrada y la coloca en la mesa de ensamble
Ep_1	Sub ensamble 1 terminado
Rr1_E	El robot ensambla la rueda 1

Ep_2	Sub ensamble 2 terminado
Rr1_E	El robot ensambla la rueda 2
Ep_3	Sub ensamble 3 terminado
Ra_E	El robot toma un acrílico del buffer, lo ensambla y se lleva el producto terminado a la banda
PO	Saca el producto terminado por la banda.

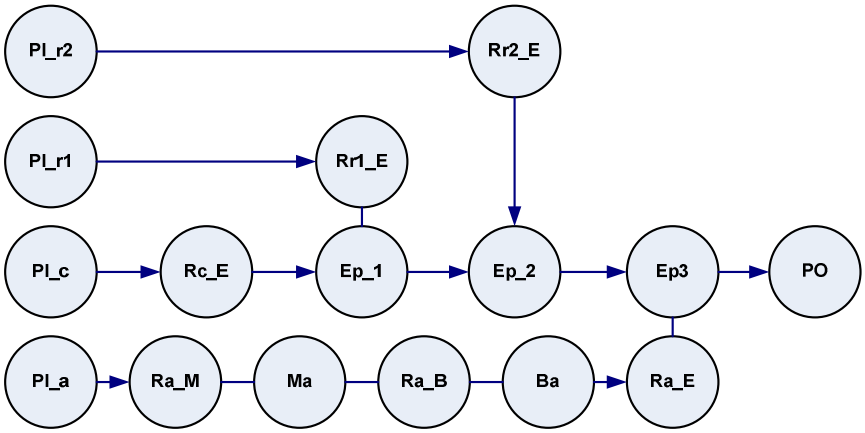


Figura 14. Árbol de ensamble de la celda de manufactura, para la fabricación de una caja reductora. (MS Visio 2007).

7.2.2 Red de Petri de la celda de manufactura.

Se realiza la red de petri de la celda de manufactura, a partir de un estudio del ensamble de las partes : caja, rueda 1. rueda 2, y tapa de acrílico. A continuación se muestra la red de Petri realizada en el software Petri.NET.

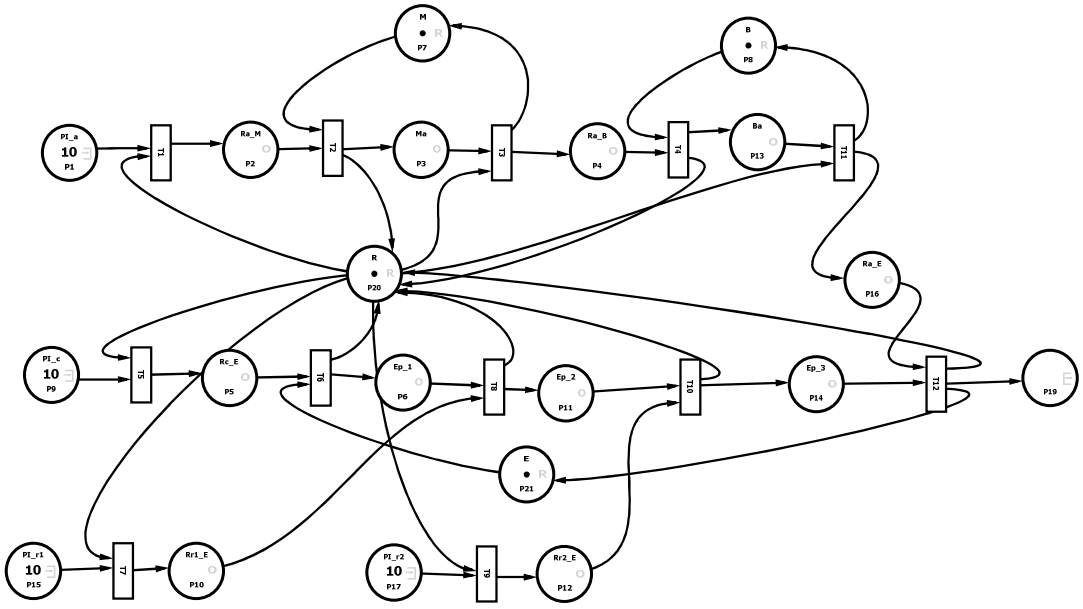


Figura 15. Red de Petri del laboratorio de manufactura sin lugares de control. (Editor Petri.Net)

7.2.3 Obtención de los lugares de conflicto del sistema.

Los lugares de conflicto del sistema son aquellas transiciones donde se utilizan los recursos compartidos, en este caso el robot. Todas las transiciones con conflicto deben llevar un lugar de control. Los nodos lugar de control, U_{d_k1}, se ubican en todos los lugres de control, que estén en conflicto, de acuerdo con la ecuación (11).

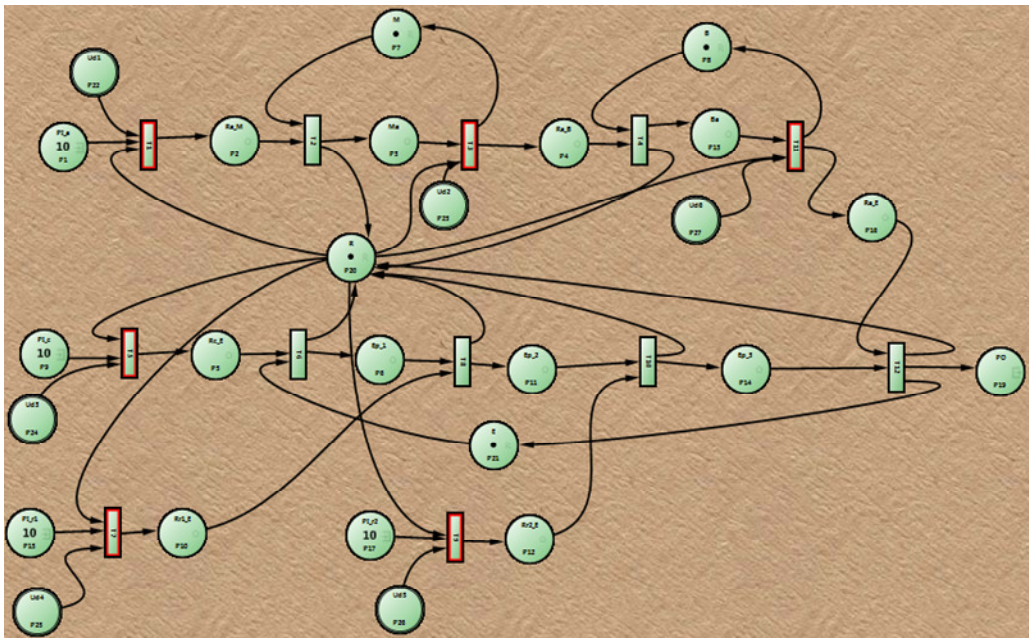


Figura 16. RdP con los lugares de control, en las transiciones donde en conflicto. (Editor Petri.Net).

7.2.4 Obtención de las esperas circulares y sifones críticos del sistema

Si se calculan las esperas circulares simples, se obstinen los siguientes dos conjuntos de recursos:

C1={M,R}

C2={B,R}

El cálculo de los sifones críticos para ambas esperas , se realiza de acuerdo a la ecuación (6):

Sc1 ={M, R, Ra_B, Rc_E, Rr1_E, Rr2_E, Ra_E}

Sc2={B, R, Ra_E, Ra_M, Rc_E, Rr1_E, Rr2_E}

Para evitar los bloqueos en el sistema es necesario que tanto Sc1 y Sc2, nunca estén vaciós, es decir sin *tokens* en los respectivos nodos lugar.

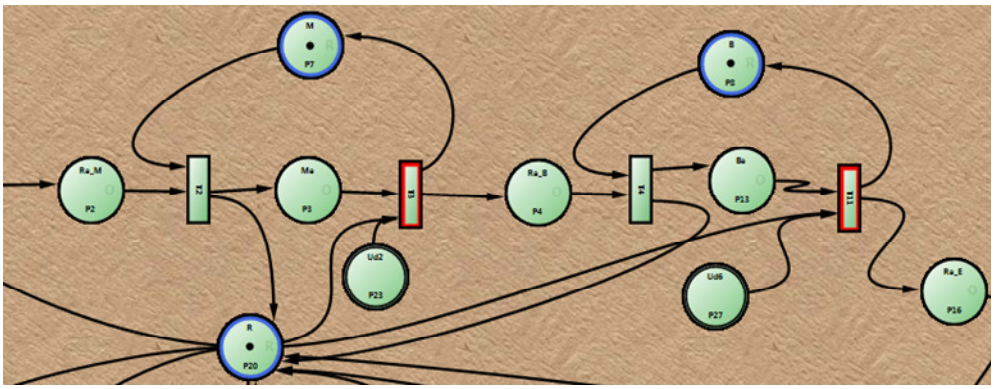


Figura 17. Los recursos R, B, M, que conforman las esperas circulares del proceso de ensamble.

8 SIMULACIONES DE LAS CELDAS DE MANUFACTURA

En este apartado se mostrarán las simulaciones realizadas de los controladores implementados tanto para la celda teórica, como para la celda de laboratorio.

8.1 Implementación del controlador para la celda teórica en Petri.Net

El algoritmo de control implementado corresponde a la algoritmo descritos en e.c (16), este seudocódigo se han modificado ligeramente debido a que el editor de reglas no soportar el ELSE IF, por otra parte, la estructura WHILE es parte del ciclo de simulación de programa. La figura 18, muestra la pantalla donde se implemento el controlador.

Rules Editor

Expression
<Double-click here to add a new rule>
IF (1==1) THEN (Ud_A2=0 AND Ud_A3=0 AND Ud_A4=0 AND Ud_B2=0 AND Ud_B3=0 AND Ud_B4=0 AND Ud_C2=0 AND Ud_C3=0
IF (RB+CNC)>1 AND PI_A>0 THEN (Ud_A2=1)
IF CNC==1 AND RP_A_CNC==1 THEN Ud_A3=1
IF RB==1 AND CNC_A==1 THEN Ud_A4=1
IF((RB+VMC)>1) AND PI_B>0 AND Ud_A2==0 AND Ud_A4==0 THEN Ud_B2=1
IF VMC==1 AND RP_B_VMC==1 AND Ud_C81==0 THEN Ud_B3=1
If RB==1 AND VMC_B ==1 AND Ud_A2==0 AND Ud_A4==0 AND Ud_B2==0 THEN Ud_B4=1
IF ((RB+CNC)>1) AND PI_C>0 AND (Ud_A2+Ud_A4+ Ud_B2+ Ud_B4)==0)THEN Ud_C2=1
If CNC==1 AND RP_C_CNC==1 AND Ud_A3==0 THEN Ud_C3=1
If (BF+RB)>1 AND CNC_C==1 AND VMC==0 AND ((Ud_A2+Ud_A4+ Ud_B2+ Ud_B4+Ud_C2)==0) THEN Ud_C4=1
If (RB+VMC)>1 AND Bf_C==1 AND ((Ud_A2+Ud_A4+ Ud_B2+ Ud_B4+Ud_C2+Ud_C7)==0) THEN Ud_C6=1
If (((RB+VMC)>1) AND CNC_C==1 AND Bf_C==0 AND((Ud_A2+Ud_A4+ Ud_B2+ Ud_B4+Ud_C2)==0)THEN Ud_C7=1
IF VMC==1 AND RP_C_VMC==1 AND Ud_B3==0 THEN Ud_C81=1
If VMC==1 AND RP_C_B_VMC==1 AND Ud_C81==0 THEN Ud_C82=1
IF RB==1 AND VMC_C== 1 AND((Ud_A2+Ud_A4+ Ud_B2+ Ud_B4+Ud_C2+Ud_C4+Ud_C6+Ud_C7)==0) THEN Ud_C9=1

Figura 18. Editor de reglas del programa Petri.Net, donde se implementó las reglas del control descritas en e.c (16).

8.2 Simulación de la celda teórica

Una vez implementada la RdP de la celda de manufactura en el software de simulación, se procedió a ejecutar mil pasos de simulación. La llegada de las piezas tipo A, B, y C se seleccionó en forma aleatoria, para buscar una mayor diversidad de secuencias de piezas de entrada a la celda.

8.2.1 Simulaciones realizadas de la celda sin controlador.

En la ejecución realizada de 1000 pasos de simulación para la RdP, se muestra que la celda se bloquea en los primeros pasos. Se observa que existen numerosos conflictos por la solicitud de los recursos. Por ejemplo en la figura 19, para el Robot (fila RB), los pasos 5, 7, 25, 27, 29, 31, 40, 42, 48, indican valores negativos, esto indica que se solicita el recurso más veces que lo disponible. El simulador para resolver el conflicto atiende todas las tareas, pero indica en el oscilograma la cantidad de recursos en deuda para atender las tareas adecuadamente.

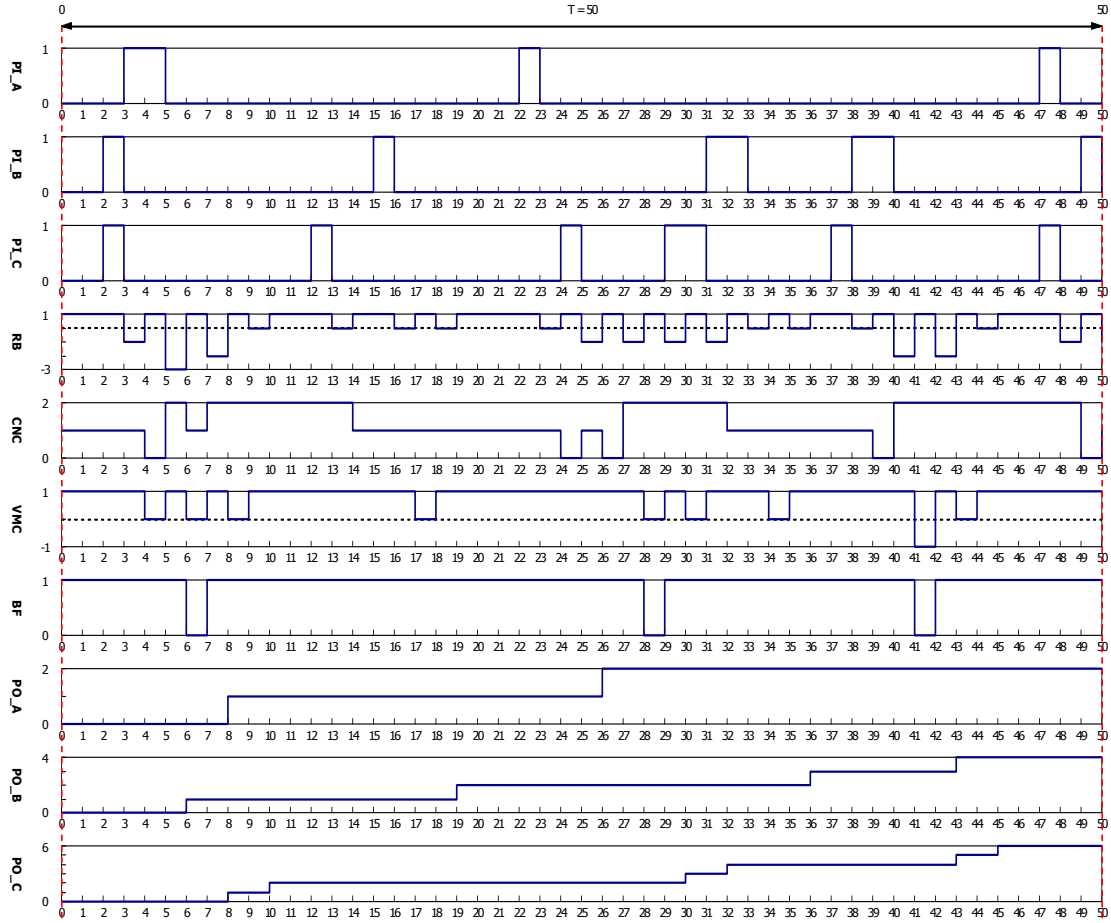


Figura 19. Oscilograma de las colas de entrada y salida de las piezas A, B, C, Así como, la utilización de las máquinas y buffer de la celda. Para las máquinas el valor de cero o negativo indica maquinaria en uso.

La figura 20, muestra el resultado numérico del oscilograma. En esta tabla se indica mediante la letra C, todas las simulaciones que están en conflicto, ya que la cantidad de recursos es deficiente. Por otra parte, se observa que para el recurso CNC, tanto en la tabla de resultados como en el oscilograma, el marcado llega a ser dos, esto se debe a que el simulador introduce marcas a la RdP con la idea de resolver el conflicto, pero estas marcas al final incrementan el marcado final.

No	PI_A	PI_B	PI_C	RB	CNC	VMC	BF	PO_A	PO_B	PO_C
20	0	0	0	1	1	1	1	1	2	2
21	0	0	0	1	1	1	1	1	2	2
22	1	0	0	1	1	1	1	1	2	2
23	0	0	0	0	1	1	1	1	2	2
24	0	0	1	1	0	1	1	1	2	2
25	0	0	0	-1	1	1	1	1	2	2
26	0	0	0	1	0	1	1	2	2	2
27	0	0	0	-1	2	1	1	2	2	2
28	0	0	0	1	2	0	0	2	2	2
29	0	0	1	-1	2	1	1	2	2	2
30	0	0	1	1	2	0	1	2	2	3
31	0	1	0	-1	2	1	1	2	2	3
32	0	1	0	1	1	1	1	2	2	4
33	0	0	0	0	1	1	1	2	2	4
34	0	0	0	1	1	0	1	2	2	4
35	0	0	0	0	1	1	1	2	2	4
36	0	0	0	1	1	1	1	2	3	4
37	0	0	1	1	1	1	1	2	3	4
38	0	1	0	0	1	1	1	2	3	4
39	0	1	0	1	0	1	1	2	3	4
40	0	0	0	-2	2	1	1	2	3	4
41	0	0	0	1	2	-1	0	2	3	4
42	0	0	0	-2	2	1	1	2	3	4
43	0	0	0	1	2	0	1	2	4	5
44	0	0	0	0	2	1	1	2	4	5
45	0	0	0	1	2	1	1	2	4	6
46	0	0	0	1	2	1	1	2	4	6
47	1	0	1	1	2	1	1	2	4	6
48	0	0	0	-1	2	1	1	2	4	6
49	0	1	0	1	0	1	1	2	4	6
50	0	0	0	-1	1	1	1	2	4	6

Figura 20. Resultado numérico parcial de la simulación de la RdP sin controlar. El software indica los pasos específicos en que el sistema se bloquea,la fila se marca la fila con la letra L. Si existe conflicto de recursos, la fila se marca la letra C.

Finalmente, en la simulación se indica con la letra L el paso de simulación donde existe bloqueo del sistema, es decir es un marcado del cual no se puede evolucionar. La figura 20, muestra resultados parciales de los primeros 50 pasos de simulación realizados. La idea del algoritmo de control, es evitar tanto los conflictos como los bloqueos de la celda de manufactura.

8.2.2 Simulaciones realizadas de la celda con el algoritmo de control.

La RdP implementada en el simulador corresponde a la figura 13. La simulación de la RdP con la política de asignación de recursos establecida en la e.c (16), arroga el oscilograma de la figura 21. En dicho diagrama de tiempo todos los recursos están acotados entre cero y uno, no aparecen marcados de recursos con valores negativos o mayor a uno, por lo que el sistema ha sido controlado en forma exitosa. Se observa por ejemplo que el Robot (RB) siempre está operando, no existen colas en la entrada salvo las piezas C que posee la prioridad menor.

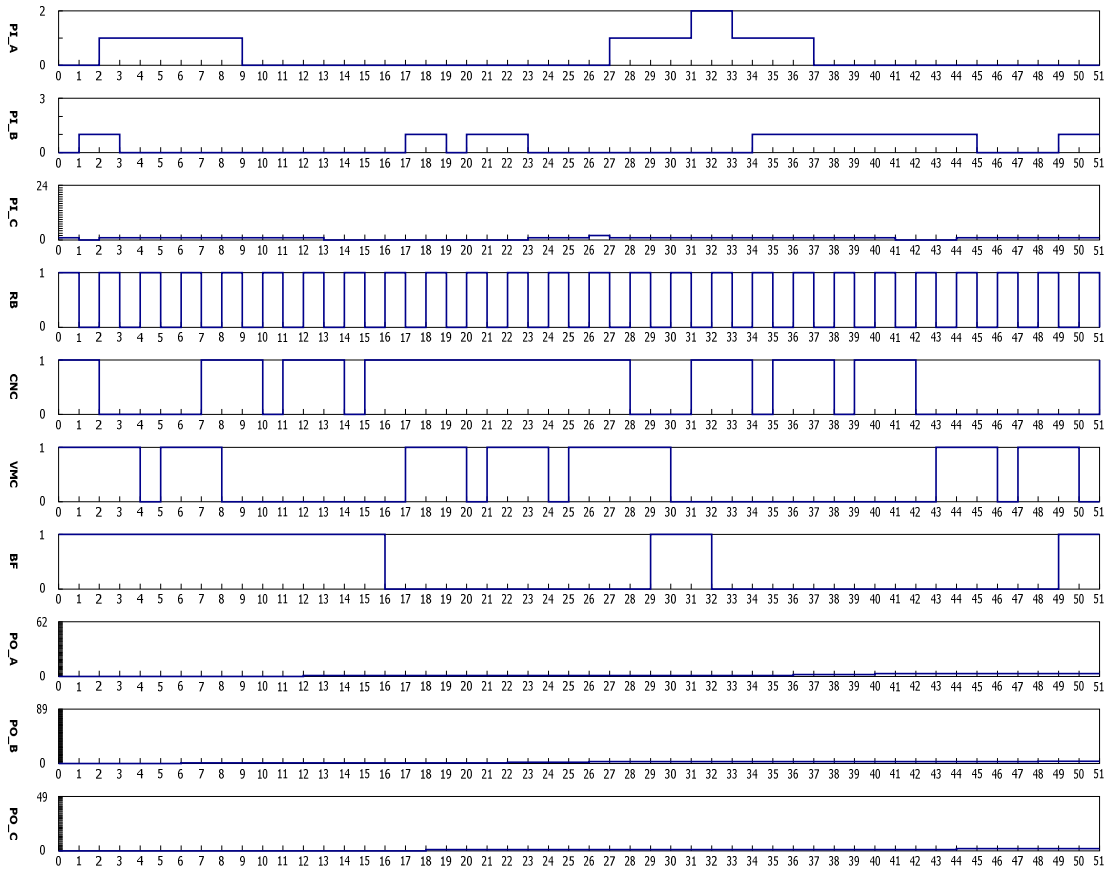


Figura 21. Oscilograma para la RdP controlada con el algoritmo propuesto. Se observa que los conflictos y bloqueos se han eliminado.

En 1000 pasos de simulación el sistema no presento ningún conflicto, tampoco se presentó situaciones de bloqueo. En la figura 22 se muestra los resultados parciales de las últimas simulaciones realizadas.

No	PI_A	PI_B	PI_C	RB	CNC	VMC	BF	PO_A	PO_B	PO_C
968	1	1	22	1	0	1	0	59	86	48
969	1	0	22	0	0	1	0	59	86	48
970	1	0	22	1	0	0	0	59	86	48
971	1	0	23	0	0	1	0	59	86	48
972	1	1	23	1	0	1	0	59	87	48
973	1	0	23	0	0	1	0	59	87	48
974	1	0	23	1	0	0	0	59	87	48
975	1	0	23	0	0	1	0	59	87	48
976	1	1	23	1	0	1	0	59	88	48
977	2	0	23	0	0	1	0	59	88	48
978	2	0	23	1	0	0	0	59	88	48
979	2	0	23	0	0	1	0	59	88	48
980	2	0	23	1	0	1	0	59	89	48
981	2	0	23	0	0	1	1	59	89	48
982	2	0	23	1	0	0	1	59	89	48
983	2	0	23	0	1	0	1	59	89	48
984	2	0	23	1	1	0	0	59	89	48
985	1	0	23	0	1	0	0	59	89	48
986	1	0	23	1	0	0	0	59	89	48
987	1	0	24	0	1	0	0	59	89	48
988	1	0	24	1	1	0	0	60	89	48
989	0	1	24	0	1	0	0	60	89	48
990	0	1	24	1	0	0	0	60	89	48
991	1	1	24	0	1	0	0	60	89	48
992	1	1	24	1	1	0	0	61	89	48
993	0	1	24	0	1	0	0	61	89	48
994	0	1	24	1	0	0	0	61	89	48
995	0	1	24	0	1	0	0	61	89	48
996	0	1	24	1	1	0	0	62	89	48
997	0	1	23	0	1	0	0	62	89	48
998	0	1	23	1	0	0	0	62	89	48
999	0	1	23	0	0	1	0	62	89	48
1000	0	1	23	1	0	1	0	62	89	49

Figura 22. Resultado parciales de las últimas 32 simulaciones. Se observó que en mil pasos de simulación el sistema no presento conflictos ni bloqueos.

8.3 Pruebas del controlador para la celda de laboratorio.

A continuación se muestran la simulaciones de los algoritmos de despacho, LBFS y FBFS, para la celda de manufactura de laboratorio.

8.3.1 Simulación de la celda de laboratorio sin reglas de control

Para corroborar que el sistema requiera de una política de despacho se simuló la red de Petri del sistema omitiendo los lugares de control. Se obtuvo que el sistema se bloquea desde la tercera corrida si se le proveen entradas constantes, de manera que será necesario agregarle el controlador. En la figura 23 se puede apreciar el orden de operaciones.

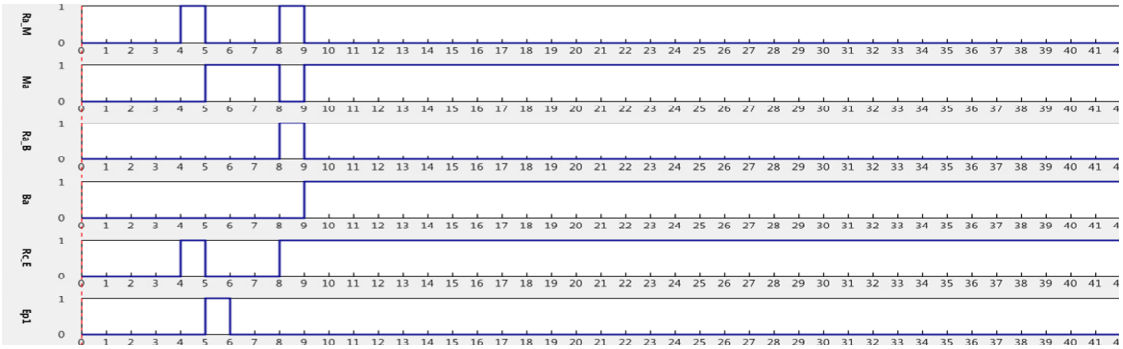


Figura 23 Tareas bloqueadas a partir de la novena iteración con entradas periódicas cada 3 ciclos.

8.3.2 Diseño y simulación del algoritmo de control LBFS

Una vez que se tiene la red de Petri se puede pasar a diseñar la ley de control que cumpla con la política LBFS. En este caso se tiene que hacer una pequeña modificación a la regla, ya que por lo general se aplica a procesos multi-reentrantes tipo 1, siendo nuestro proceso uno de tipo especial. La modificación consiste en que los lugares que controlan el ensamble de las ruedas dentadas y de la caja siguen reglas un poco diferentes, que dependen de la disponibilidad de recursos, y toman precedencia sobre todas las demás operaciones ya que son requisito para hacer la última operación de la espera circular:

```
If (1==1) THEN ((Ud1=0 AND Ud2=0 AND Ud3=0 AND Ud4=0 AND Ud5=0 AND Ud6=0))
If(PI_c>0 AND E==1)  THEN Ud3=1
Else If (PI_r1>0 AND Ep_1==1) THEN Ud4=1
Else If (PI_r2>0 AND Ep_2==1) THEN Ud5=1
Else If ( Ba==1 AND Ep_3==1) THEN Ud6=1
Else If (Ma==1 AND Ba==1 AND Ep_3==1) THEN Ud6=1
Else If(PI_a>0 AND Ma==1 AND Ba==1 AND Ep_3==1) THEN Ud6=1
Else If( Ma==1 AND Ba==0 AND  Ep_3==1) THEN Ud2=1
Else If (PI_a>0 AND  Ma==0 AND Ba==0 AND Ep_3==1) THEN (Ud1=1)
```

En la simulación de la red de petri con esta política se logran evitar los bloqueos, pero el funcionamiento del sistema se vuelve secuencial.

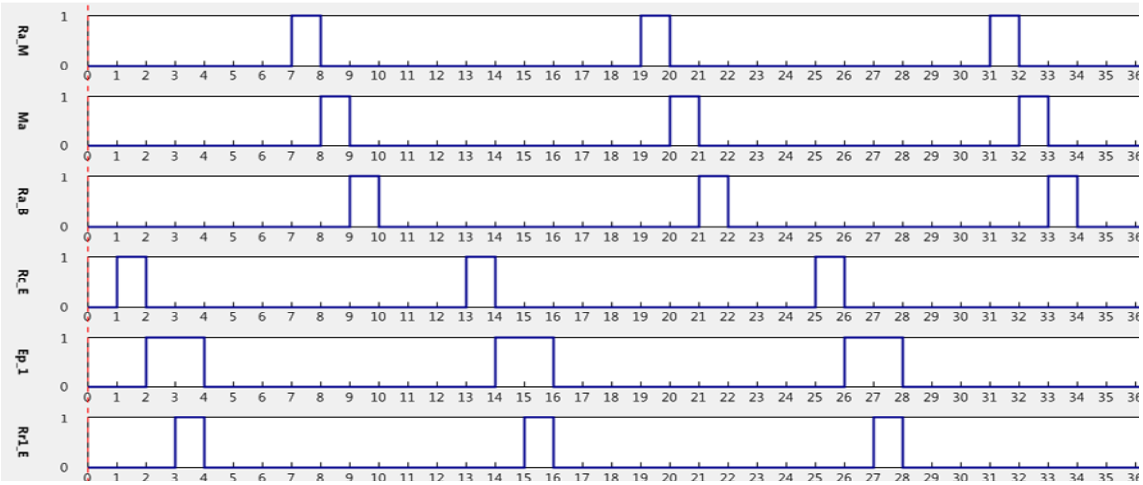


Figura 24: Comportamiento de algunas de las tareas con la política de LBFS

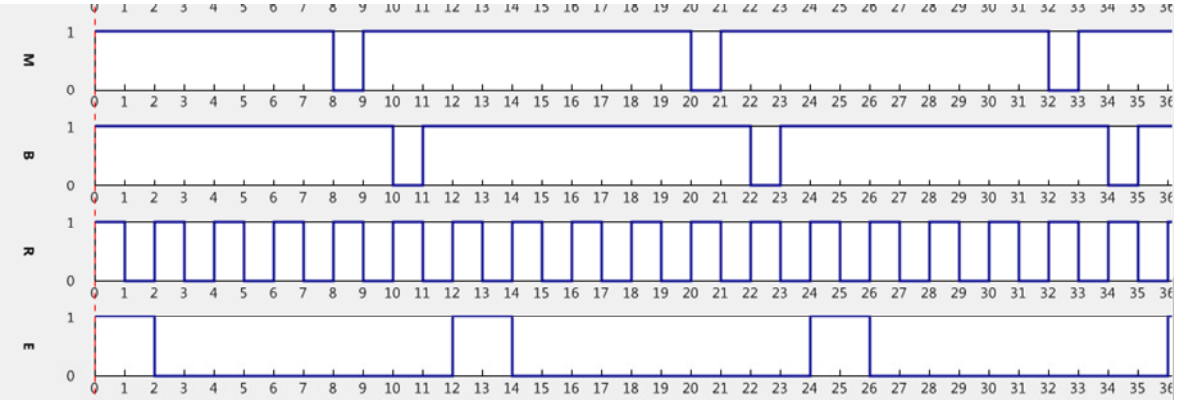


Figura 25. Comportamiento de los recursos con la política de LBFS

8.3.3 Diseño del algoritmo de control FBFS

Finalmente, se implementó la política FBFS. Como en este caso los recursos en espera circular son solo dos no se requiere de un contador para implementar esta política, por lo que solo se deben de cuidar las operaciones activas. El algoritmo es el siguiente:

```
If (1==1) THEN ((Ud1=0 AND Ud2=0 AND Ud3=0 AND Ud4=0 AND Ud5=0 AND Ud6=0))
Else If ( Ba==1 AND Ep_3==1) THEN Ud6=1
Else If (Ma==1 AND Ba==1 AND Ep_3==1) THEN Ud6=1
Else If(PI_a>=0 AND Ma==1 AND Ba==0) THEN Ud2=1
Else If (PI_a>0 AND Ma==0) THEN (Ud1=1)
Else If(PI_c>0 AND E==1 AND Ud1==0 AND Ud2==0)  THEN Ud3=1
Else If (PI_r1>0 AND Ep_1==1) THEN Ud4=1
Else If (PI_r2>0 AND Ep_2==1) THEN Ud5=1
```

En este caso la simulación entrega que las operaciones se realizan con menos tiempos muertos, por lo que el proceso se llega a optimizar.

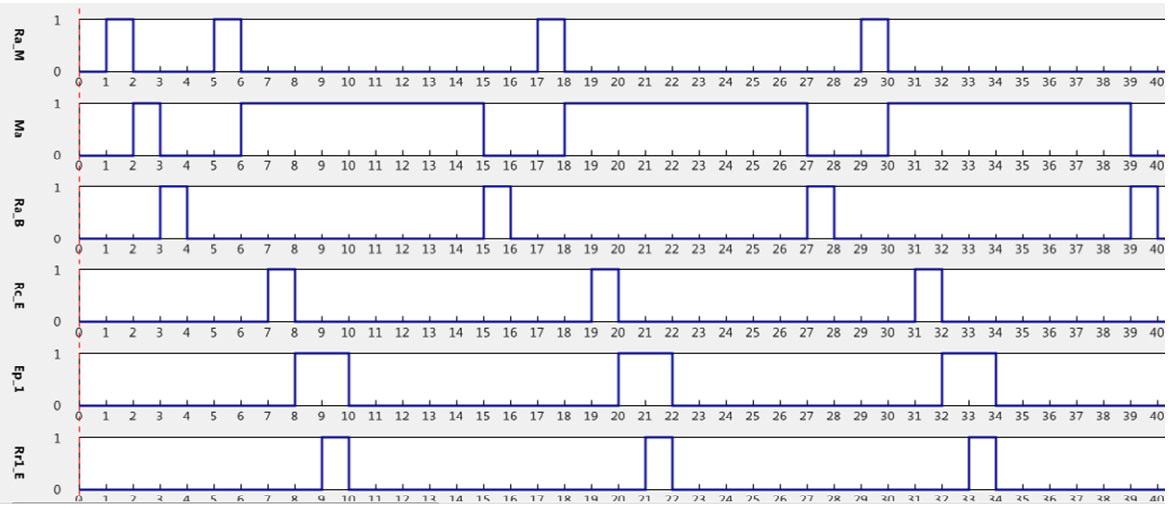


Figura 26. Comportamiento de algunas tareas con la política de FBFS

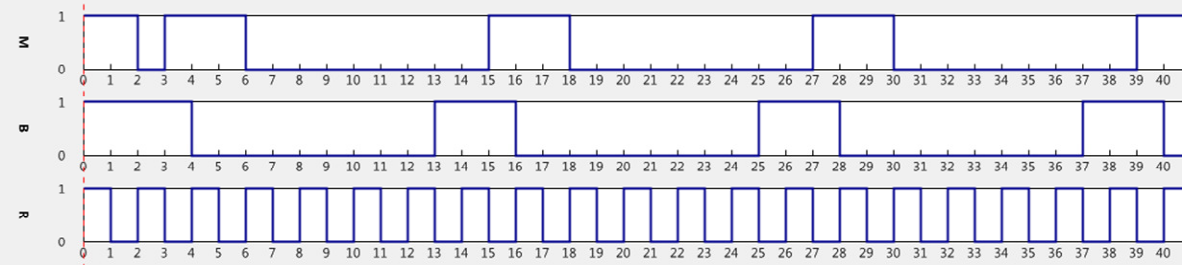


Figura 27. Comportamiento de los recursos con la política de FBFS

9 IMPLEMENTACIÓN DEL MODELO MATRICIAL EN EL PLC

En esta sección se detalla la implementación del controlador de la celda de manufactura ubicada en el laboratorio de sistemas modernos de manufactura de la Escuela de Producción Industrial. A continuación se muestra la estructura de control del algoritmo programado en el PLC, así como el flujo de datos, además de las características de los programas en todas las máquinas involucradas.

El modelo matricial se programó usando el lenguaje escalera del PLC RSLogix 5330. De acuerdo a la figura 4, para poder obtener los estados del sistema se requiere de tres entradas del proceso: El vector de tareas completadas Vc, el vector de productos a la entrada u y el vector de recursos disponibles Rc. Además, se requiere que la ley de control se ejecute para otorgarle un token a uno de los lugares de control. Cuando se tienen estas entradas se puede pasar a calcular el estado del sistema. Finalmente, cuando ya se tiene el estado actual se mandan salidas al sistema en forma del vector de comando de inicio de tareas Vs y el vector de comando de liberación de recursos Rs.

De esta forma, se siguen los siguientes pasos para la programación del modelo matricial en ecuaciones lógicas:

1. Obtener el vector de tareas completadas de acuerdo con sensores que hay en la planta.
2. Obtener el vector de recursos disponibles, que también proviene de sensores de la planta.
3. Obtener el vector de entrada de piezas nuevas.
4. Calcular la ley de control de acuerdo con las entradas, recursos y tareas completadas actuales.
5. Verificar el estado del sistema, para ver si se dispara alguna transición.
6. Mandar las instrucciones de liberación de recursos de acuerdo con las transiciones disparadas.
7. Mandar las instrucciones de inicio de tareas de acuerdo con las transiciones disparadas.

9.1 Estructura del control del proceso y de la comunicación entre las máquinas

El PLC es el controlador maestro del proceso, toma las decisiones sobre que tareas realizar y lleva el control de recursos. También lleva programada la dinámica del sistema en forma del vector de estados.

El robot tiene programadas todas las secuencias que debe realizar para ensamblar el producto, donde se selecciona que secuencia se ejecutará en base a un código enviado por el PLC en 5 entradas digitales. Además, el controlador del robot ya esta interfazado con la CNC, por lo que se

controlará desde el mismo. Finalmente, las demás salidas y entradas del PLC van hacia la banda. En la figura siguiente se puede apreciar las conexiones entre equipos del sistema.

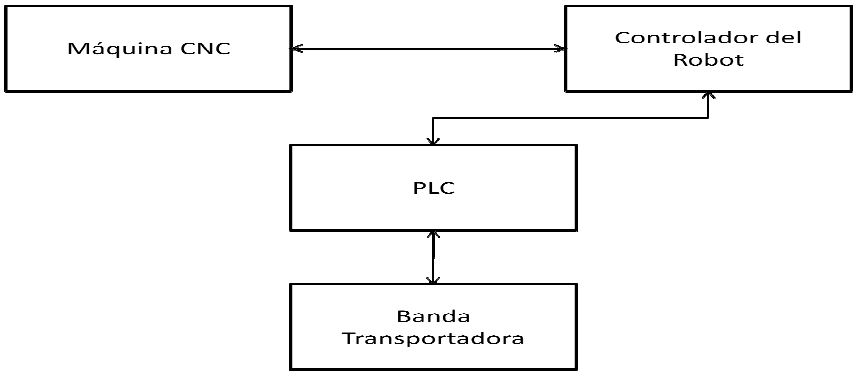


Figura 28. Conexiones entre las máquinas de la celda.

9.2 Lista de conexiones entre las máquinas

Para llevar un mejor control del cableado que existe entre los dispositivos se creó una lista de las conexiones existentes, donde se agrega el nombre que se les dio en el programa del PLC.

TABLA 4: LISTA DE SALIDAS DEL ROBOT Y SUS CONEXIONES

Puerto de salida	Destino	Nombre en el programa del PLC
Out 1	Local:2:I.Data.4	Fin del maquinado
Out2	Local:2:I.Data.7	---------
Out 3	Prensa de la CNC	---------
Out 4	Pistón en la entrada de cajas	---------
Out 5	Prensa de la mesa de ensamble	---------
Out 6	Entrada auxiliar 1 de la CNC	---------
Out 7	NC	---------
Out 8	NC	---------
Out 9	NC	---------
Out 10	Local:3:I.Data.9	Fin. Rut. Rob.
Out 11	Local:3:I.Data.11	Robot en home
Out 12	Local:3:I.Data.13	CNC disponible
Out 13	Local:3:I.Data.10	Buffer disponible*
Out 14	Local:3:I.Data.12	Ensamble disponible*
Out 15	NC	---------
Out 16	NC	---------

* En la implementación final no se utilizaron estas conexiones.

TABLA 5: LISTA DE ENTRADAS DEL ROBOT Y SUS CONEXIONES

Puerto de salida	Destino	Nombre en el programa del PLC
In 1	---------	---------
In 2	---------	---------
In 3	Local:5:O.Data.0	R0
In 4	Local:5:O.Data.2	R1
In 5	Local:5:O.Data.4	R2
In 6	Local:5:O.Data.6	R3
In 7	Local:5:O.Data.8	Iniciar maquinado
In 16	Aux 1 CNC	---------

9.3 Programa del Controlador Lógico Programable

El programa completo del PLC se encuentra en el apéndice 1. Cada módulo de la ecuación (1) es un módulo en el programa del PLC. Adicionalmente, se tuvo que agregar el programa una lógica de Set-Reset debido a la baja velocidad del robot en relación con la velocidad del PLC, esto con la finalidad de asegurar que las entradas sean leídas y el sistema no pierda datos.

9.4 Programa del controlador del brazo robótico

En el apéndice 2, se incluye el programa completo del brazo robótico. La estructura consiste en un ciclo donde se revisan los 4 bits que provienen del PLC (R3, R2, R1 y R0, de más a menos significativo), después se ejecuta la rutina que corresponde al código del PLC. La rutina de maquinado tiene un pin independiente (In 7) para poder ejecutarse de manera paralela al resto de las rutinas. A continuación se muestran las rutinas con los códigos correspondientes.

TABLA 6: RUTINAS DEL ROBOT Y SUS CÓDIGOS DEL PLC

Valor de las entradas del PLC	Rutina
0	N/A
1	Ra_M
2	Ir a home
3	Ra_B
5	Rc_E
6	Rr1_E
7	Rr2_E
8	Ra_E
9	Ms (Liberar máquina)

10 RESULTADOS Y DISCUSIÓN

10.1 Celda de manufactura teórica

En la sección 7.1.3 se proponen cuatro lineamientos para construir el algoritmo de control de celdas de manufactura flexible hibridas, combinación de MRF y FMRF. Con las reglas propuestas se propone para el problema planteado por Peng [**1**], un algoritmo de control en la sección 7.1.4. Esté algoritmo permite construir controlar la celda de manufactura evitando los conflictos y bloqueos del sistema.

Sin embargo este algoritmo no es único, existen otros cinco posibles algoritmos de control de acuerdo a la prioridad asignada a las piezas, de acuerdo a la regla dos. Por ejemplo si se prioriza el procesamiento de las piezas de la formas ACB, o BAC, o BCA, o CAB, o CBA, se obtendrán algoritmos de control distintos, donde cada sentencia de control cambia su posición dentro de la estructura ELSE IF propuesta en la e.c (14). La escogencia de cuál secuencia se selecciona está asociada usualmente a la:

- Tasa de llegada λi_{Π} o tasa de salida λo_{Π} , de cada una de las piezas respectivamente.
- Duración de las tareas para el procesamiento de las piezas, $J^{\Pi}=\{ J^{A}, J^{B}, J^{C}\}$.

Con esta información es posible seleccionar la prioridad de piezas más conveniente para el rendimiento de la celda de manufactura.

Si bien el algoritmo propuesto resuelve situaciones de conflicto y bloqueo, no fue posible evaluar su rendimiento, ya que el simulador utilizado no soporta las RdP temporizadas de libre escogencia, por tanto el algoritmo de control fue probado a partir de eventos simulados.

Como posibles trabajos futuros esta demostrar la generalidad de las reglas propuestas para otros tipos de celdas de manufactura. Además es necesario el desarrollo de un criterio que permita asignar la mejor prioridad de piezas utilizando λ_{Π} y J^{Π}. Por otra parte, es necesario probar la validez de los lineamientos propuestos involucrando la variable tiempo, Finalmente, es necesario realizar una implementación real de una celda, utilizando la Aproximación basada en Matrices y la Generación del algoritmo de control, con la finalidad de demostrar la teoría.

10.2 Celda de manufactura de laboratorio.

Si bien las bases del control por aproximaciones matriciales están bien fundamentadas en la teoría, algunas consideraciones se tomaron para poder aplicarlas sin tener errores. El primer problema con que se topó fue que si bien en la teoría se puede igualar cada transición, inicio y fin de tarea con un solo sensor de manera directa, en la práctica es muy fácil que se pierdan las señales, por lo que es necesario retenerlas cuando se leen. Por ello se cambiaron todas las asignaciones a variables por lógica de Set-Reset, donde los sensores asociados con una marca interna se convierten en la condición para ponerla en Set, y la próxima transición o tarea que se dispara después del evento se convierte en su Reset. De esta manera se puede garantizar que todas las señales sean leídas.

Al agregar la lógica de Set-Reset también se agrega otro problema, ya que el orden en que se escriben los escalones en el programa se vuelve primordial para el comportamiento del mismo por la manera secuencial en que se ejecuta. En este caso hay dos recomendaciones diferentes:

- Si una marca interna no tiene salidas a actuadores o al robot se debe colocar su Reset inmediatamente después del Set, para forzarla a estar prendida por lo menos un ciclo del PLC completo y asegurar que se lea en todos los puntos en los que puede impactar.

- Si una marca interna tiene salidas al exterior, se debe colocar su Reset inmediatamente después del escalón donde se evalúa la condición para ponerla en 0. De esta forma se evita que llegue activa a la sección de salidas en un momento en el que ya debería estar apagada.

Además, por la forma en que se realiza la interface del robot con el PLC,así como por su velocidad de ejecución, es necesario reiniciar las salidas en todos los finales de operación para que no se repitan tareas accidentalmente en caso de que la dinámica de sistema haga que no se active una nueva transición.

Un cambio importante que se realizó entre el modelo obtenido y el implementado es el momento en donde se consumen los recursos diferentes del robot. Por ejemplo, si bien en el diagrama de Petri menciona que M se consume al iniciar Ma, en la implementación se debe marcar como ocupado desde que inicia Ra_M (cuando ya se sabe que la máquina será ocupada). Esto se realiza por que debido a la diferencia de velocidades entre el Robot y el PLC se toma la nueva decisión antes de que el robot llegue a indicar que la máquina está ocupada.

Las operaciones de ensamble Ep1, Ep2 y Ep3 no tienen una función real dentro de la implementación, son solo operaciones ficticias que se utilizan para marcar que los sub-ensambles están listos. Por lo tanto, podrían ser omitidas con solo modificaciones menores al programa, pero se dejaron para que el programa sea congruente con el modelo obtenido de la celda.

En cuanto a la implementación de las políticas de control se encontraron dos problemas en la implementación que no habían sido contemplados en la teoría:

Primero, en el primer algoritmo se tenían casos donde dos posibles activaciones del mismo lugar de control eran posibles. Si bien en la teoría no presentaban ningún problema, por el barrido del PLC esto puede llegar a causar que se activen dos lugares de control al mismo tiempo. Es decir, el orden en el que se escriben las posibles activaciones de los lugares de control tiene un efecto directo sobre el comportamiento del sistema. Esto se resolvió agrupando todas las declaraciones sobre un mismo lugar de control.

Segundo, debido a la velocidad de ejecución del robot había un problema donde se activaban dos tareas al mismo tiempo y una de las dos era ignorada. La razón por la que sucedía era por que si bien en teoría la estructura de IF-THEN impide que se active más de un lugar de control a la vez, en la implementación dos lugares se podían llegar a activar antes de que el robot pudiera marcarse como ocupado.

Para ilustrar el caso anterior se puede pensar en el siguiente ejemplo: Hay todas las condiciones necesarias para activar X1 (anterior a Ra_M) y X5 (anterior a Rc_E), y corresponde al algoritmo de control despachar solamente una de las dos. Se tiene una política de FBFS, pot lo que se activa Ud1 para poder activar X1, y Ud3 no se activa; en ese ciclo se llega al vector de estados y se activa X1 y después Ra_Mi. En este momento, en teoría se debería de marcar el robot como un recurso ocupado y no se ejecutaría ninguna otra tarea que lo requiera, pero como el robot ni siquiera ha leido la instrucción que recibió del PLC no se marca como ocupado.

Como resultado a lo anterior en el próximo ciclo del programa llega al vector de control y Ud1 ya no se activa por que acaba de gastar las entradas que tenía. Por lo tanto, el único punto que detenía a Ud3 (Ud1 activo) ahora lo deja activarse, y como resultado se dispara X5 y después Rc_Ei al mismo tiempo en que Ra_Mi se encuentra activo. Lo anterior causa un conflicto en las salidas, donde el robot solo obedece a uno de los dos comandos y cuando lo termina el PLC toma los dos comandos como terminados y se pierde una operación.

Para resolver este problema a las políticas de control se agrega que no puedan activarse los lugares de control si cualquier otra transición o tarea que involucre uso del robot se encuentra activa. Esta condición no tiene impacto sobre la teoría sobre la que se trabajó, pero soluciona el problema causado por la diferencia de velocidades entre e PLC y el robot.

10.2.1 Comportamiento de la celda de manufactura de laboratorio

Los siguientes resultados son producto de la implementación en la celda de manufactura, de las siguientes políticas de control de tareas: LBFS y FBFS.

Se programó el robot, para que siempre se siguen los mismos movimientos, independientemente de si las piezas a la entrada llegan o en que orden lleguen. En el caso del controlador con política LBFS, el sistema también se comporta como una rutina rígida, pero ya no realiza las operaciones si no recibe las partes a la entrada, por lo que reduce el riesgo de partes defectuosas. Finalmente, cuando se corre el controlador con política FBFS el orden de tareas se vuelve mucho más flexible, y varía según la combinación de entradas, tareas completadas y recursos disponibles que tenga la celda. Es importante misionar que ninguna políticas programada generó bloqueos de la celda. Esto ratifica la teoría que sistemas MRF son fácilmente controlables con estas dos políticas de control.

En cuanto a resultados medibles, existe un pequeño aumento del tiempo de fabricación al utilizar el controlador basado en matrices con política LBFS, esto por que la velocidad del robot no es buena y las operaciones de regresar a home del robot consumen alrededor de 10 segundos cada vez que se va al punto. Sin embargo, cuando se imprentó la política FBFS, se mejoró el tiempo respecto a la política LBFS y a la programación lineal, a continuación se muestran los tiempos.

1. Con la secuencia pre-programada le tomó 17:39 minutos para producir 3 piezas.
2. Con el algoritmo de control LBFS le tomó 19:30 minutos para producir 3 piezas.
3. Con el algoritmo de control FBFS le tomó 14:56 minutos para producir 3 piezas.

11 CONCLUSIONES Y RECOMENDACIONES.

Se considera que se cumplió con los objetivos planteados inicialmente, las principales conclusiones son las siguientes:

- Se desarrolló un algoritmo de control, con base a los lineamiento propuesto para una FMS teórica, descrita en [**1**]. .
- Se desarrolló dos algoritmos de control para una FMS de laboratorio, basados en políticas LBFS y FBFS. Y la política FBFS es más eficiente que LBFS presentando un mejorar en tiempo de producción de un 24.6%.
- Para ambas celdas de manufactura, tanto teórica como la de laboratorio, se realizaron simulaciones para estudiar su dinámica, así como sus políticas.
- Se implementó dos políticas para la FMS en PLC Allen-Bradley PS Logix 5330.
- La forma en que se programó, es una diferente alternativa de plantear el algoritmo de control, debido a que es fácil cambiar la secuencia o el control sin necesidad de cambiar todo el programa. Una vez que se tiene el modelo en el PLC es suficiente con cambiar la función de la política de control para que el sistema se comporte de manera diferente.
- Se concluye que el método matricial es de difícil aplicación a sistemas FMRF y para sistemas híbridos como casó [1] prácticamente no hay literatura conocida al respecto, por lo que fue necesario plantear nuevas reglas para desarrollar el controlador

12 APORTES

Se propuso 4 lineamiento de control para caldas de manufactura hibridas, que controlaron adecuadamente la celda teórica. De ser aplicables y generalizables las reglas a cualquier celda de manufactura, se estaría ante un procedimiento de diseño, rápido y sencillo.

13 REFERENCIA BIBLIOGRÁFICA

[1] Shihsen Peng and Mengchu Zhou, "Sensor-based stage Petri net modelling of PLC logic programs for Discrete-event control design," *International Journal of Production Research*, vol. 41, no. 3, pp. 629-644, 2003.

[2] Stjepan Bogdan, Frank L. Lewis, Zadenko Kovacic, and José Mireles, *Manufacturing System Control Design: A Matrix-based Approach*. London: Springer-Verlag, 2006.

[3] M. Groover, *Automation, Production System and Computer Integrated Manufacturing*. New Jersey: Prentice-Hall, 1987.

[4] José Mireles and Frank L. Lewis, "Intellligent Material Handing: Development and Implementation of Matrix-Based Discrete-Event Controller," *IEEE Transactions on Industrial Electronics*, vol. 48, no. 6, pp. 1087-1097, Diciembre 2001.

[5] Stjepan Bogdan, Frank L. Lewis, Zadenko Kovacic, Ayla Gurel, and Mario Stajdohar, "An Implementation of Matrix-Based Supervisory Control of Flexible Manufacturing Systems," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 5, pp. 709-725, September 2002.

[6] José Mirelles and Frank L. Lewis, "Deadlock analysis and routing on free-choice multipart reentrant flow lines using a matrix-based discrete event controller," in *Proceedings of the 41st IEEE Conference on Decision and Control*, vol. 1, Las Vegas, 2002, pp. 793 - 798.

[7] Prasanna Mohan Ballal, "Decision and Control of Distributed Cooperative Systemes," University of Texas, Arlingtong, USA, Ph.D Thesis 2008.

[8] Stjepan Bogdan, Zdenko Kovacic, Nenad Smolic-Rocak, and Bruno Birgmajer, "A matrix approach to an FMS control design," *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 92-109, Dic 2004.

[9] Hsiang-Hsi Huang, "Matrix Controller Design and Deadlock Analysis of Automated Manufacturing Systems. Part 1. Designing the Matrix-Based Controller," *The International Journal of Advaced Manufacturing Technology*, vol. 18, no. 6, pp. 434-447, 2001.

[10] Frank L. Lewis, Ayla Gurel, Stjepan Bogdan, Alper Doganalp, and Octavian C. Pastravanu, "Analysis of Deadlock and Circular Waits Using a Matrix Model for Flexible Manufacturing Systems," vol. 34, no. 9, pp. 1083-1100, 1998.

[11] Hsiang-Hsi Huang, "Matrix Controller Design and Deadlock Analysis of Automated Manufacturing Systems. Part 2: Deadlock Avoidance Policy," *The International Journal of Advanced Manufacturing Technology*, vol. 18, no. 7, pp. 490-501, 2001.

[12] MengChu Zhou and Maria Pia Fanti, Eds., *Deadlock Resolution in Computer Integrated Systems*. New York: Mercel Dekker, 2004.

Hsiang-Hsi Huang, Frank L. Lewis, and Diego Tacconi, "Deadlock analysis using a new

[13] matrix-based controller for reentrant flow line design," in *Proceedings IEEE IECON 22nd International Conference*, Taiwan, 1996, pp. 463-468.

[14] Diego A. Tacconi and Frank L. Lewis, "A new matrix model for discrete event systems: application to simulation," *IEEE Control Systems Magazine*, vol. 17, no. 5, pp. 62-71, October 1997.

[15] P. Ballal, F. Lewis, J. Mireles, and K Sreenath, "Deadlock avoidance for free choice multi-reentrant flow lines: Critical siphons and critical subsystems," in *IEEE Proc. Mediterranean Conf. Control & Automation*, Athens, 2007.

[16] Abraham Silberschatz and Peter C. Galvin, *Sistemas Operativos*, 5th ed. Mexico: Addison Wesley, 1999.

[17] Tadao Murata, "Petri nets: Properties, Analysis and Application," *Proceeding of the IEEE*, vol. 77, no. 4, pp. 541-580, 1989.

[18] Tien-Chien Chang, Richard A. Wysk, and Hsu-Pin Wang, *Computer Aided Manufacturing*. New Jersey: Prentice Hall, 1998.

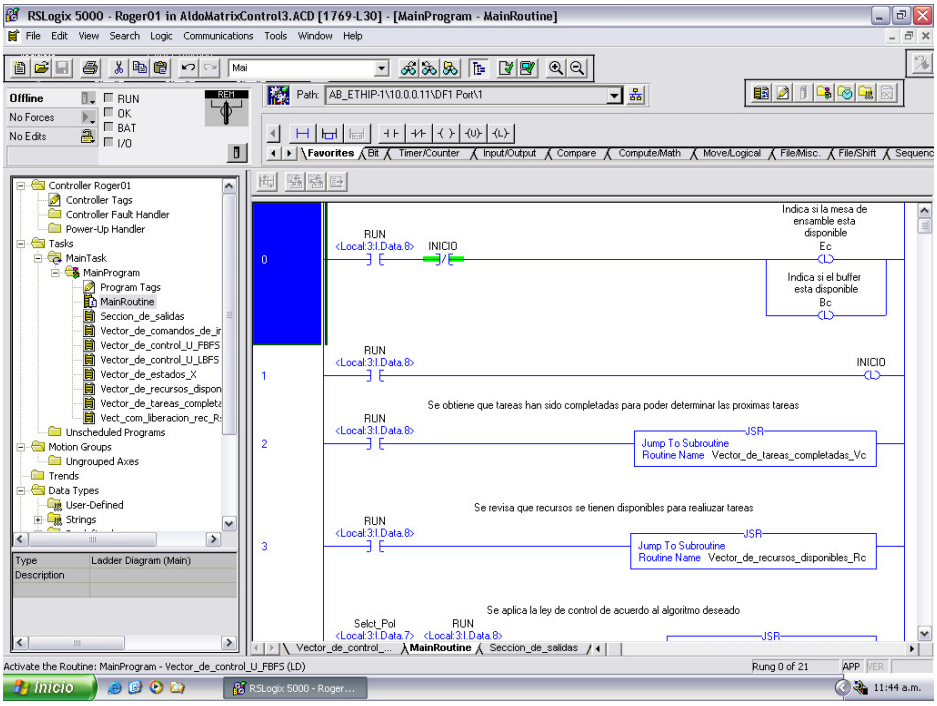
[19] Manuel Silva, "Petri nets and Production Systems," *LNCS: Lecture on Petri Nets Application II*, no. 1492, pp. 85-124, 1998.

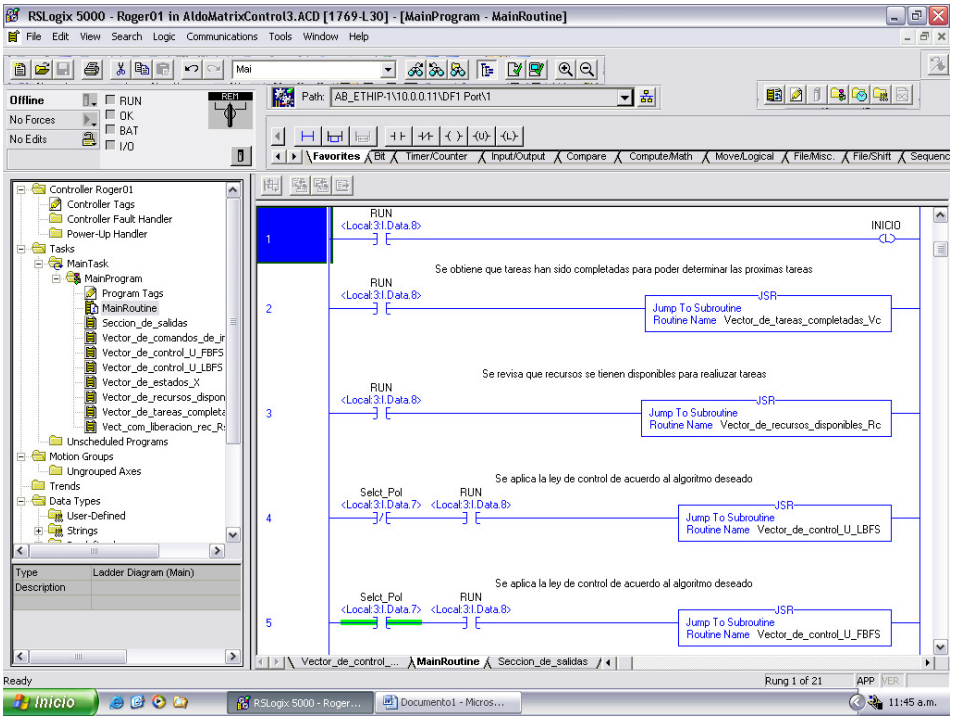
[20] MengChu Zhou and Venkatesh Kurapati, *Modeling, Simulation and Control of Manufacturing Systems: A Petri nets Approach*. New Jersey: World Scientific, 2000.

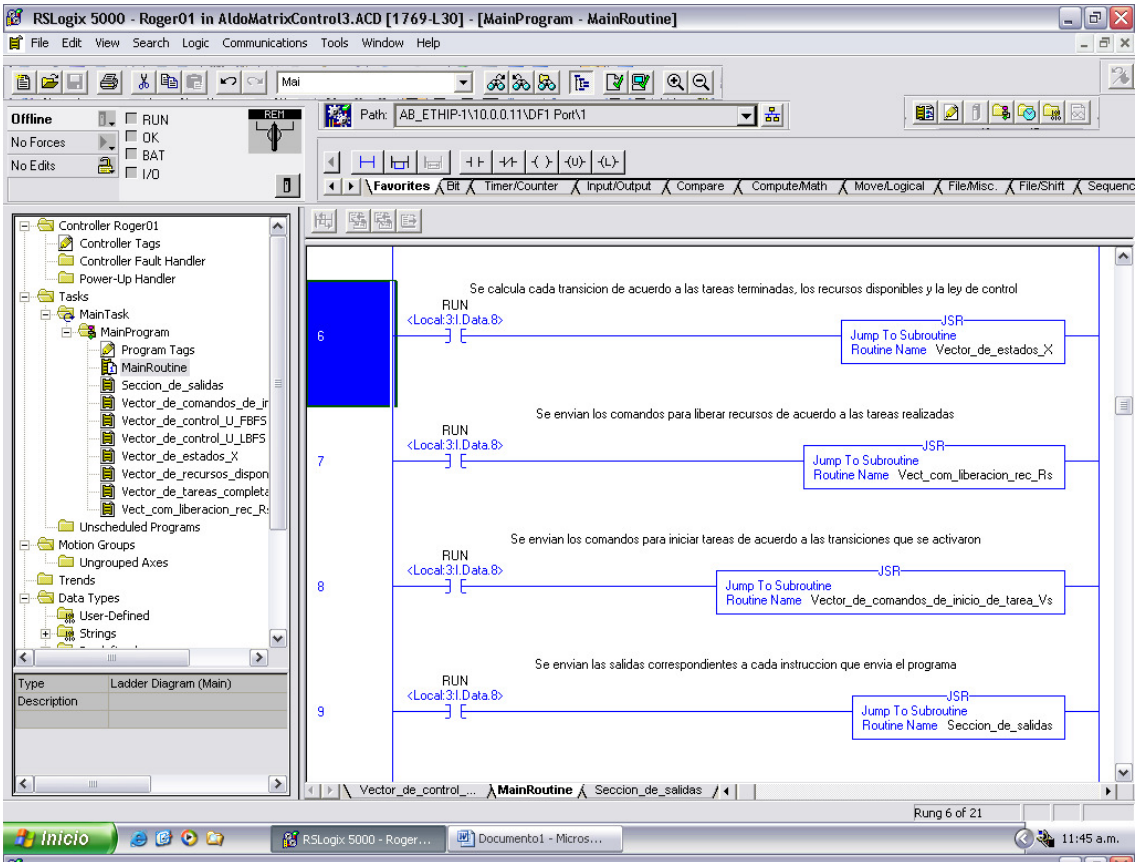
[21] Branislav Hruz and MengChu Zhou, *Modelling and Control of Discrete-event Dynamic Sytems - (Advanced texbooks in controland signal processing)*. London: Springer-Verlag, 2007.

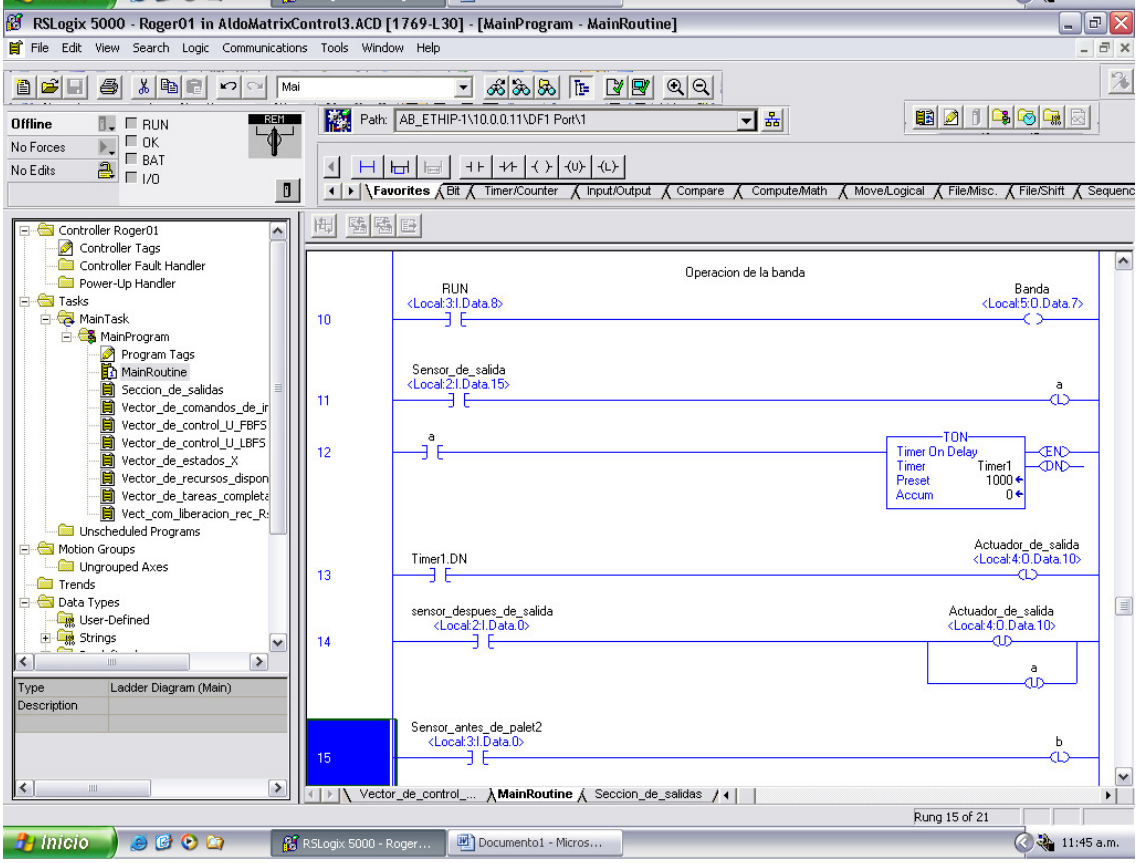
14 APENDICE 1: PROGRAMA PLC CELDA

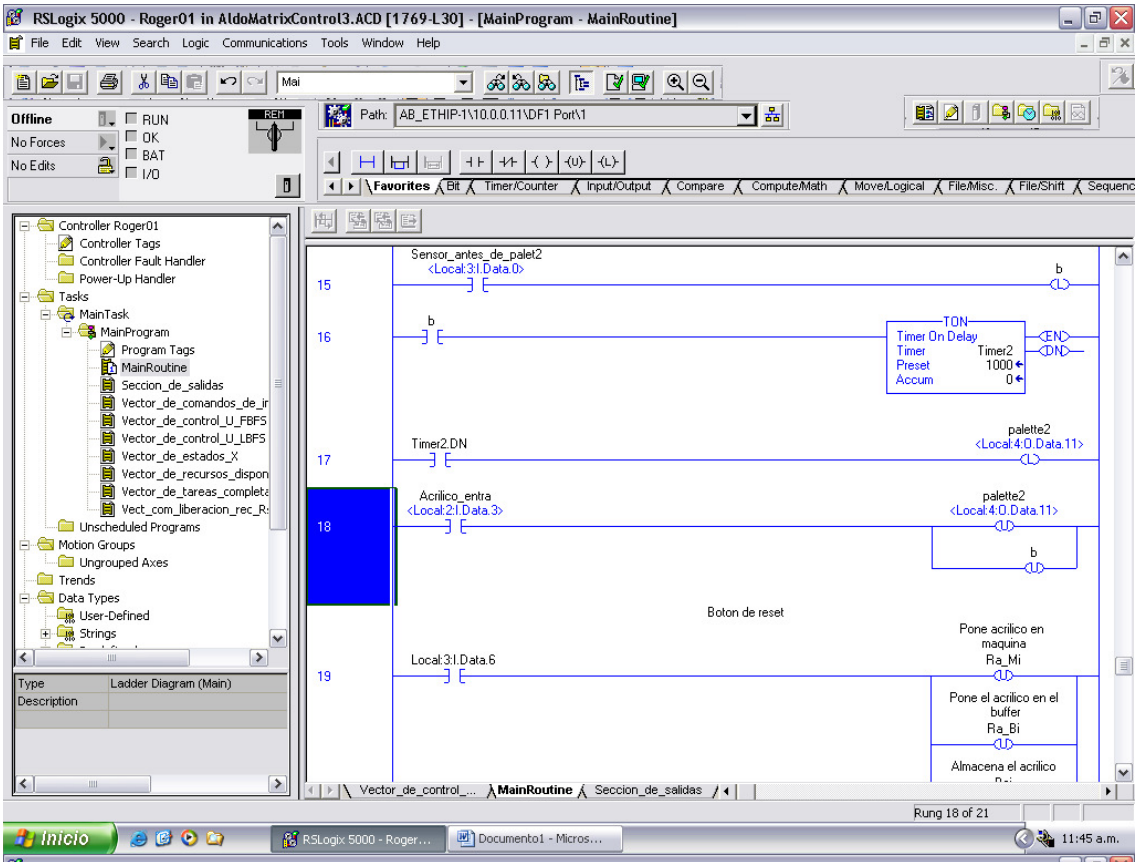
Programa del PLC

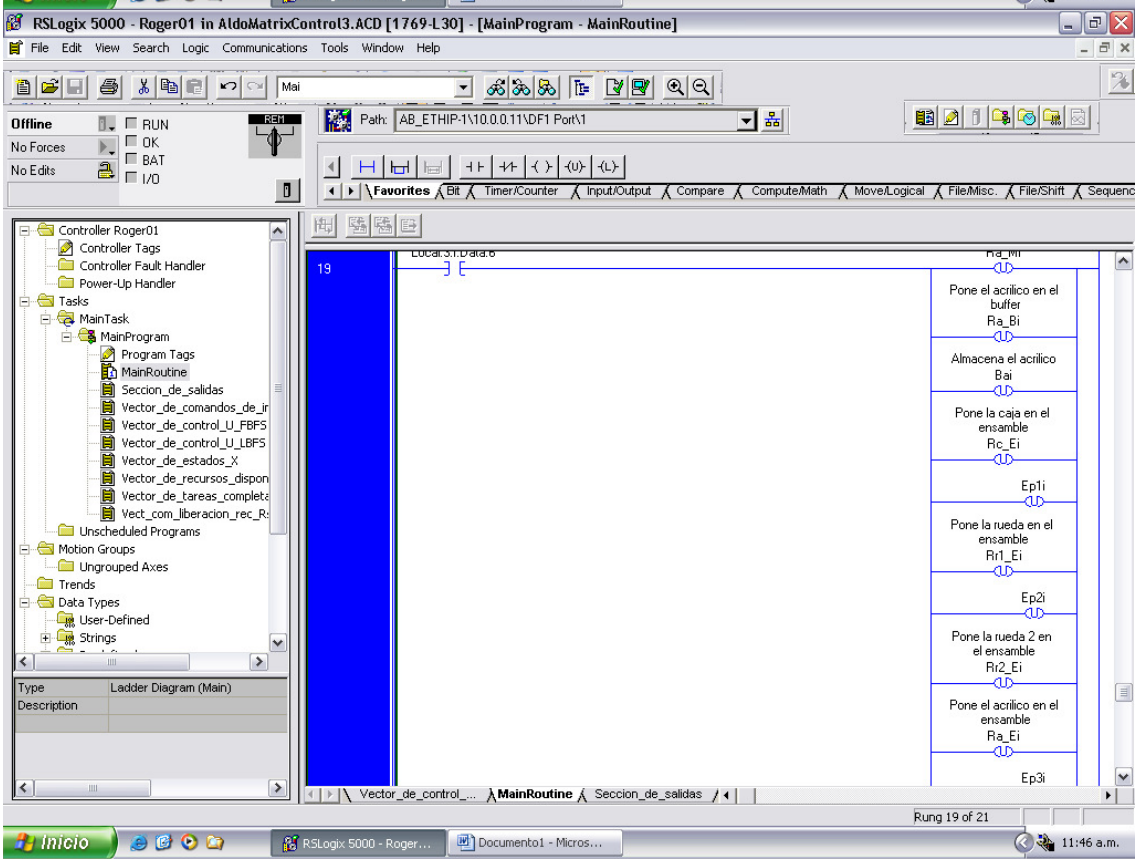


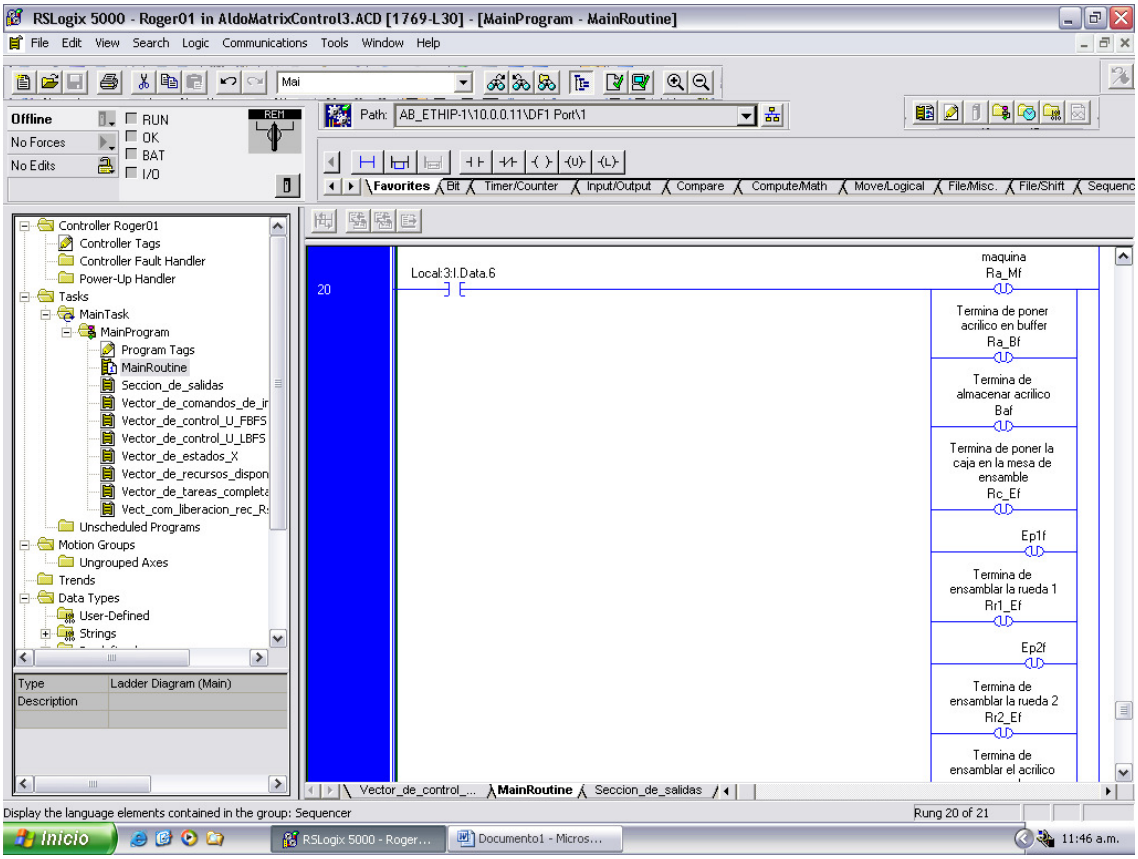



RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - MainRoutine]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_complete
Vect_com_liberacion_rec_Rs
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram (Main)
Description
Se calcula cada transicion de acuerdo a las tareas terminadas, los recursos disponibles y la ley de control
6
RUN
<Local:3:I.Data.8>
JSR
Jump To Subroutine
Routine Name Vector_de_estados_X
Se envian los comandos para liberar recursos de acuerdo a las tareas realizadas
7
RUN
<Local:3:I.Data.8>
JSR
Jump To Subroutine
Routine Name Vect_com_liberacion_rec_Rs
Se envian los comandos para iniciar tareas de acuerdo a las transiciones que se activaron
8
RUN
<Local:3:I.Data.8>
JSR
Jump To Subroutine
Routine Name Vector_de_comandos_de_inicio_de_tarea_Vs
Se envian las salidas correspondientes a cada instruccion que envia el programa
9
RUN
<Local:3:I.Data.8>
JSR
Jump To Subroutine
Routine Name Seccion_de_salidas
Vector_de_control_... MainRoutine Seccion_de_salidas
Rung 6 of 21
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:45 a.m.

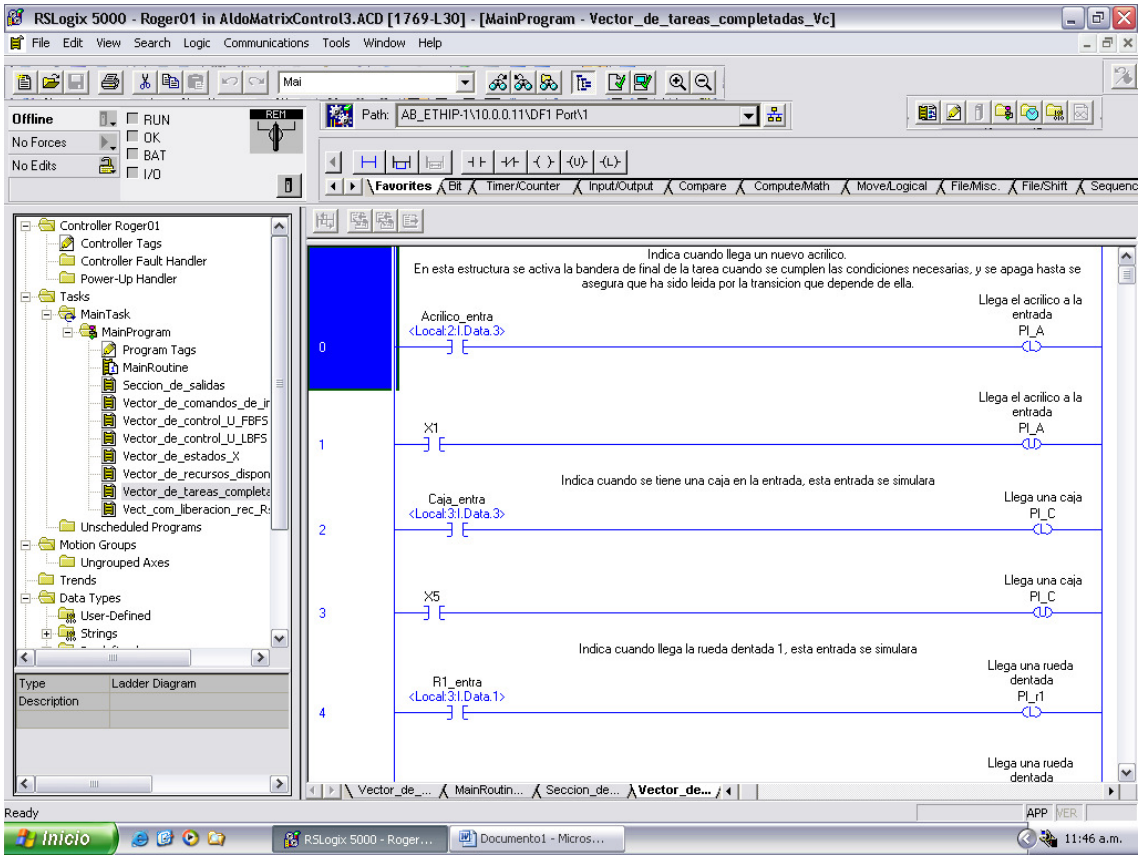

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - MainRoutine]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_complete
Vect_com_liberacion_rec_Rs
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram (Main)
Description
Operacion de la banda
10
RUN
<Local:3:I.Data.8>
Banda
<Local:5:O.Data.7>
11
Sensor_de_salida
<Local:2:I.Data.15>
a
12
a
TON
Timer On Delay
Timer Timer1
Preset 1000
Accum 0
EN
DN
13
Timer1.DN
Actuador_de_salida
<Local:4:O.Data.10>
14
sensor_despues_de_salida
<Local:2:I.Data.0>
Actuador_de_salida
<Local:4:O.Data.10>
a
15
Sensor_antes_de_palet2
<Local:3:I.Data.0>
b
Vector_de_control_... MainRoutine Seccion_de_salidas
Rung 15 of 21
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:45 a.m.

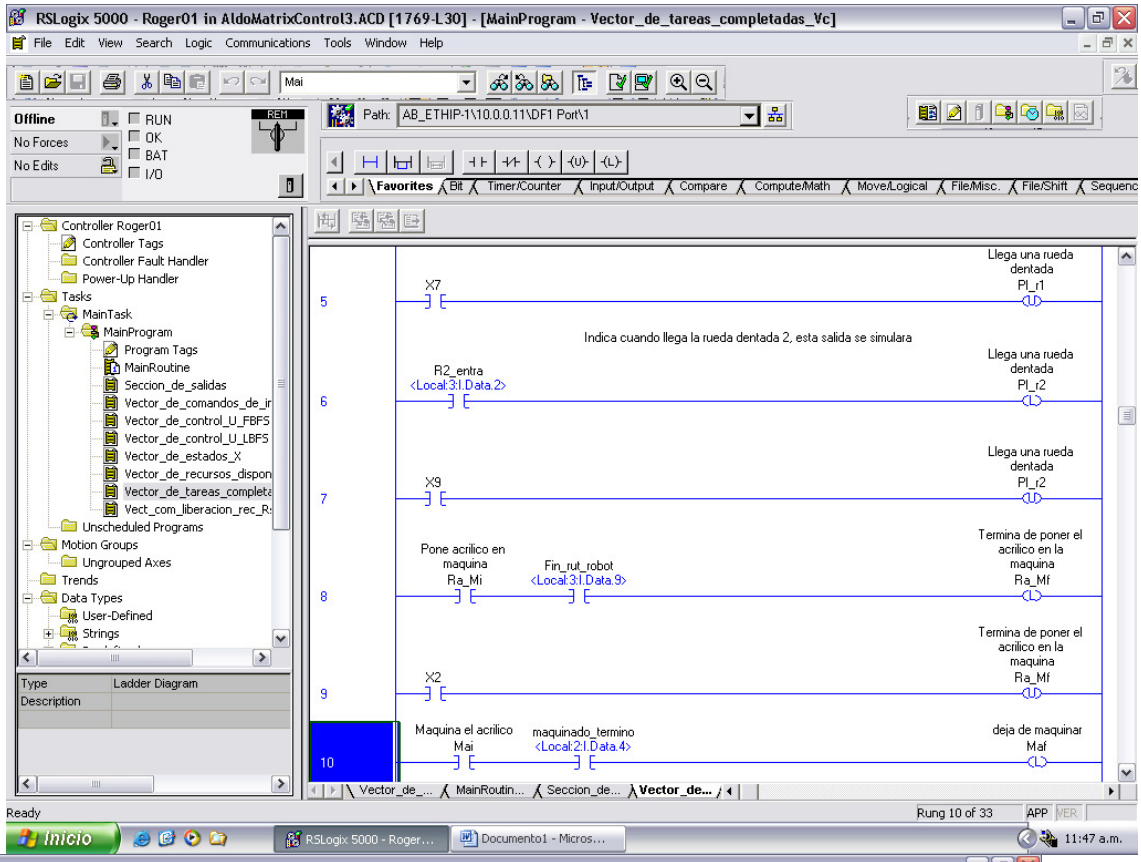

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - MainRoutine]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_complete
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram (Main)
Description
Sensor_antes_de_palet2
<Local:3:I.Data.0>
b
TON
Timer On Delay
Timer Timer2
Preset 1000
Accum 0
EN
DN
Timer2.DN
palette2
<Local:4:O.Data.11>
Acrilico_entra
<Local:2:I.Data.3>
Boton de reset
Local:3:I.Data.6
Pone acrilico en maquina
Ra_Mi
Pone el acrilico en el buffer
Ra_Bi
Almacena el acrilico
Vector_de_control_... MainRoutine Seccion_de_salidas
Rung 18 of 21
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:45 a.m.

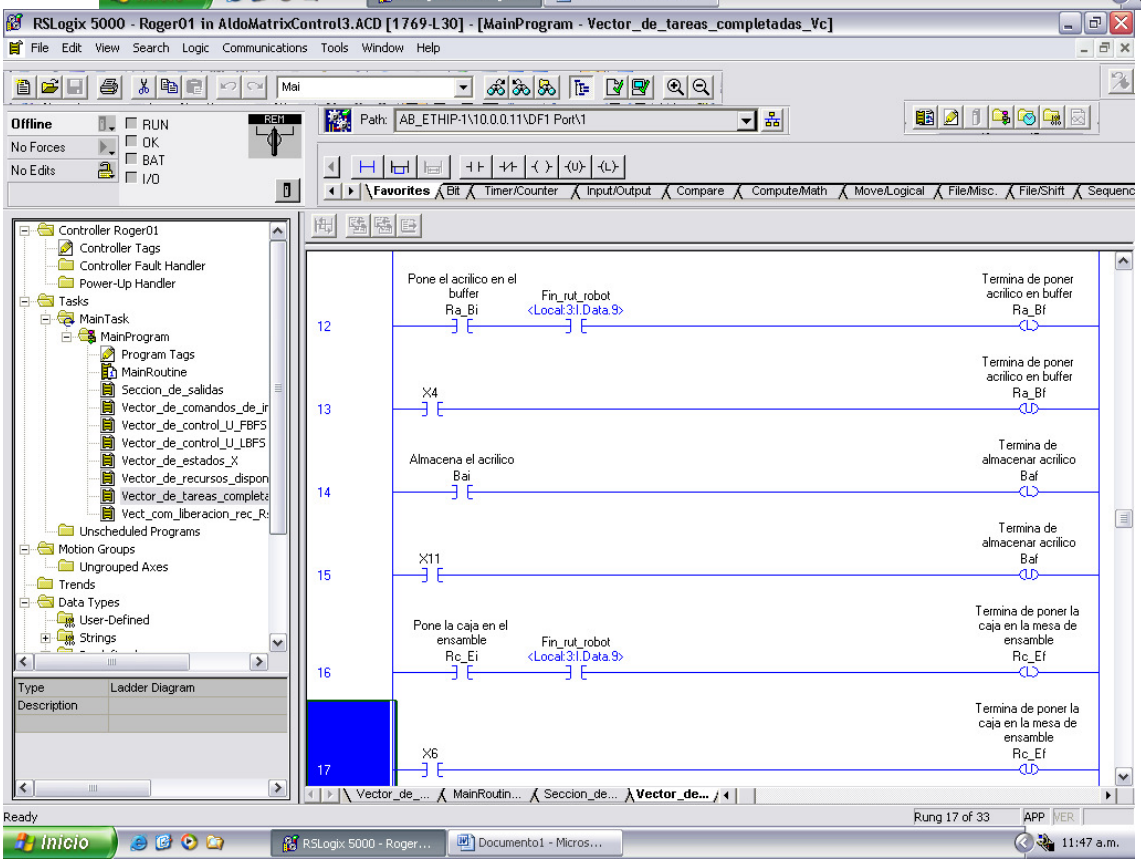

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - MainRoutine]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_complete
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram (Main)
Description
Local:3:I.Data.6
Pone el acrilico en el buffer
Ra_Bi
Almacena el acrilico
Bai
Pone la caja en el ensamble
Rc_Ei
Ep1i
Pone la rueda en el ensamble
Rr1_Ei
Ep2i
Pone la rueda 2 en el ensamble
Rr2_Ei
Pone el acrilico en el ensamble
Ra_Ei
Ep3i
Vector_de_control_... MainRoutine Seccion_de_salidas
Rung 19 of 21
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:46 a.m.

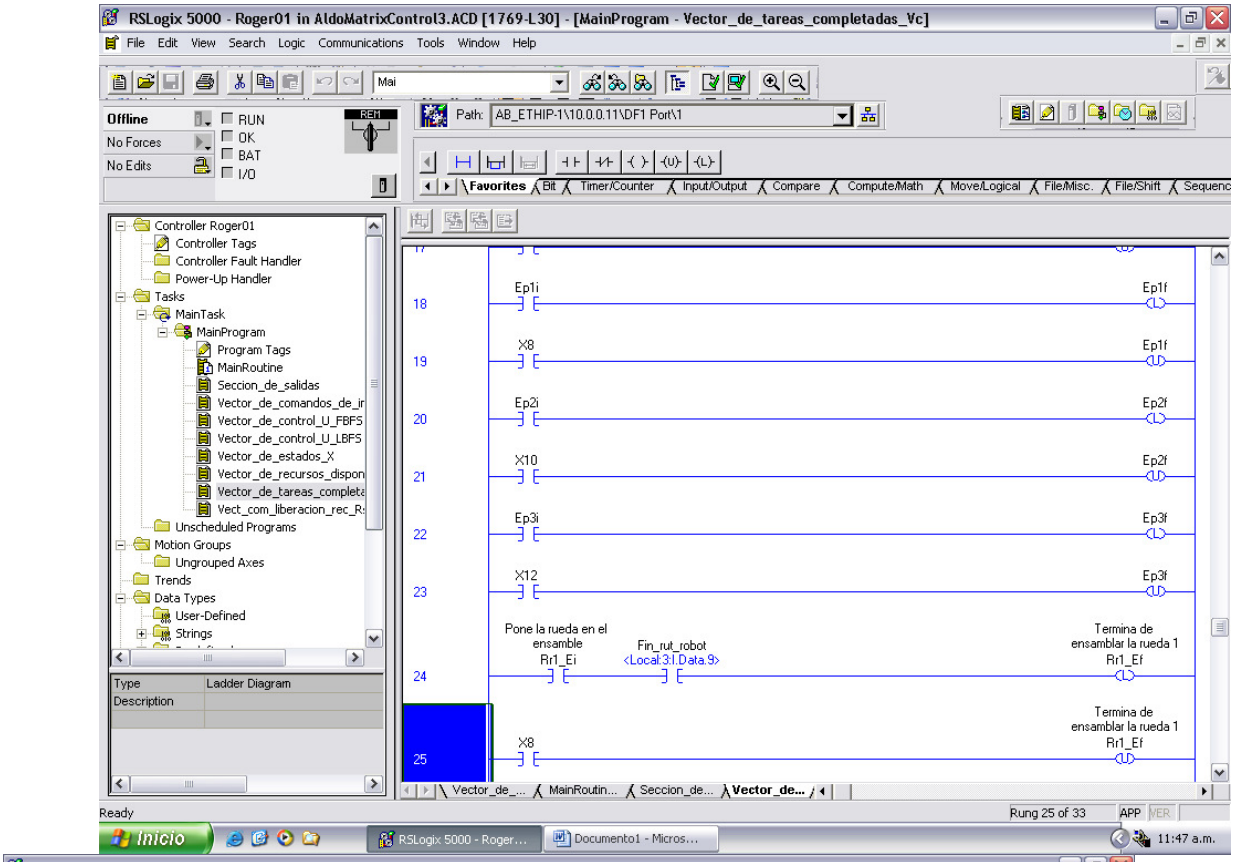


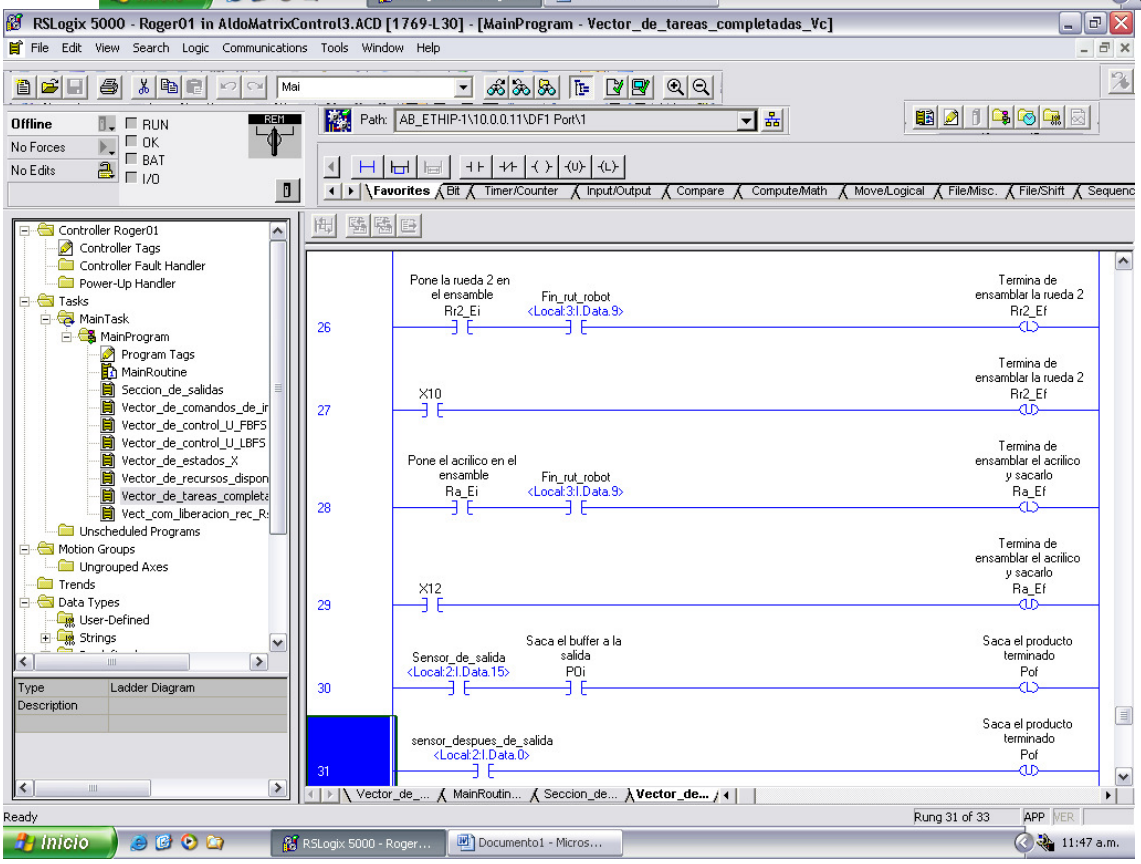
Tareas completadas

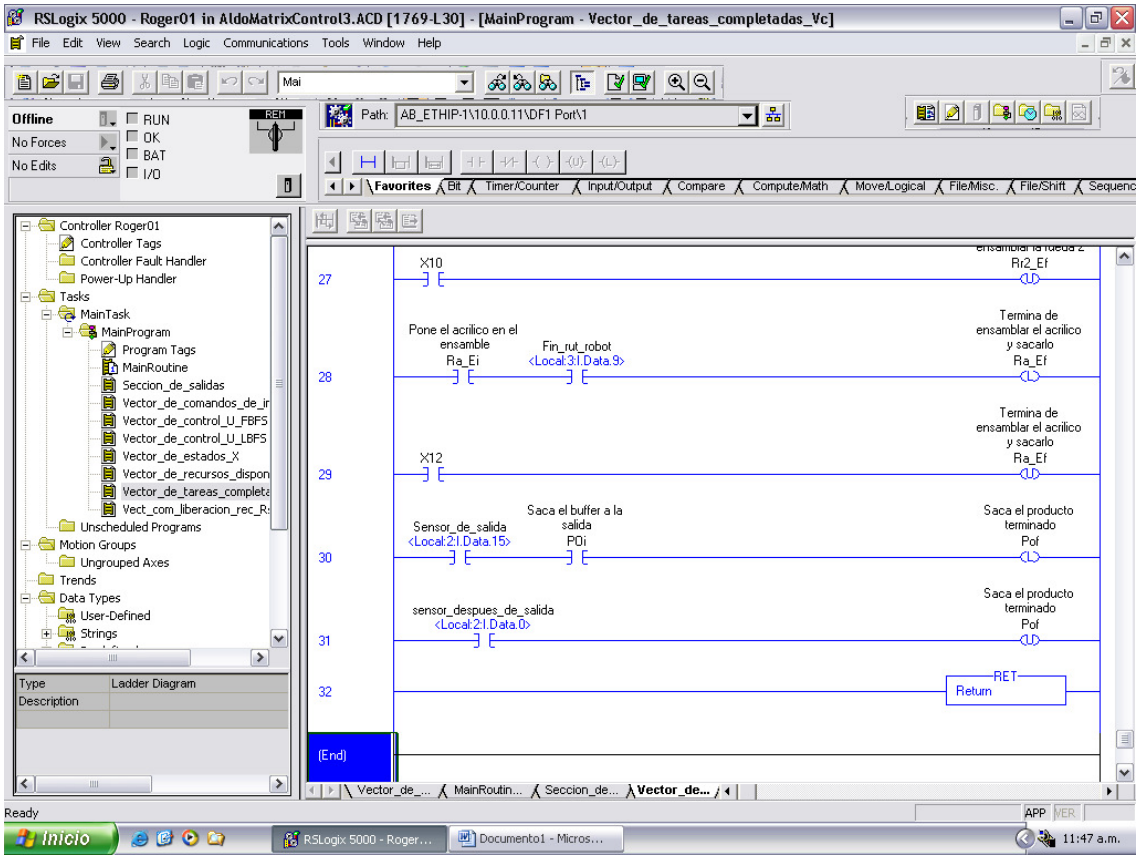

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_tareas_completadas_Vc]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
Indica cuando llega un nuevo acrilico.
En esta estructura se activa la bandera de final de la tarea cuando se cumplen las condiciones necesarias, y se apaga hasta se asegura que ha sido leida por la transicion que depende de ella.
Acrilico_entra
<Local:2:I.Data.3>
Llega el acrilico a la entrada
PI_A
X1
Llega el acrilico a la entrada
PI_A
Indica cuando se tiene una caja en la entrada, esta entrada se simulara
Caja_entra
<Local:3:I.Data.3>
Llega una caja
PI_C
X5
Llega una caja
PI_C
Indica cuando llega la rueda dentada 1, esta entrada se simulara
R1_entra
<Local:3:I.Data.1>
Llega una rueda dentada
PI_r1
Llega una rueda dentada
Vector_de_... MainRoutin... Seccion_de... Vector_de...
Ready
APP
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:46 a.m.


RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_tareas_completadas_Vc]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type
Ladder Diagram
Description
X7
Llega una rueda dentada
Pl_r1
Indica cuando llega la rueda dentada 2, esta salida se simulara
R2_entra
<Local:3:I.Data.2>
Llega una rueda dentada
Pl_r2
X9
Llega una rueda dentada
Pl_r2
Pone acrilico en maquina
Ra_Mi
Fin_rut_robot
<Local:3:I.Data.9>
Termina de poner el acrilico en la maquina
Ra_Mf
Termina de poner el acrilico en la maquina
Ra_Mf
X2
Maquina el acrilico
Mai
maquinado_termino
<Local:2:I.Data.4>
deja de maquinar
Maf
Vector_de_... MainRoutin... Seccion_de... Vector_de...
Ready
Rung 10 of 33
APP
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:47 a.m.

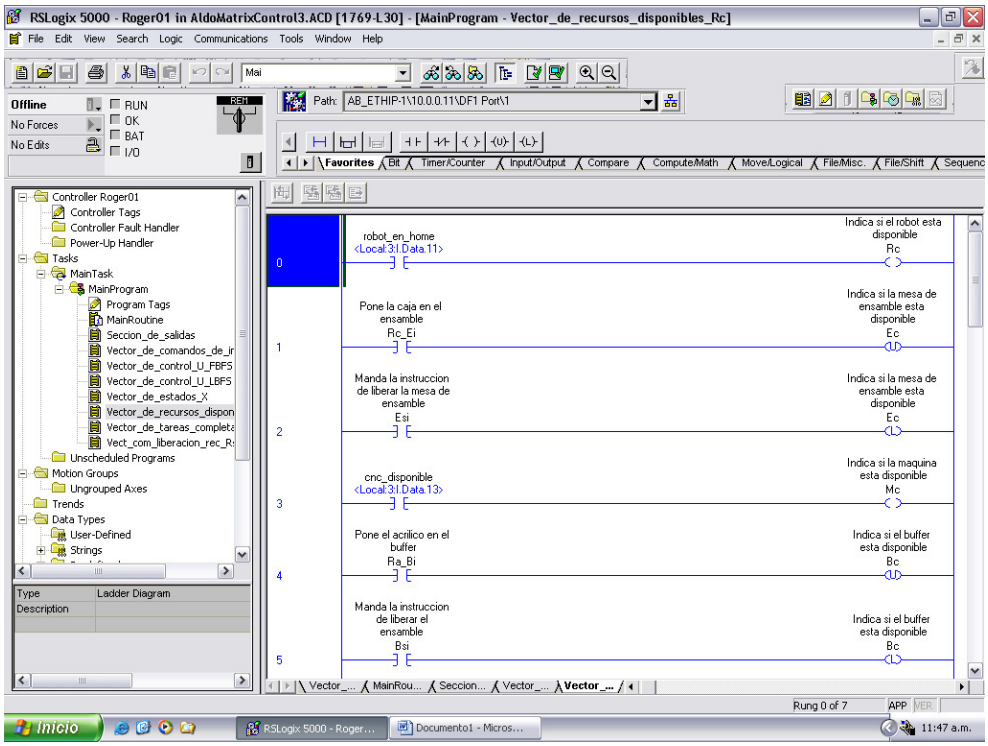

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_tareas_completadas_Vc]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type
Ladder Diagram
Description
Pone el acrilico en el buffer
Ra_Bi
Fin_rut_robot
<Local:3:I.Data.9>
Termina de poner acrilico en buffer
Ra_Bf
X4
Termina de poner acrilico en buffer
Ra_Bf
Almacena el acrilico
Bai
Termina de almacenar acrilico
Baf
X11
Termina de almacenar acrilico
Baf
Pone la caja en el ensamble
Rc_Ei
Fin_rut_robot
<Local:3:I.Data.9>
Termina de poner la caja en la mesa de ensamble
Rc_Ef
X6
Termina de poner la caja en la mesa de ensamble
Rc_Ef
Vector_de_... MainRoutin... Seccion_de... Vector_de...
Ready
Rung 17 of 33
APP
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:47 a.m.

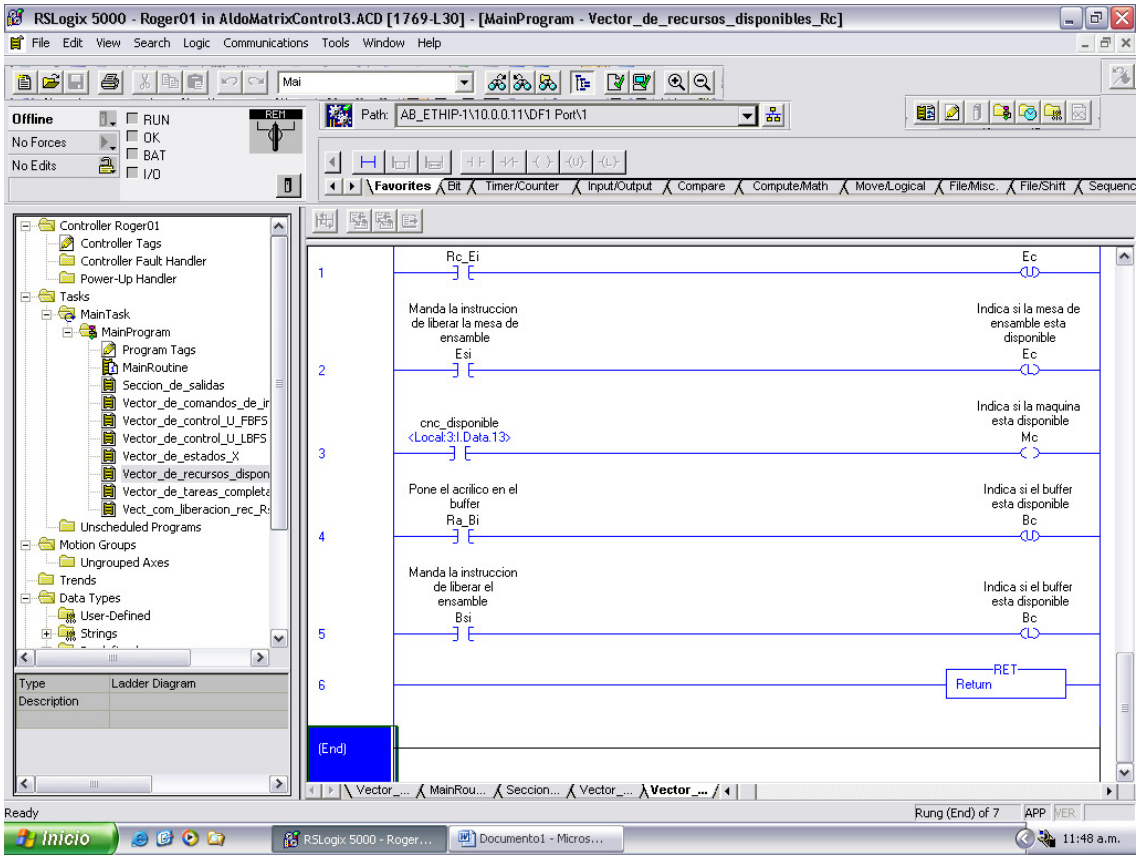

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_tareas_completadas_Vc]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type
Ladder Diagram
Description
18 Ep1i Ep1f
19 X8 Ep1f
20 Ep2i Ep2f
21 X10 Ep2f
22 Ep3i Ep3f
23 X12 Ep3f
24 Pone la rueda en el ensamble Rr1_Ei
Fin_rut_robot <Local:3:I.Data.9>
Termina de ensamblar la rueda 1 Rr1_Ef
25 X8 Termina de ensamblar la rueda 1 Rr1_Ef
Vector_de_... MainRoutin... Seccion_de... Vector_de...
Ready
Rung 25 of 33
APP
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:47 a.m.


RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_tareas_completadas_Vc]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type
Ladder Diagram
Description
26 Pone la rueda 2 en el ensamble Rr2_Ei
Fin_rut_robot <Local:3:I.Data.9>
Termina de ensamblar la rueda 2 Rr2_Ef
27 X10 Termina de ensamblar la rueda 2 Rr2_Ef
28 Pone el acrilico en el ensamble Ra_Ei
Fin_rut_robot <Local:3:I.Data.9>
Termina de ensamblar el acrilico y sacarlo Ra_Ef
29 X12 Termina de ensamblar el acrilico y sacarlo Ra_Ef
30 Sensor_de_salida <Local:2:I.Data.15>
Saca el buffer a la salida POi
Saca el producto terminado Pof
31 sensor_despues_de_salida <Local:2:I.Data.0> Saca el producto terminado Pof
Vector_de_... MainRoutin... Seccion_de... Vector_de...
Ready
Rung 31 of 33
APP
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:47 a.m.

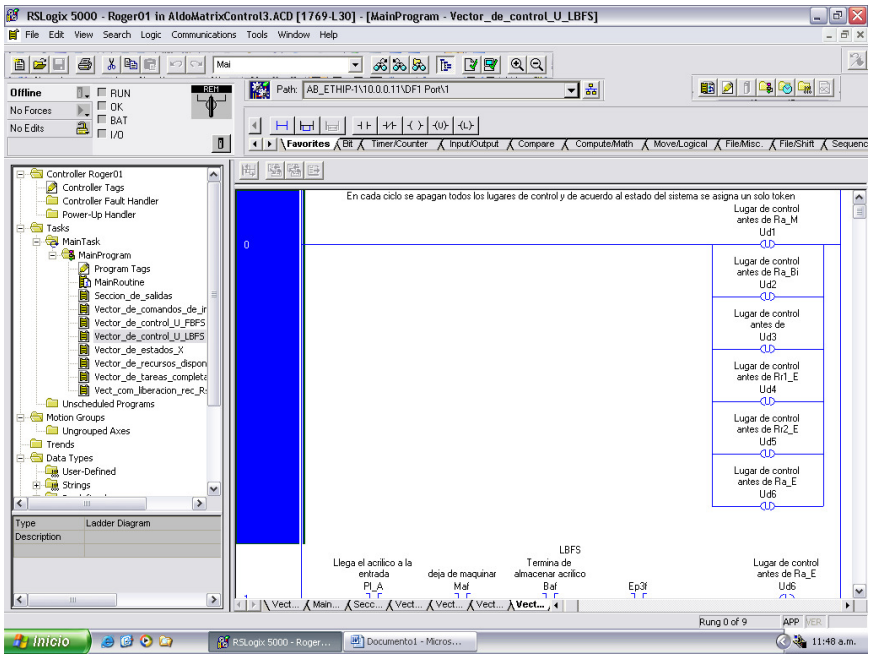


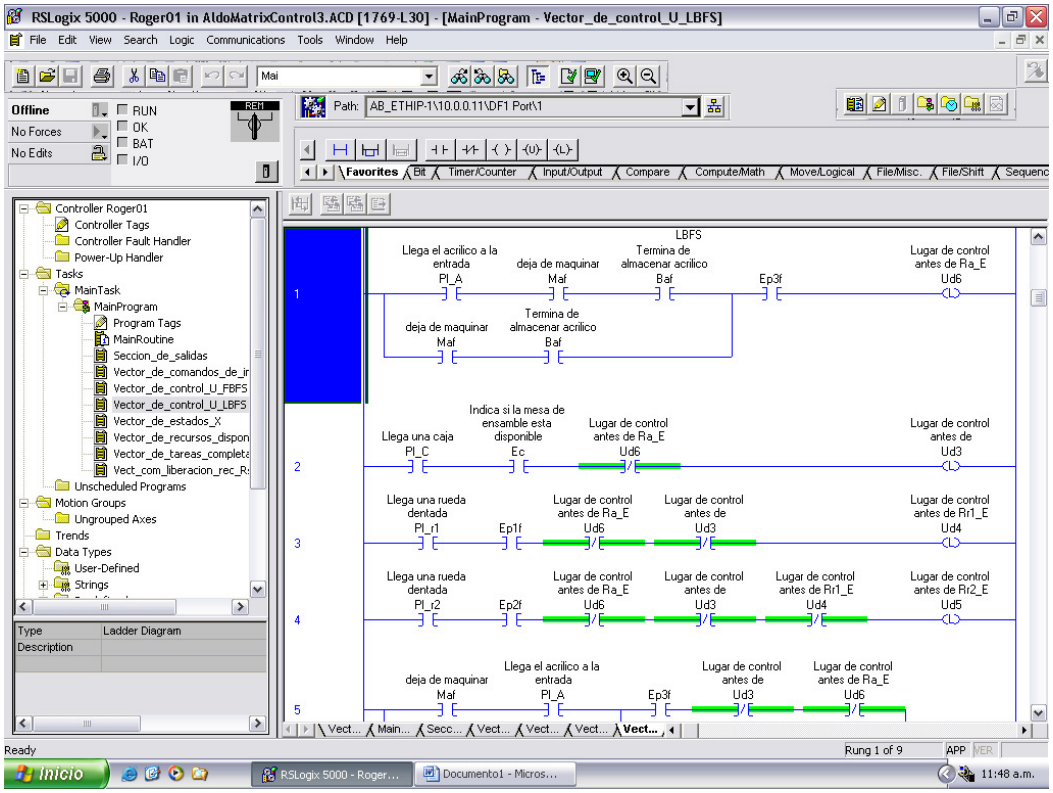
Vector de recursos disponibles

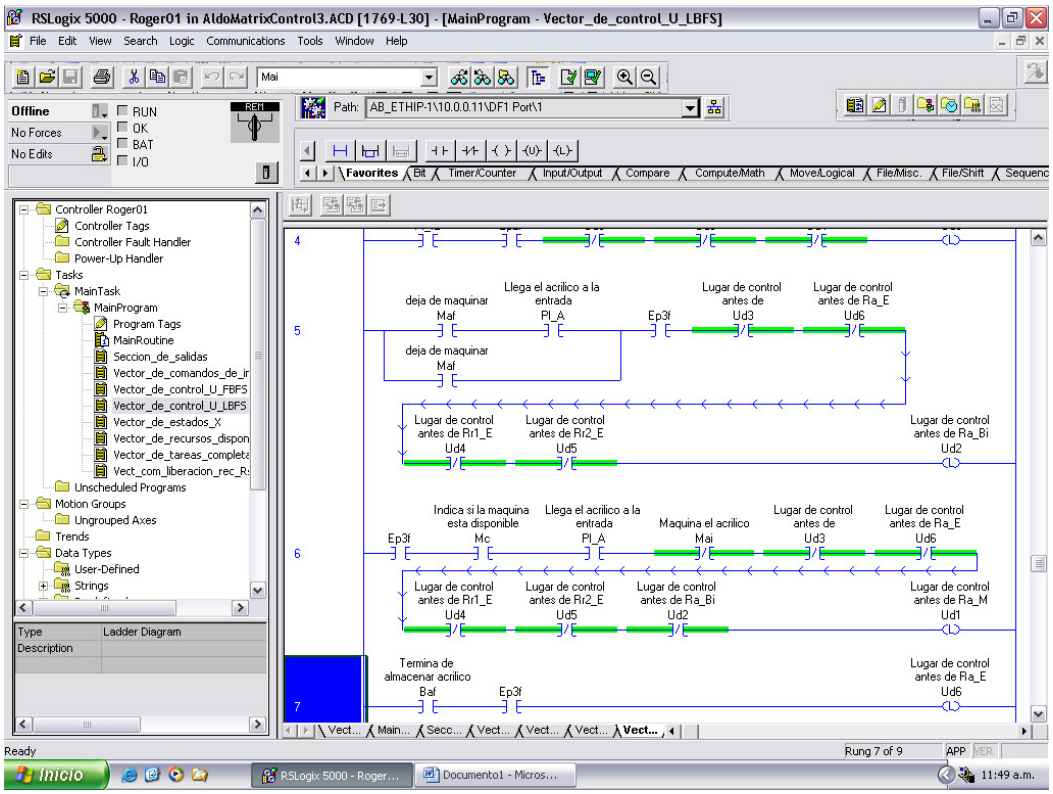


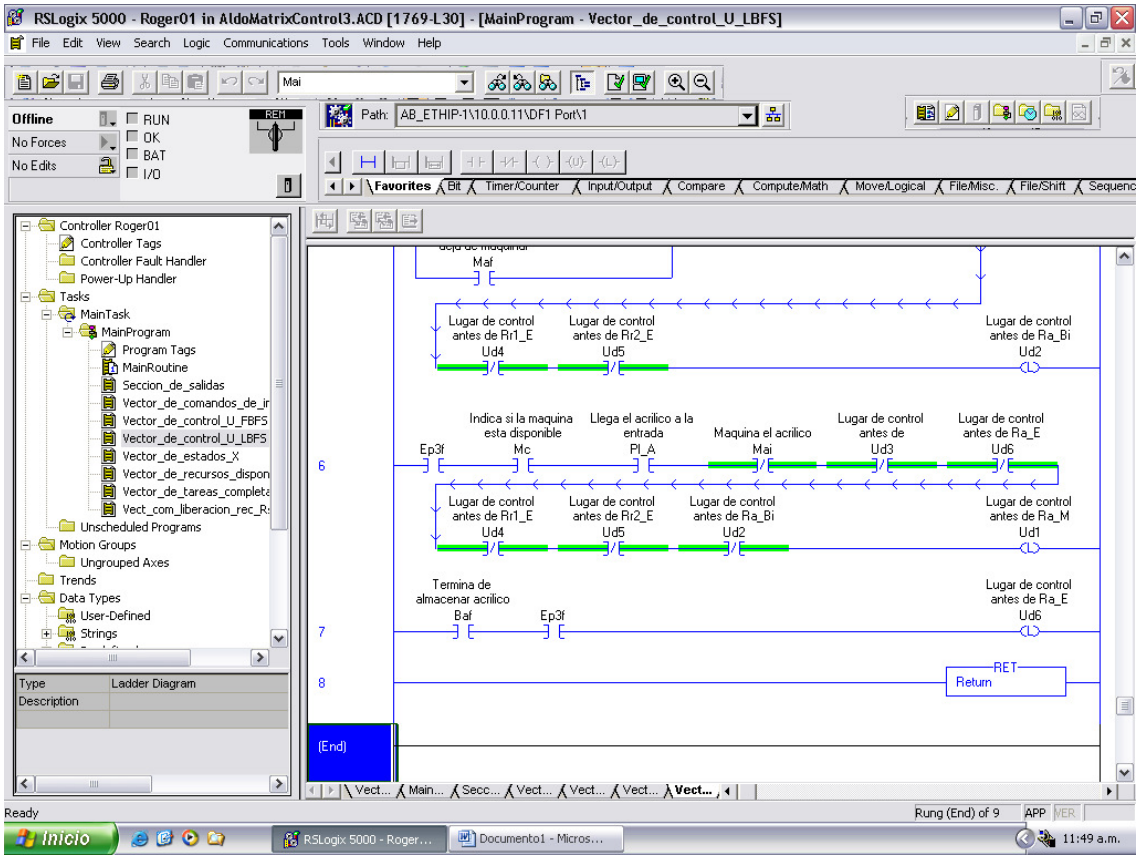

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_recursos_disponibles_Rc]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
Rc_Ei
Ec
Manda la instruccion de liberar la mesa de ensamble
Esi
Indica si la mesa de ensamble esta disponible
Ec
cnc_disponible
<Local:3:I.Data.13>
Indica si la maquina esta disponible
Mc
Pone el acrilico en el buffer
Ra_Bi
Indica si el buffer esta disponible
Bc
Manda la instruccion de liberar el ensamble
Bsi
Indica si el buffer esta disponible
Bc
RET
Return
(End)
Vector_... MainRou... Seccion... Vector_... Vector_...
Ready
Rung (End) of 7
APP
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:48 a.m.

LBFS

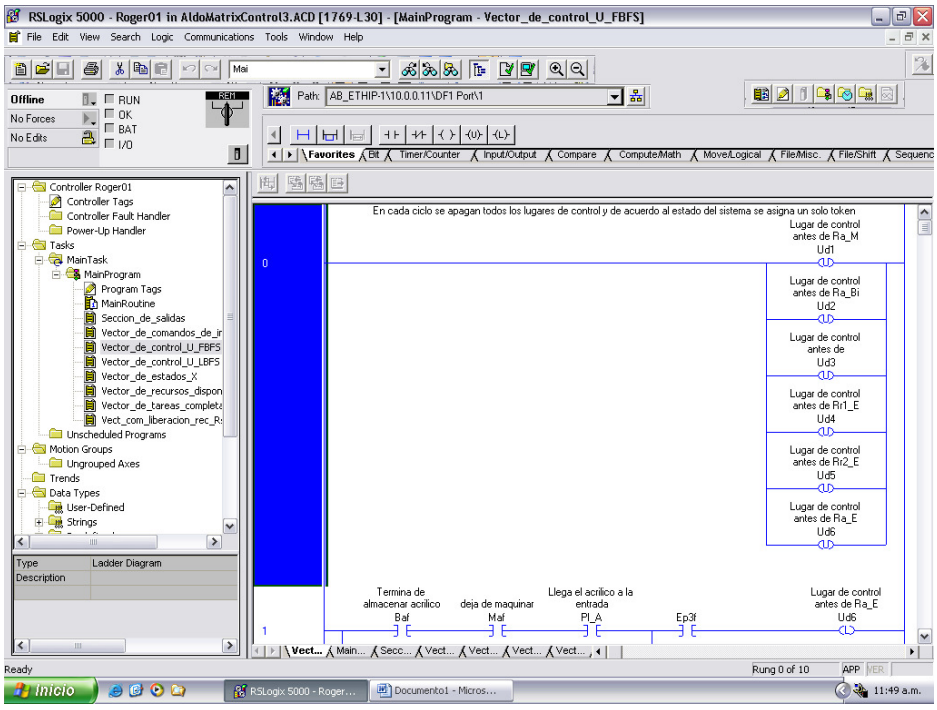

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_control_U_LBFS]
En cada ciclo se apagan todos los lugares de control y de acuerdo al estado del sistema se asigna un solo token
Lugar de control antes de Ra_M
Ud1
Lugar de control antes de Ra_Bi
Ud2
Lugar de control antes de
Ud3
Lugar de control antes de Rr1_E
Ud4
Lugar de control antes de Rr2_E
Ud5
Lugar de control antes de Ra_E
Ud6
Llega el acrilico a la entrada
PI_A
deja de maquinar
Maf
LBFS
Termina de almacenar acrilico
Baf
Ep3f
Lugar de control antes de Ra_E
Ud6
Rung 0 of 9

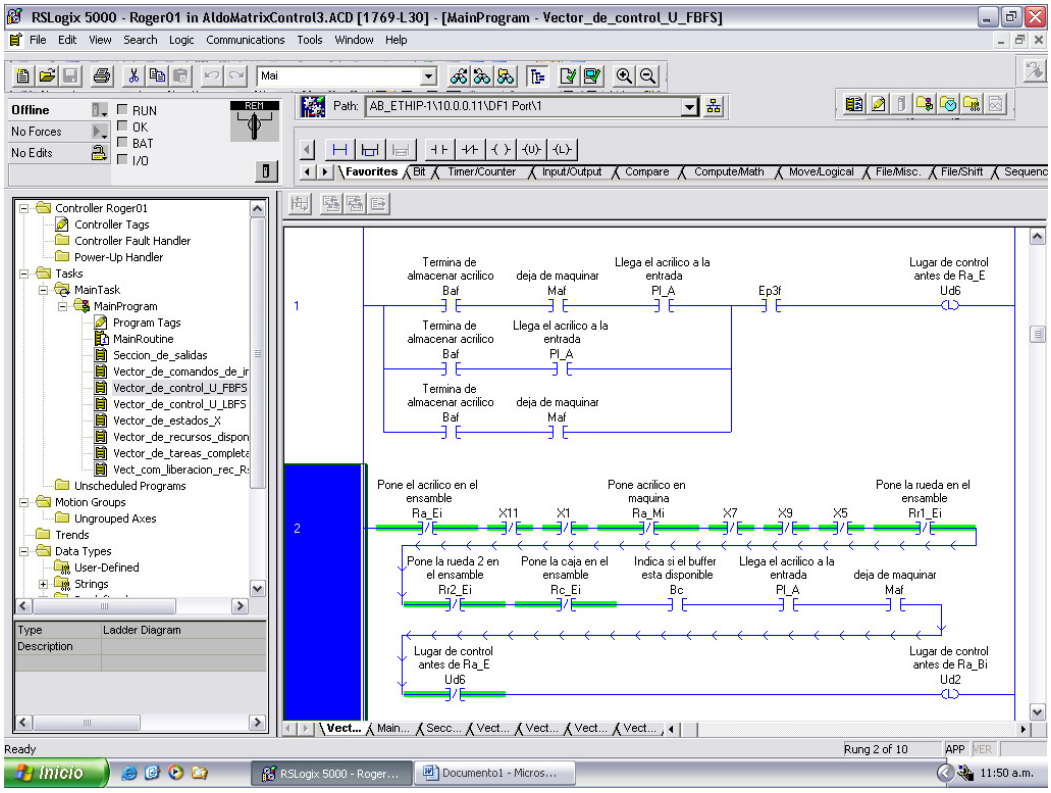

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_control_U_LBFS]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
LBFS
Llega el acrilico a la entrada PI_A
deja de maquinar Maf
Termina de almacenar acrilico Baf
Ep3f
Lugar de control antes de Ra_E Ud6
Llega una caja PI_C
Indica si la mesa de ensamble esta disponible Ec
Lugar de control antes de Ud3
Llega una rueda dentada PI_r1
Ep1f
Lugar de control antes de Rr1_E Ud4
Llega una rueda dentada PI_r2
Ep2f
Lugar de control antes de Rr2_E Ud5
Vect... Main... Secc... Vect... Vect... Vect... Vect...
Ready
Rung 1 of 9
APP
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:48 a.m.

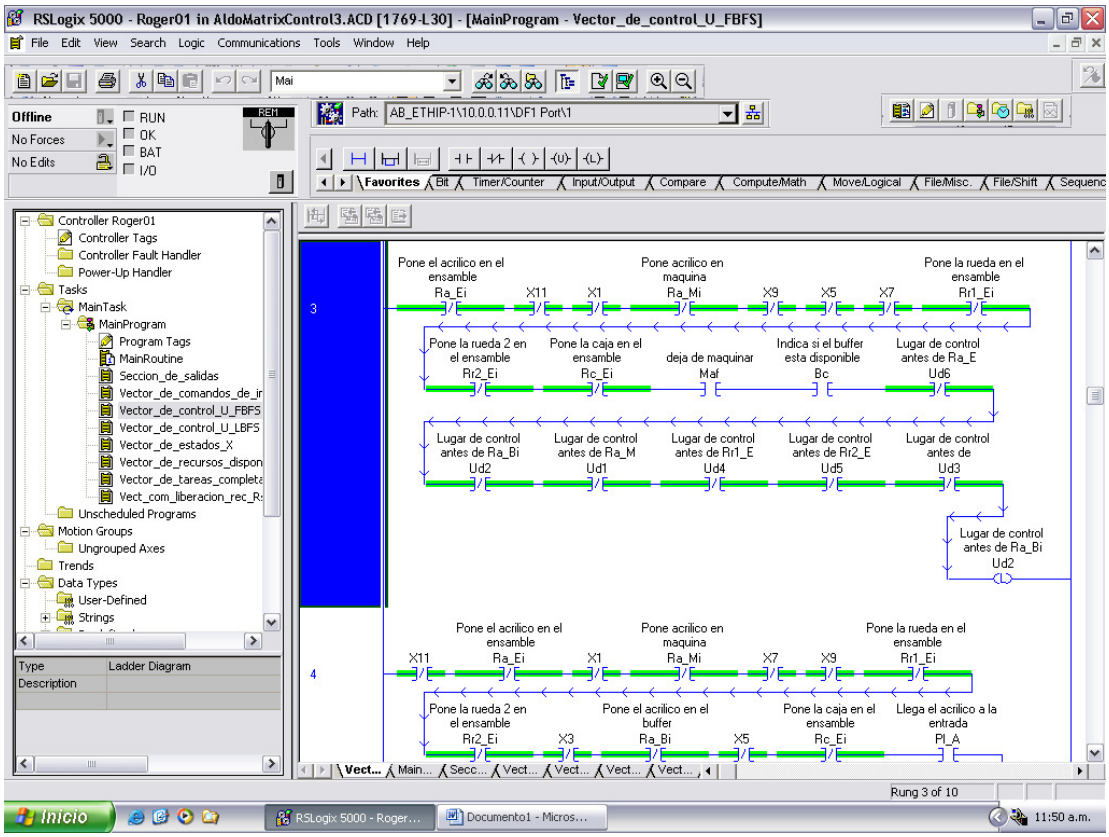

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_control_U_LBFS]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
deja de maquinar Maf
Llega el acrilico a la entrada PI_A
Ep3f
Lugar de control antes de Ud3
Lugar de control antes de Ra_E Ud6
Lugar de control antes de Rr1_E Ud4
Lugar de control antes de Rr2_E Ud5
Lugar de control antes de Ra_Bi Ud2
Indica si la maquina esta disponible Mc
Maquina el acrilico Mai
Lugar de control antes de Ra_M Ud1
Termina de almacenar acrilico Baf
Vect... Main... Secc... Vect... Vect... Vect... Vect...
Ready
Rung 7 of 9
APP
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:49 a.m.

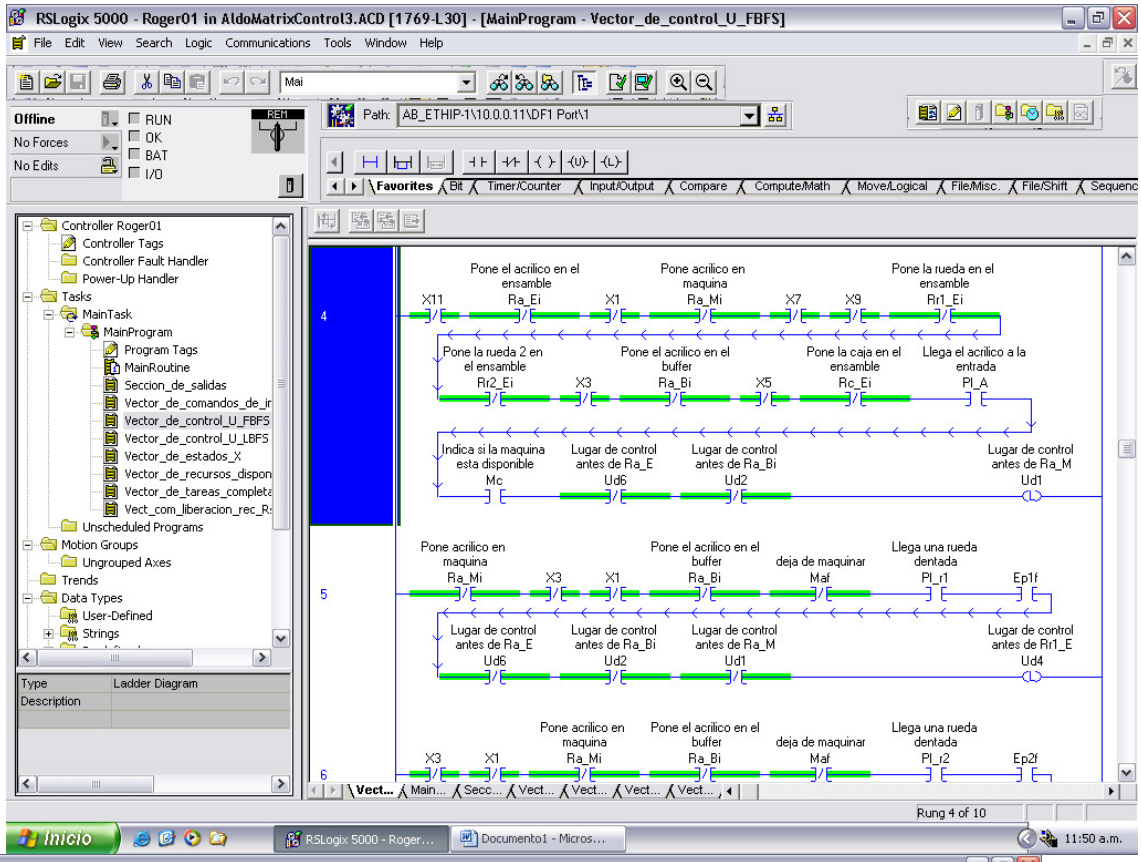

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_control_U_LBFS]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type
Ladder Diagram
Description
Maf
Lugar de control antes de Rr1_E Ud4
Lugar de control antes de Rr2_E Ud5
Lugar de control antes de Ra_Bi Ud2
Indica si la maquina esta disponible Mc
Llega el acrilico a la entrada PI_A
Maquina el acrilico Mai
Lugar de control antes de Ud3
Lugar de control antes de Ra_E Ud6
Ep3f
Lugar de control antes de Ra_M Ud1
Termina de almacenar acrilico Baf
RET
Return
(End)
Vect... Main... Secc... Vect... Vect... Vect... Vect...
Ready
Rung (End) of 9
APP
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:49 a.m.

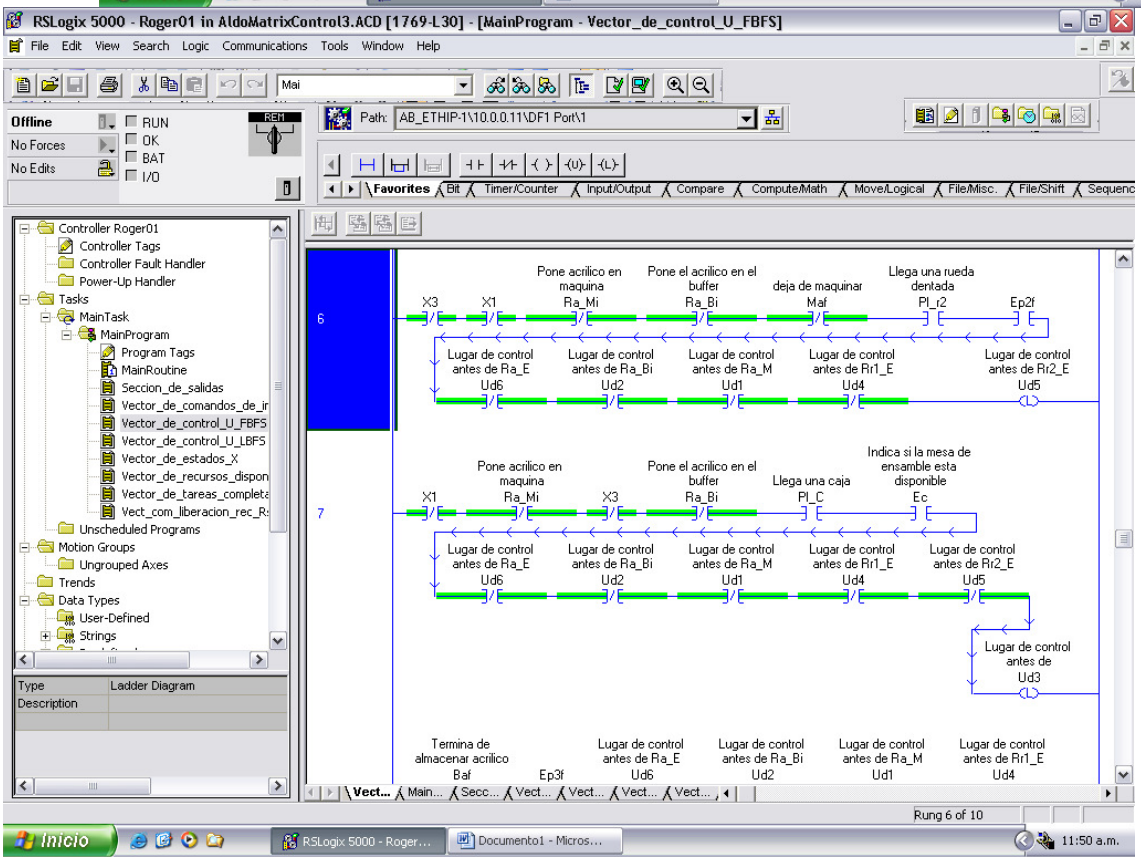
FBFS

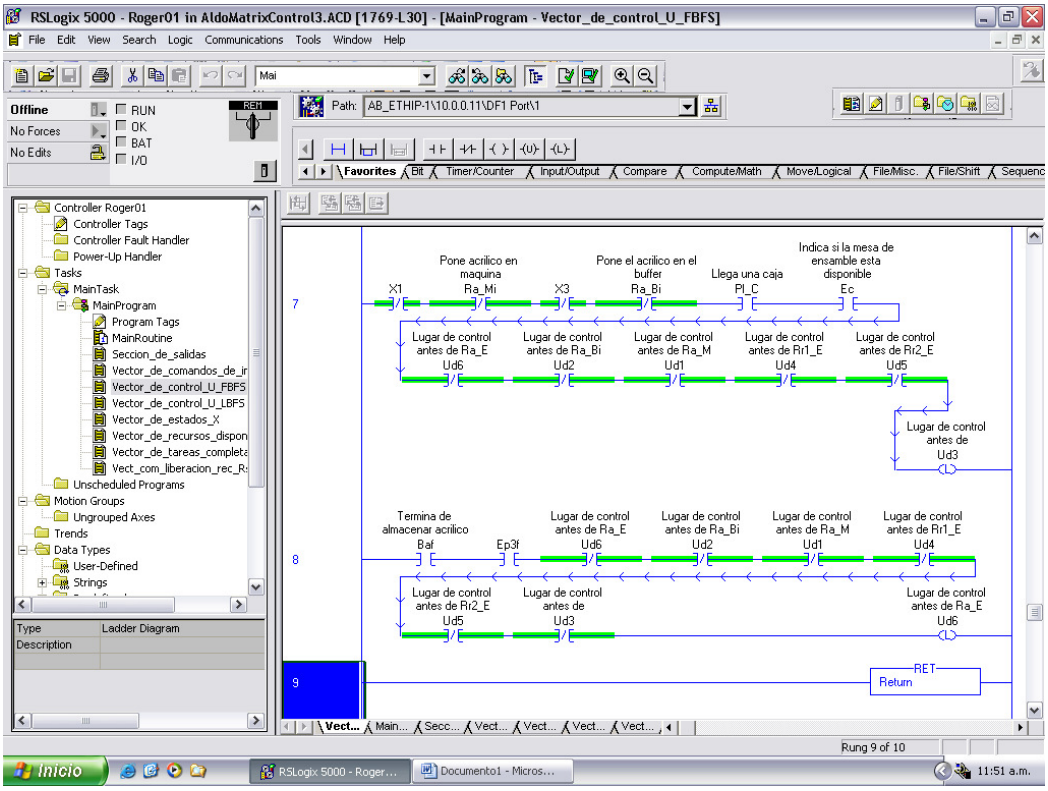

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_control_U_FBFS]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type
Ladder Diagram
Description
En cada ciclo se apagan todos los lugares de control y de acuerdo al estado del sistema se asigna un solo token
Lugar de control antes de Ra_M Ud1
Lugar de control antes de Ra_Bi Ud2
Lugar de control antes de Ud3
Lugar de control antes de Rr1_E Ud4
Lugar de control antes de Rr2_E Ud5
Lugar de control antes de Ra_E Ud6
Termina de almacenar acrilico Baf
deja de maquinar Maf
Llega el acrilico a la entrada PI_A
Ep3f
Vect... Main... Secc... Vect... Vect... Vect... Vect...
Ready
Rung 0 of 10
APP
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:49 a.m.

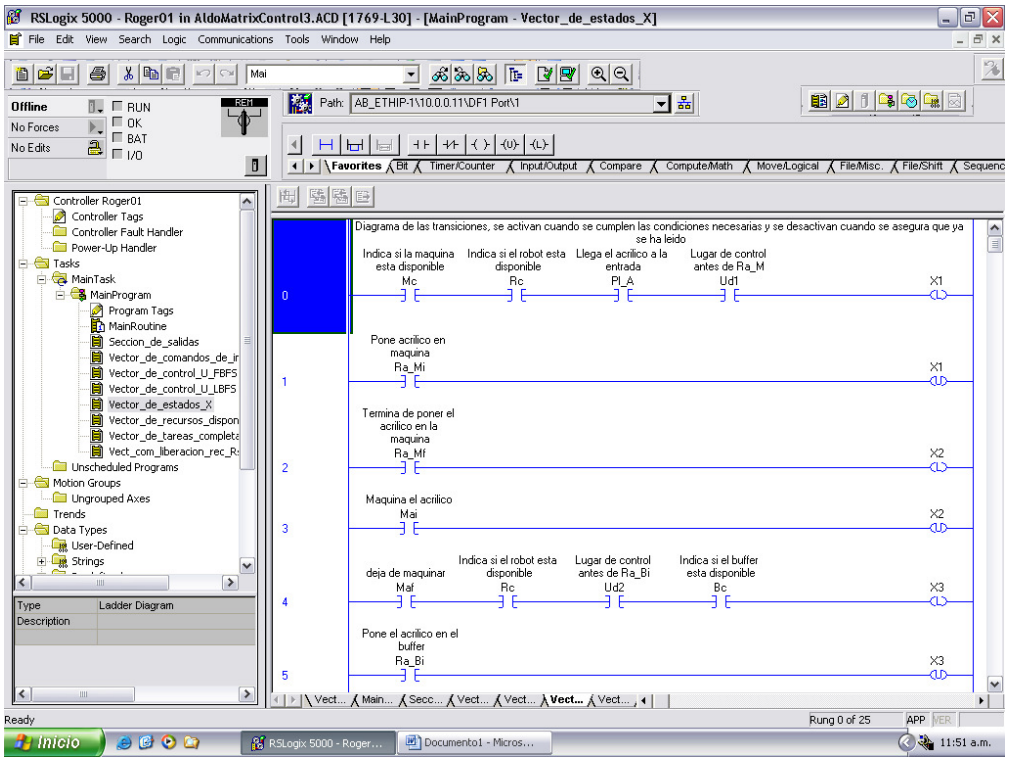

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_control_U_FBFS]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
Termina de almacenar acrilico Baf
deja de maquinar Maf
Llega el acrilico a la entrada Pl_A
Ep3f
Lugar de control antes de Ra_E Ud6
Pone el acrilico en el ensamble Ra_Ei
X11
X1
Pone acrilico en maquina Ra_Mi
X7
X9
X5
Pone la rueda en el ensamble Rr1_Ei
Pone la rueda 2 en el ensamble Rr2_Ei
Pone la caja en el ensamble Rc_Ei
Indica si el buffer esta disponible Bc
Lugar de control antes de Ra_Bi Ud2
Ready
Rung 2 of 10
APP
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:50 a.m.

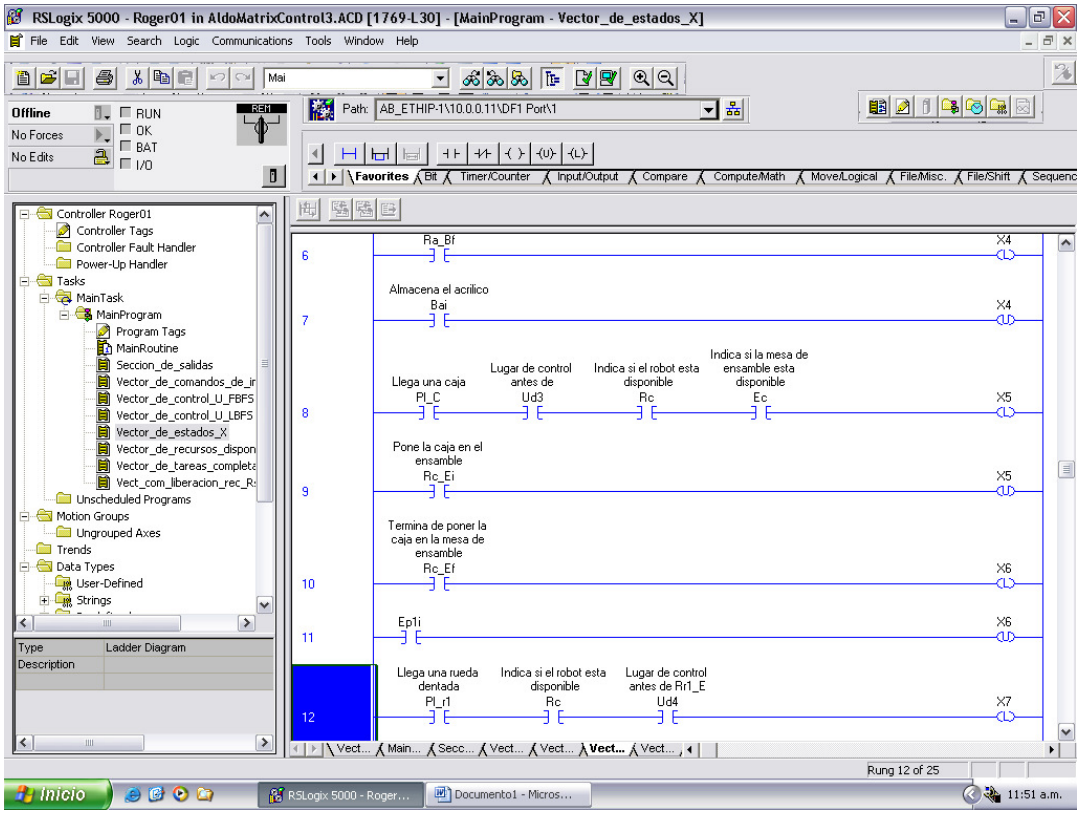

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_control_U_FBFS]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
Pone el acrilico en el ensamble Ra_Ei
X11
X1
Pone acrilico en maquina Ra_Mi
X9
X5
X7
Pone la rueda en el ensamble Rr1_Ei
Pone la rueda 2 en el ensamble Rr2_Ei
Pone la caja en el ensamble Rc_Ei
deja de maquinar Maf
Indica si el buffer esta disponible Bc
Lugar de control antes de Ra_E Ud6
Lugar de control antes de Ra_Bi Ud2
Lugar de control antes de Ra_M Ud1
Lugar de control antes de Rr1_E Ud4
Lugar de control antes de Rr2_E Ud5
Lugar de control antes de Ud3
Pone el acrilico en el ensamble Ra_Ei
Pone acrilico en maquina Ra_Mi
Pone la rueda en el ensamble Rr1_Ei
X11
X1
X7
X9
Pone la rueda 2 en el ensamble Rr2_Ei
X3
Pone el acrilico en el buffer Ra_Bi
X5
Pone la caja en el ensamble Rc_Ei
Llega el acrilico a la entrada Pl_A
Rung 3 of 10
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:50 a.m.

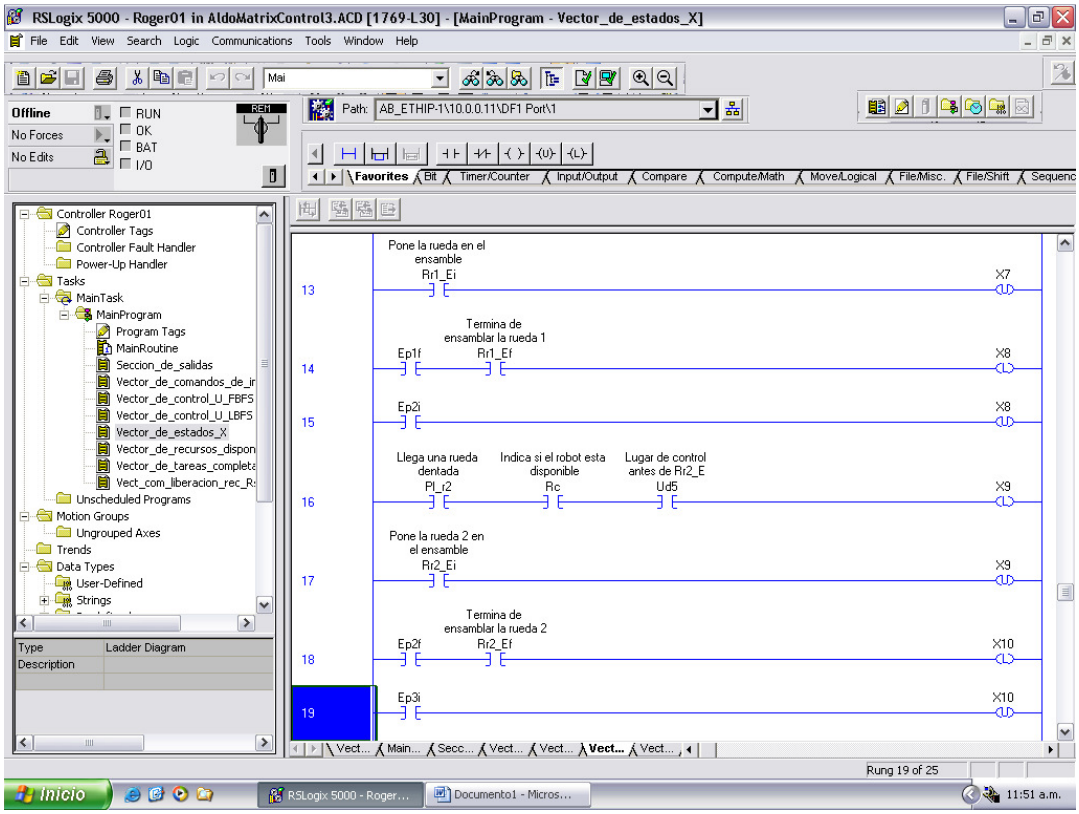

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_control_U_FBFS]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type
Ladder Diagram
Description
Pone el acrilico en el ensamble
Ra_Ei
Pone acrilico en maquina
Ra_Mi
Pone la rueda en el ensamble
Rr1_Ei
Pone la rueda 2 en el ensamble
Rr2_Ei
Pone el acrilico en el buffer
Ra_Bi
Pone la caja en el ensamble
Rc_Ei
Llega el acrilico a la entrada
Pl_A
Indica si la maquina esta disponible
Mc
Lugar de control antes de Ra_E
Ud6
Lugar de control antes de Ra_Bi
Ud2
Lugar de control antes de Ra_M
Ud1
Llega una rueda dentada
Pl_r1
Maf
Ep1f
Lugar de control antes de Rr1_E
Ud4
Pl_r2
Ep2f
Rung 4 of 10
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:50 a.m.

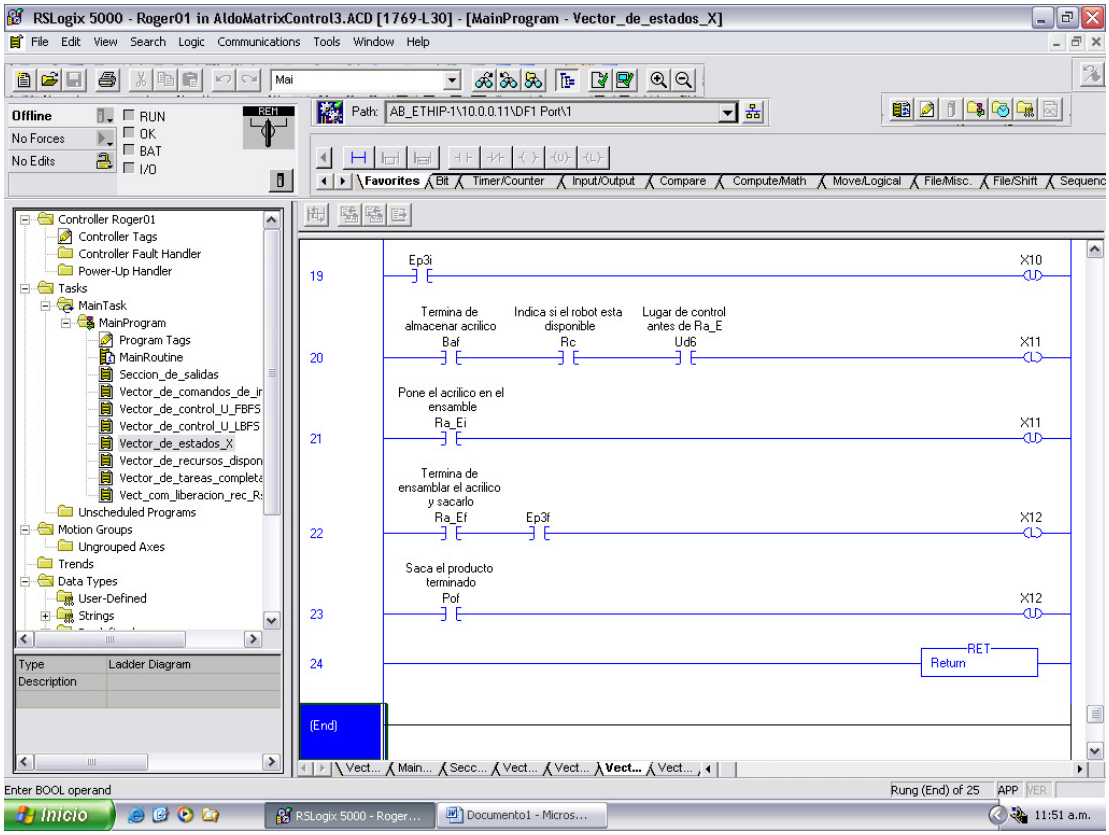

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_control_U_FBFS]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type
Ladder Diagram
Description
Pone acrilico en maquina
Ra_Mi
Pone el acrilico en el buffer
Ra_Bi
deja de maquinar
Maf
Llega una rueda dentada
Pl_r2
Ep2f
Lugar de control antes de Ra_E
Ud6
Lugar de control antes de Ra_Bi
Ud2
Lugar de control antes de Ra_M
Ud1
Lugar de control antes de Rr1_E
Ud4
Lugar de control antes de Rr2_E
Ud5
Llega una caja
Pl_C
Indica si la mesa de ensamble esta disponible
Ec
Lugar de control antes de
Ud3
Termina de almacenar acrilico
Baf
Ep3f
Rung 6 of 10
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:50 a.m.


RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_control_U_FBFS]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
Pone acrilico en maquina
Ra_Mi
Pone el acrilico en el buffer
Ra_Bi
Llega una caja
Pl_C
Indica si la mesa de ensamble esta disponible
Ec
X1
X3
7
Lugar de control antes de Ra_E Ud6
Lugar de control antes de Ra_Bi Ud2
Lugar de control antes de Ra_M Ud1
Lugar de control antes de Rr1_E Ud4
Lugar de control antes de Rr2_E Ud5
Lugar de control antes de Ud3
Termina de almacenar acrilico
Bal
Ep3f
8
Lugar de control antes de Rr2_E Ud5
Lugar de control antes de Ud3
Lugar de control antes de Ra_E Ud6
9
RET
Return
Vect... Main... Secc... Vect... Vect... Vect... Vect...
Rung 9 of 10
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:51 a.m.

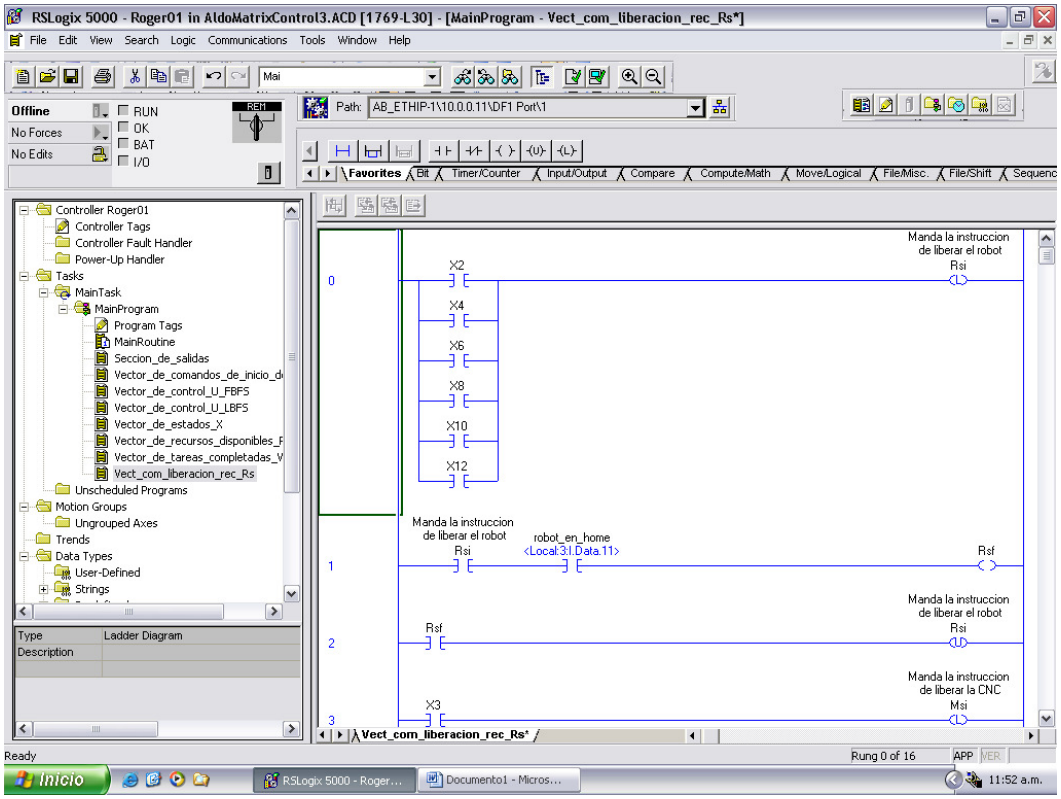

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_estados_X]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
Diagrama de las transiciones, se activan cuando se cumplen las condiciones necesarias y se desactivan cuando se asegura que ya se ha leido
Indica si la maquina esta disponible
Mc
Indica si el robot esta disponible
Rc
Llega el acrilico a la entrada
Pl_A
Lugar de control antes de Ra_M
Ud1
X1
0
Pone acrilico en maquina
Ra_Mi
1
Termina de poner el acrilico en la maquina
Ra_Mf
X2
2
Maquina el acrilico
Mai
3
deja de maquinar
Maf
Indica si el robot esta disponible
Rc
Lugar de control antes de Ra_Bi
Ud2
Indica si el buffer esta disponible
Bc
X3
4
Pone el acrilico en el buffer
Ra_Bi
5
Vect... Main... Secc... Vect... Vect... Vect... Vect...
Ready
Rung 0 of 25
APP
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:51 a.m.

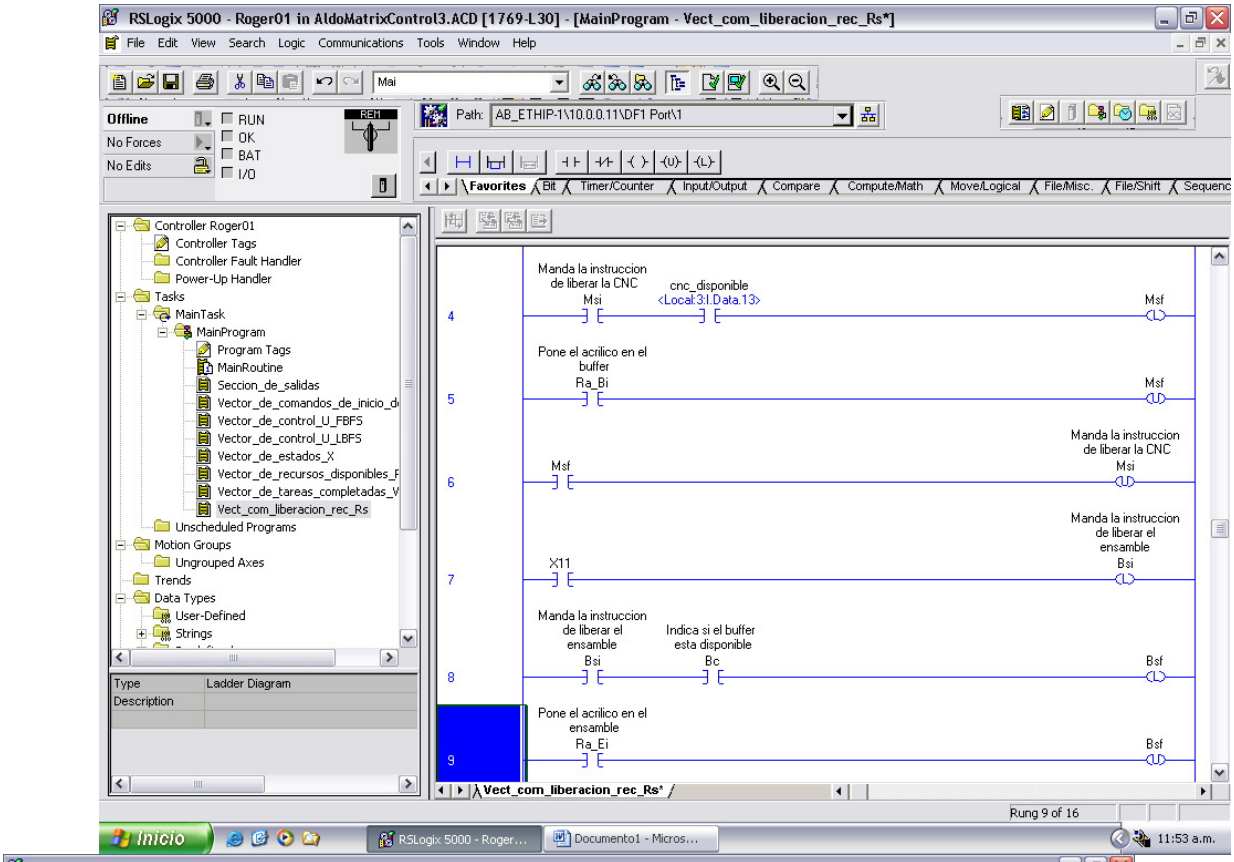

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_estados_X]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
Ra_Bf
X4
6
Almacena el acrilico
Bai
X4
7
Llega una caja
Pl_C
Lugar de control antes de
Ud3
Indica si el robot esta disponible
Rc
Indica si la mesa de ensamble esta disponible
Ec
X5
8
Pone la caja en el ensamble
Rc_Ei
X5
9
Termina de poner la caja en la mesa de ensamble
Rc_Ef
X6
10
Ep1i
X6
11
Llega una rueda dentada
Pl_r1
Indica si el robot esta disponible
Rc
Lugar de control antes de Rr1_E
Ud4
X7
12
Vect... Main... Secc... Vect... Vect... Vect... Vect...
Rung 12 of 25
inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:51 a.m.

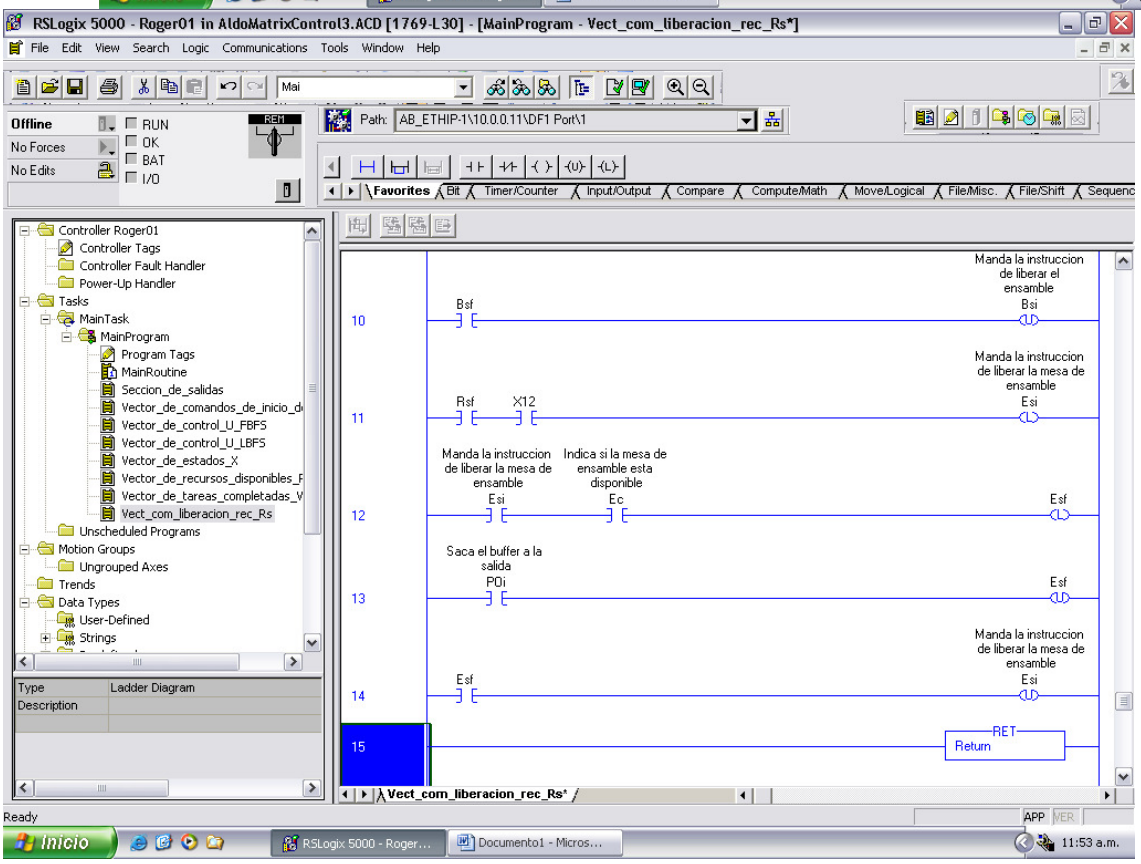

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_estados_X]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_ir
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_dispon
Vector_de_tareas_completa
Vect_com_liberacion_rec_R
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
Pone la rueda en el ensamble
Rr1_Ei
X7
13
Termina de ensamblar la rueda 1
Ep1f
Rr1_Ef
X8
14
Ep2i
X8
15
Llega una rueda dentada
Pl_r2
Indica si el robot esta disponible
Rc
Lugar de control antes de Rr2_E
Ud5
X9
16
Pone la rueda 2 en el ensamble
Rr2_Ei
X9
17
Termina de ensamblar la rueda 2
Ep2f
Rr2_Ef
X10
18
Ep3i
X10
19
Vect... Main... Secc... Vect... Vect... Vect... Vect...
Rung 19 of 25
inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:51 a.m.

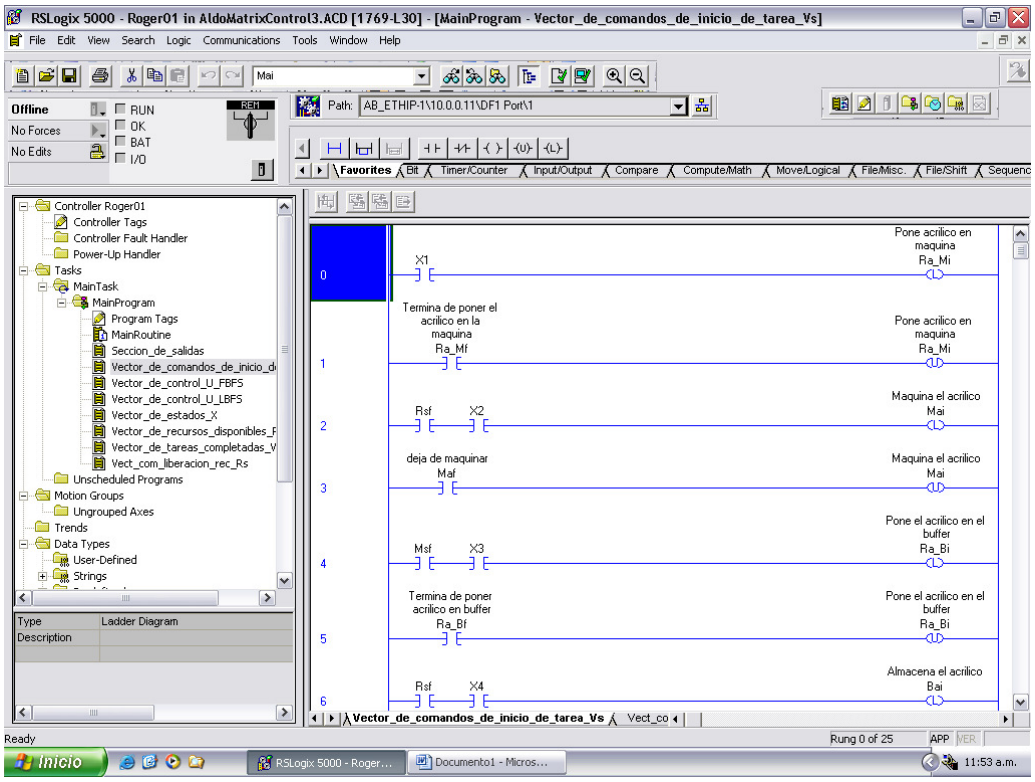


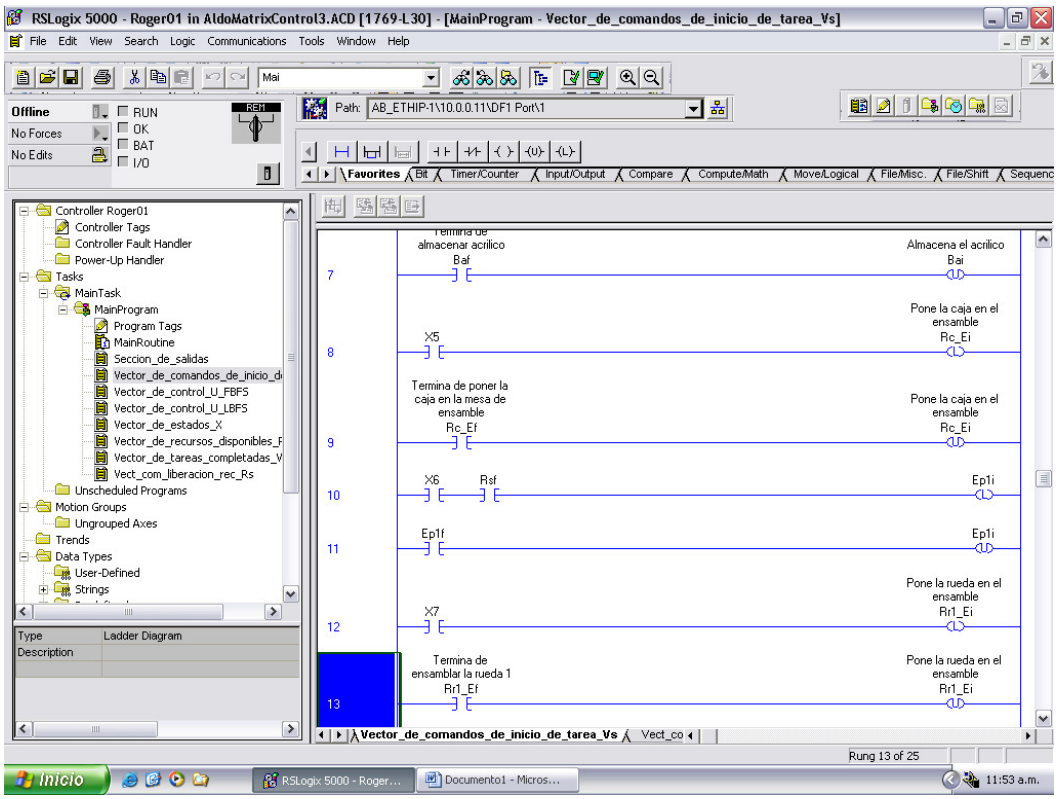
Liberación de recursos

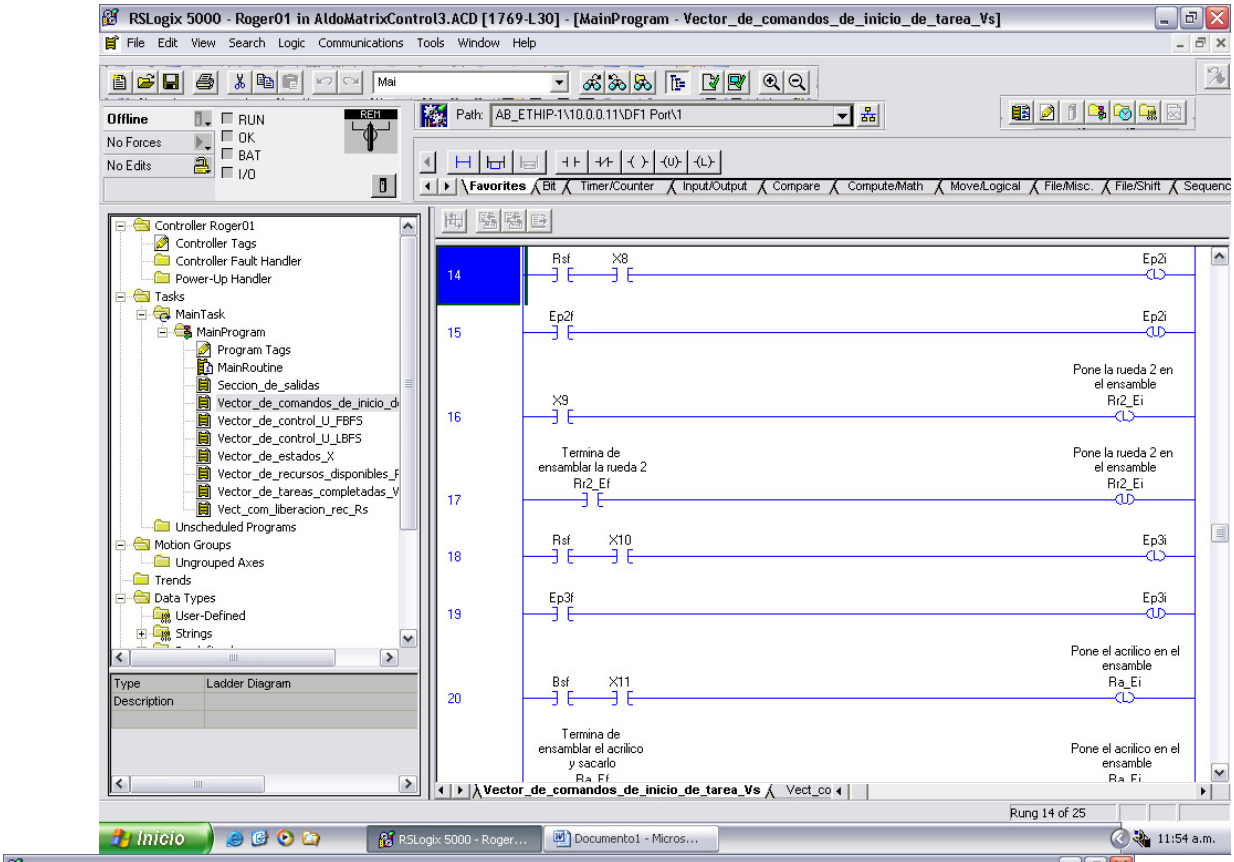


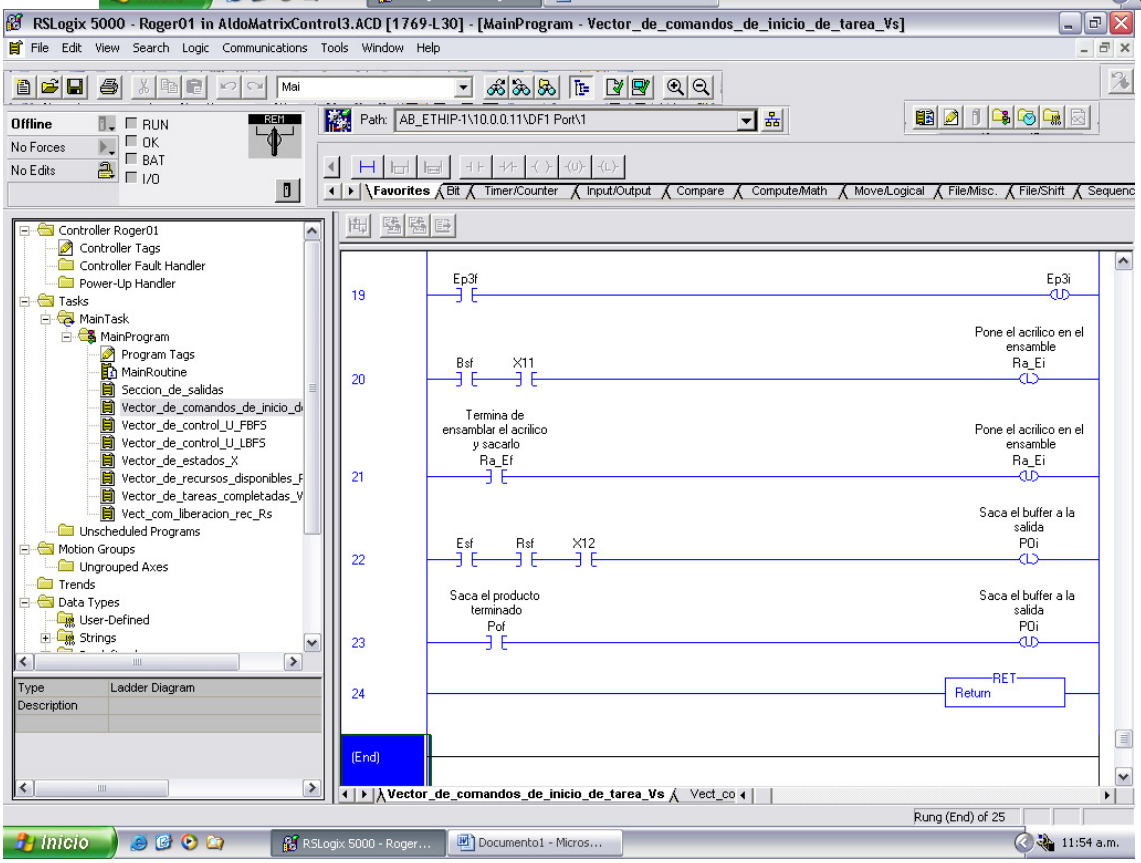

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vect_com_liberacion_rec_Rs*]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_inicio_d
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_disponibles_F
Vector_de_tareas_completadas_V
Vect_com_liberacion_rec_Rs
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
Manda la instruccion de liberar la CNC
Msi
cnc_disponible
<Local:3:I.Data.13>
Msf
Pone el acrilico en el buffer
Ra_Bi
Manda la instruccion de liberar el ensamble
Bsi
X11
Indica si el buffer esta disponible
Bc
Bsf
Pone el acrilico en el ensamble
Ra_Ei
Vect_com_liberacion_rec_Rs*
Rung 9 of 16
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:53 a.m.


RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vect_com_liberacion_rec_Rs*]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_inicio_d
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_disponibles_F
Vector_de_tareas_completadas_V
Vect_com_liberacion_rec_Rs
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
Manda la instruccion de liberar el ensamble
Bsi
Bsf
Manda la instruccion de liberar la mesa de ensamble
Esi
Rsf
X12
Manda la instruccion de liberar la mesa de ensamble
Indica si la mesa de ensamble esta disponible
Ec
Esf
Saca el buffer a la salida
POi
RET
Return
Vect_com_liberacion_rec_Rs*
Ready
APP
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:53 a.m.

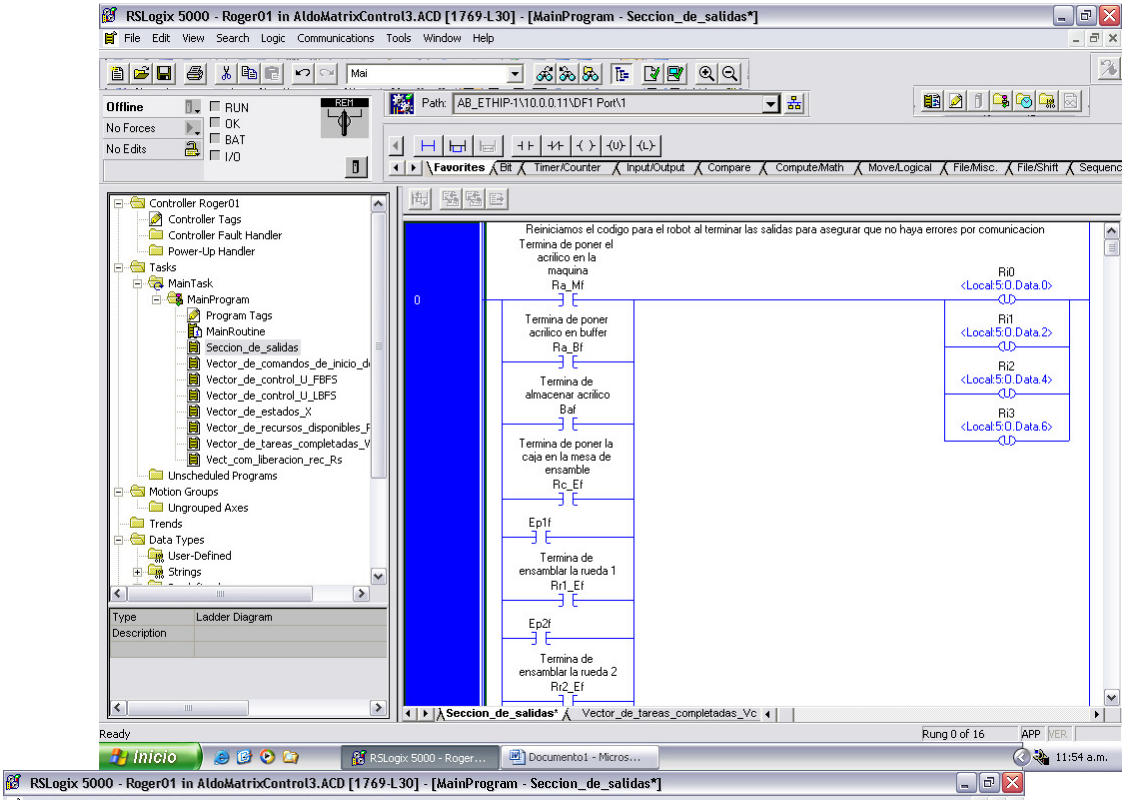

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_comandos_de_inicio_de_tarea_Vs]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_inicio_de
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_disponibles_F
Vector_de_tareas_completadas_V
Vect_com_liberacion_rec_Rs
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
0 X1 Pone acrilico en maquina Ra_Mi
1 Termina de poner el acrilico en la maquina Ra_Mf Pone acrilico en maquina Ra_Mi
2 Rsf X2 Maquina el acrilico Mai
3 deja de maquinar Maf Maquina el acrilico Mai
4 Msf X3 Pone el acrilico en el buffer Ra_Bi
5 Termina de poner acrilico en buffer Ra_Bf Pone el acrilico en el buffer Ra_Bi
6 Rsf X4 Almacena el acrilico Bai
Vector_de_comandos_de_inicio_de_tarea_Vs
Ready
Rung 0 of 25
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:53 a.m.

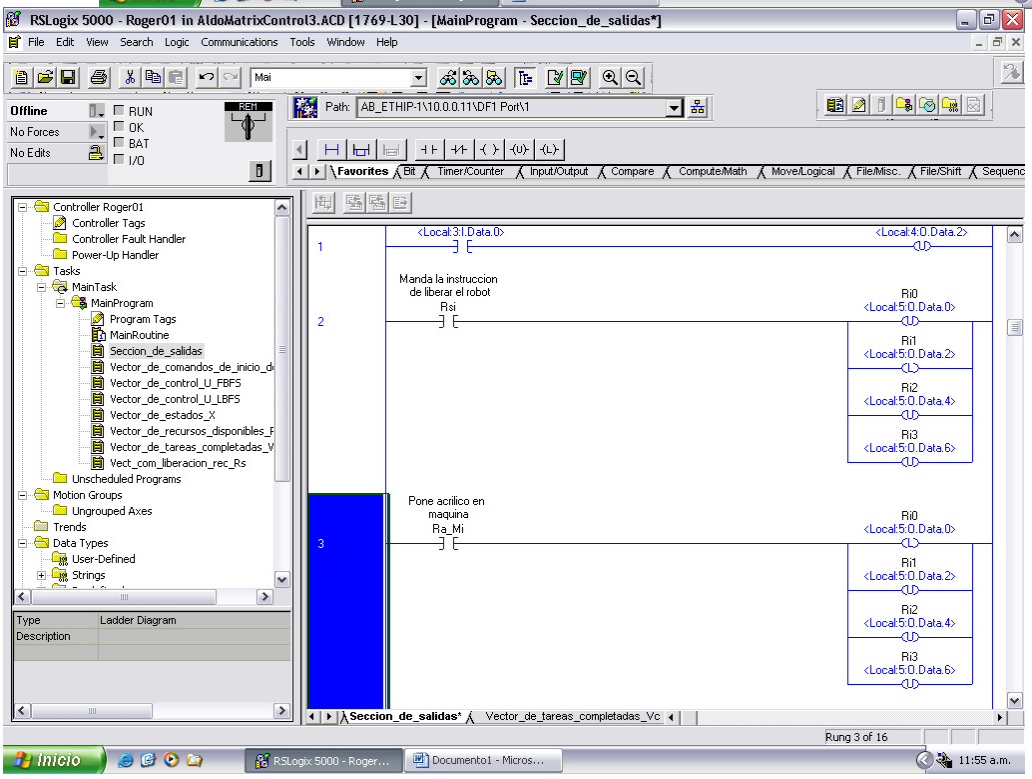

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_comandos_de_inicio_de_tarea_Vs]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_inicio_de
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_disponibles_F
Vector_de_tareas_completadas_V
Vect_com_liberacion_rec_Rs
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
7 Termina de almacenar acrilico Baf Almacena el acrilico Bai
8 X5 Pone la caja en el ensamble Rc_Ei
9 Termina de poner la caja en la mesa de ensamble Rc_Ef Pone la caja en el ensamble Rc_Ei
10 X6 Rsf Ep1i
11 Ep1f Ep1i
12 X7 Pone la rueda en el ensamble Rr1_Ei
13 Termina de ensamblar la rueda 1 Rr1_Ef Pone la rueda en el ensamble Rr1_Ei
Vector_de_comandos_de_inicio_de_tarea_Vs
Rung 13 of 25
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:53 a.m.

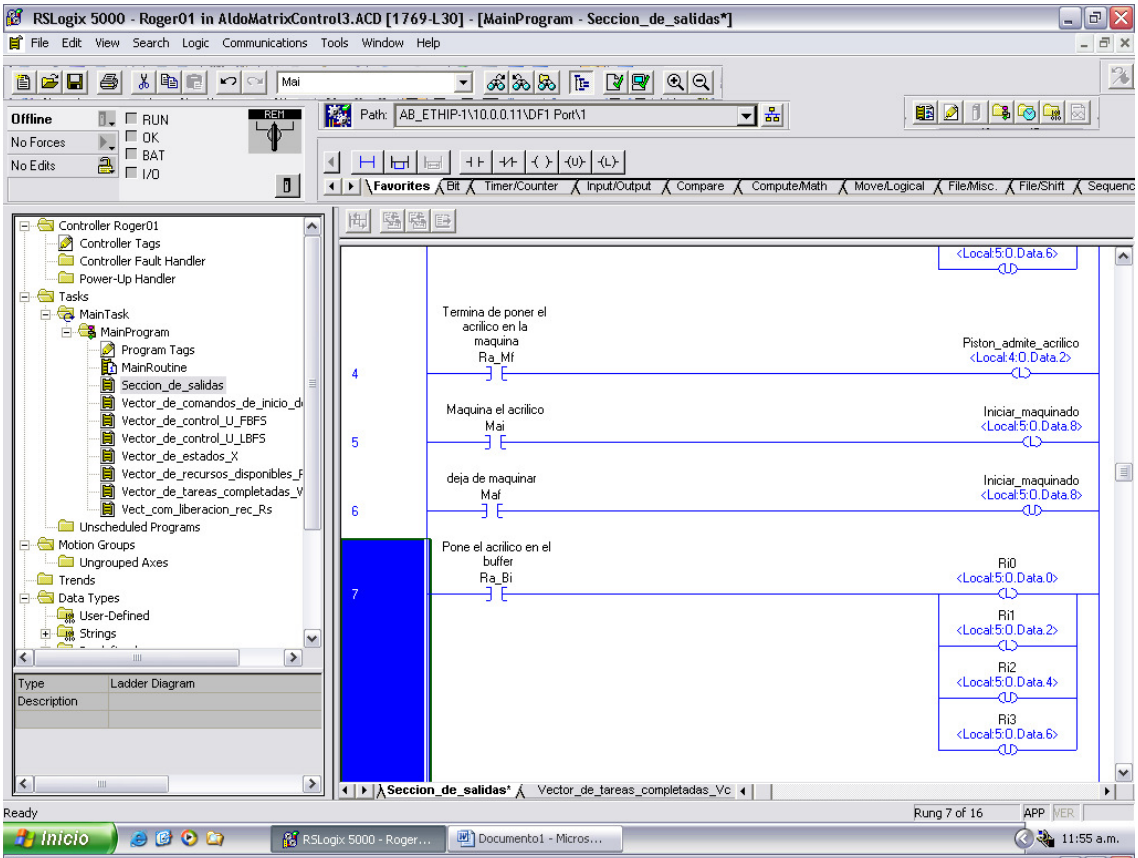

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_comandos_de_inicio_de_tarea_Vs]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_inicio_d
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_disponibles_F
Vector_de_tareas_completadas_V
Vect_com_liberacion_rec_Rs
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
14 Rsf X8 Ep2i
15 Ep2f Ep2i
16 X9 Pone la rueda 2 en el ensamble Rr2_Ei
17 Termina de ensamblar la rueda 2 Rr2_Ef Pone la rueda 2 en el ensamble Rr2_Ei
18 Rsf X10 Ep3i
19 Ep3f Ep3i
20 Bsf X11 Pone el acrilico en el ensamble Ra_Ei
Termina de ensamblar el acrilico y sacarlo Ra_Ef Pone el acrilico en el ensamble
Vector_de_comandos_de_inicio_de_tarea_Vs Vect_co
Rung 14 of 25
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:54 a.m.

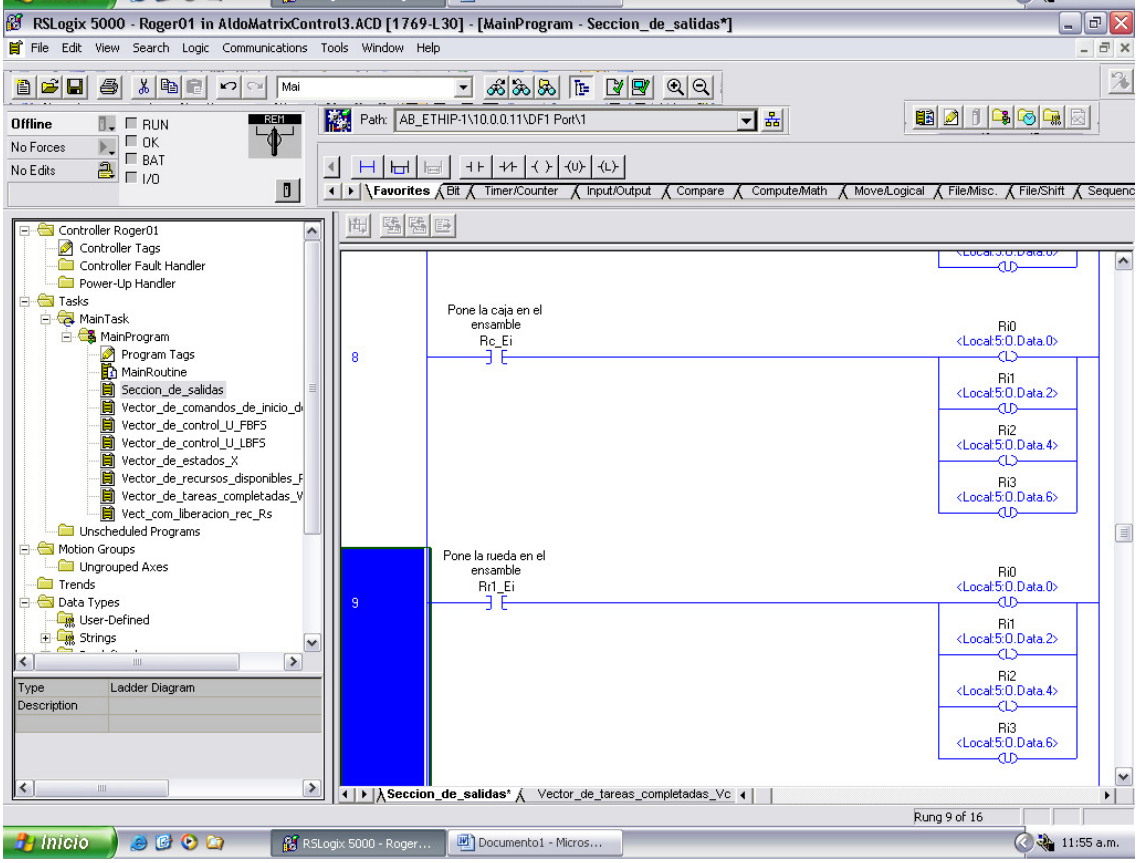

RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Vector_de_comandos_de_inicio_de_tarea_Vs]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_inicio_d
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_disponibles_F
Vector_de_tareas_completadas_V
Vect_com_liberacion_rec_Rs
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
19 Ep3f Ep3i
20 Bsf X11 Pone el acrilico en el ensamble Ra_Ei
Termina de ensamblar el acrilico y sacarlo Ra_Ef Pone el acrilico en el ensamble Ra_Ei
22 Esf Rsf X12 Saca el buffer a la salida POi
Saca el producto terminado Pof Saca el buffer a la salida POi
24 RET Return
(End)
Vector_de_comandos_de_inicio_de_tarea_Vs Vect_co
Rung (End) of 25
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:54 a.m.

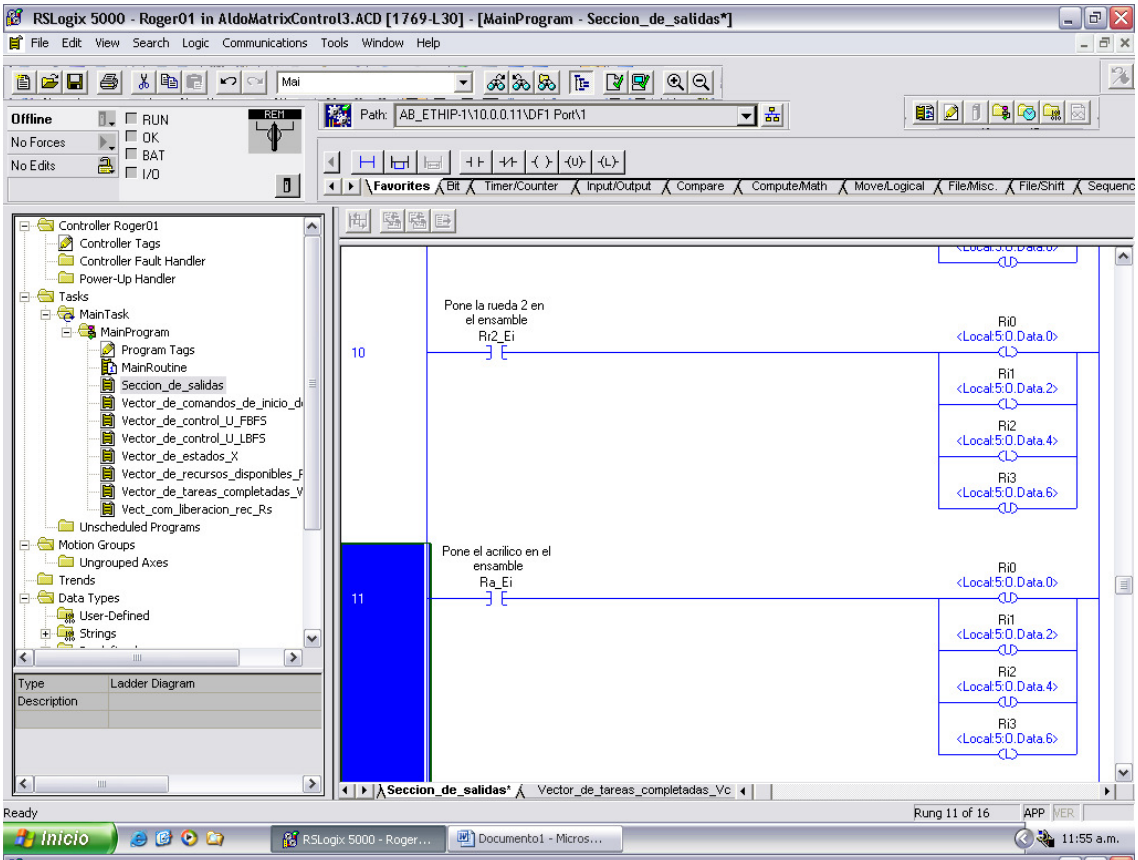
Seccion de salidas

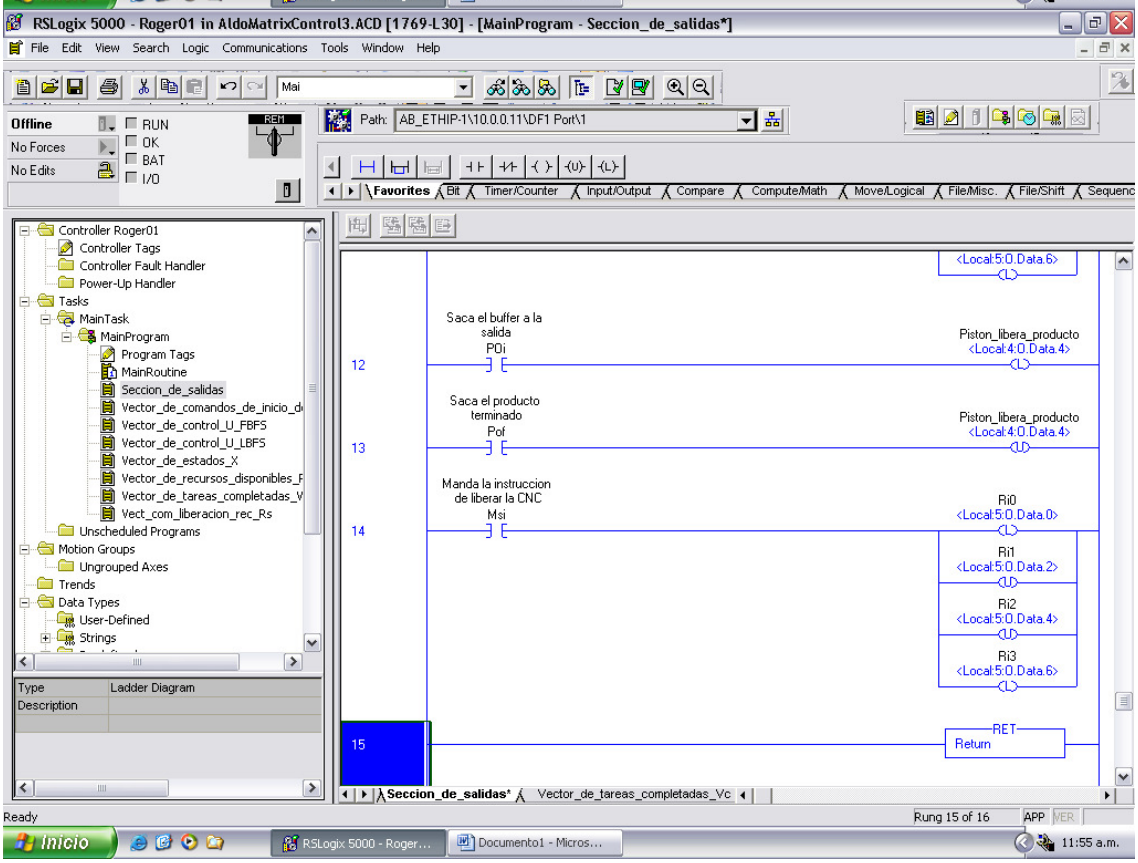





RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Seccion_de_salidas*]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_inicio_d
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_disponibles_F
Vector_de_tareas_completadas_V
Vect_com_liberacion_rec_Rs
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
Termina de poner el acrilico en la maquina
Ra_Mf
Piston_admite_acrilico
Maquina el acrilico
Mai
Iniciar_maquinado
deja de maquinar
Maf
Pone el acrilico en el buffer
Ra_Bi
Ri0
Ri1
Ri2
Ri3
Seccion_de_salidas* Vector_de_tareas_completadas_Vc
Ready
Rung 7 of 16
APP
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:55 a.m.


RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Seccion_de_salidas*]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_inicio_d
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_disponibles_F
Vector_de_tareas_completadas_V
Vect_com_liberacion_rec_Rs
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
Pone la caja en el ensamble
Rc_Ei
Ri0
Ri1
Ri2
Ri3
Pone la rueda en el ensamble
Rr1_Ei
Seccion_de_salidas* Vector_de_tareas_completadas_Vc
Rung 9 of 16
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:55 a.m.


RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Seccion_de_salidas*]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_inicio_d
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_disponibles_F
Vector_de_tareas_completadas_V
Vect_com_liberacion_rec_Rs
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
Pone la rueda 2 en el ensamble
Rr2_Ei
10
Ri0 <Local:5:O.Data.0>
Ri1 <Local:5:O.Data.2>
Ri2 <Local:5:O.Data.4>
Ri3 <Local:5:O.Data.6>
Pone el acrilico en el ensamble
Ra_Ei
11
Seccion_de_salidas* Vector_de_tareas_completadas_Vc
Ready
Rung 11 of 16
APP
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:55 a.m.


RSLogix 5000 - Roger01 in AldoMatrixControl3.ACD [1769-L30] - [MainProgram - Seccion_de_salidas*]
File Edit View Search Logic Communications Tools Window Help
Offline
No Forces
No Edits
RUN
OK
BAT
I/O
Path: AB_ETHIP-1\10.0.0.11\DF1 Port\1
Favorites Bit Timer/Counter Input/Output Compare Compute/Math Move/Logical File/Misc. File/Shift Sequenc
Controller Roger01
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
Program Tags
MainRoutine
Seccion_de_salidas
Vector_de_comandos_de_inicio_d
Vector_de_control_U_FBFS
Vector_de_control_U_LBFS
Vector_de_estados_X
Vector_de_recursos_disponibles_F
Vector_de_tareas_completadas_V
Vect_com_liberacion_rec_Rs
Unscheduled Programs
Motion Groups
Ungrouped Axes
Trends
Data Types
User-Defined
Strings
Type Ladder Diagram
Description
<Local:5:O.Data.6>
Saca el buffer a la salida
POi
12
Piston_libera_producto <Local:4:O.Data.4>
Saca el producto terminado
Pof
13
Piston_libera_producto <Local:4:O.Data.4>
Manda la instruccion de liberar la CNC
Msi
14
Ri0 <Local:5:O.Data.0>
Ri1 <Local:5:O.Data.2>
Ri2 <Local:5:O.Data.4>
Ri3 <Local:5:O.Data.6>
15
RET
Return
Seccion_de_salidas* Vector_de_tareas_completadas_Vc
Ready
Rung 15 of 16
APP
Inicio
RSLogix 5000 - Roger...
Documento1 - Micros...
11:55 a.m.

15 APENDICE 2: PROGRAMA ROBOT PEGASUS

Programa del Robot

```
//Programa para ensamble por aproximacion basada en matrices
Entradas = SENSOR 3, 4
//Pmove <home>
Release
Writeo Output_5,Off  // abrir piston
Writeo Output_3,On   //abrir prensa cnc
Writeo Output_6,Off  // no correr CNC
Writeo Output_4,Off  // activar piston de cajas

Writeo Output_1,Off    //reinicia fin de maq.
Writeo Output_10,Off    //reinicia fin de op. rob.

Writeo Output_11,On     //robot disponible
Writeo Output_12,On     //CNC disponible

Label Inicio
CNCR=INP(Input_7)         //leemos la instruccion para la CNC
Codigo=INP(Entradas)      //leemos instruccion para el robot
CNCDONE=INP(Input_16)

//Ma: Maquina el acrilico
If CNCR=1 Then
Writeo Output_12,Off  //Marca la maquina CNC como ocupada
Writeo Output_6,On  //Hecha a andar la CNC
If CNCDONE=1 Then
Writeo Output_6,Off  // Apaga la CNC
Writeo Output_1,On  //Indica el fin de la rutina
Endif
Endif

// Ir a home

IF Codigo=2 THEN
Writeo Output_11,Off
Pmove Point_5
Writeo Output_11,On

// Ra_M: Lleva el acrilico a la maquina

ELSE IF Codigo=1 THEN
Writeo Output_11,Off
Writeo Output_12,Off  //Marca la maquina CNC como ocupada
Pmove Point_4
Pmove Point_3  // punto 3 es una posicion de seguridad
Pmove Point_1  //punto 1 esta arriba del acrilico
Lmove Point_2  // punto 2 es para tomarel acrilico
Grasp
Lmove Point_1
Pmove Point_4 //va a una posicion segura antes de mover el carro
```

```
Pmove Point_5 //movemos al area de la maquina y el ensamble
Pmove Point_12 // Punto 12 posicion intermedia antes de colocar en cnc
Pmove Point_13 // punto 13 arriba de la cnc
Lmove Point_14 //punto 14 en la cnc
Release
Writeo Output_3,Off  //cerrar prensa cnc
Lmove Point_13
Pmove Point_12
Writeo Output_10,On  //Indica el fin de la rutina

// Ra_B: acrilico maquinado al  buffer
// Pmove Point_5 //movemos al area de la maquina y el ensamble


Else If Codigo=3 THEN
Writeo Output_13,Off   //ocupa buffer
Writeo Output_11,Off
// toma el acrilico de la maquina
Pmove Point_12
Pmove Point_13
Lmove Point_14
Grasp
Writeo Output_3,On  //cerrar prensa cnc
Lmove Point_13
Pmove Point_12
Pmove Point_5

Pmove Point_6  //^punto 6 es el seguro para antes de poner en el buffer
Lmove Point_7  //punto 7 esta arriba del buffer
Release
Pmove Point_6
Writeo Output_10,On  //Indica el fin de la rutina


//Rc_E: Poner la caja en la mesa de ensamble

Else If Codigo=5 THEN
Writeo Output_4,On   //activa el piston de la caja
Writeo Output_14,Off  //ocupa el ensamble
Writeo Output_11,Off //ocupa el robot
Pmove Point_5
Pmove Point_8  // punto 8 se encuentra arriba de la caja de entrada
Lmove Point_9  // punto 9 es para tomar la caja
Grasp
Lmove Point_8
Writeo Output_4,Off   //deja caer otra caja
Pmove Point_10 //punto 10 esta arriba del ensamble
Lmove Point_11 //punto 11 es para colocar la caja en el ensamble
Release
Writeo Output_5,On   //activamos piston de la mesa de ensamble
Lmove Point_10
Writeo Output_10,On  //Indica el fin de la rutina
If CNCDONE=1 Then
Writeo Output_6,Off  // Apaga la CNC
Writeo Output_1,On  //Indica el fin de la rutina
```

```
Endif

// Rr1_E
Else If Codigo=6 THEN
Writeo Output_11,Off //ocupa el robot
Pmove Point_15  //punto 15 arriba de la llanta
Lmove Point_16  //punto 16 es para tomar la rueda
Grasp
Lmove Point_15

Pmove Point_18  // punto 18 arriba del ensamble
Lmove Point_17   //punto 17 para poner la rueda
Release
Lmove Point_18
Writeo Output_10,On  //Indica el fin de la rutina
If CNCDONE=1 Then
Writeo Output_6,Off  // Apaga la CNC
Writeo Output_1,On  //Indica el fin de la rutina
endif
//Rr2_E
Else If Codigo=7 THEN
Writeo Output_11,Off //ocupa el robot
Pmove Point_15  //punto 15 arriba de la llanta
Lmove Point_16  //punto 16 es para tomar la rueda
Grasp
Lmove Point_15

Pmove Point_35  //punto 35 arriba de r2
Lmove Point_36  // punto 36 pone r2
Lmove Point_37   //^Punto 37 se acerca al otro engrane
Release
Lmove Point_35
Writeo Output_10,On  //Indica el fin de la rutina
If CNCDONE=1 Then
Writeo Output_6,Off  // Apaga la CNC
Writeo Output_1,On  //Indica el fin de la rutina
Endif
//Ra_E: Poner el acrilico en el ensamble y llevar el ensamble
//a la salida
Else If Codigo=8 THEN
Writeo Output_11,Off //ocupa el robot
Pmove Point_5
Pmove Point_21 //punto 21 esta arriba del buffer
Pmove Point_22// punto 22 toma el acrilico del buffer
Grasp
Lmove Point_21

Pmove Point_5
Pmove Point_23//pumto 23 arriba del ensamble
Lmove Point_24// punto 24 para poner el acrilico en el ensamble
Release

Lmove Point_27 //punto 27 para tomar el ensamble
Grasp
Writeo Output_5,Off  //apagar el piston del ensamble
Lmove Point_26 //punto 26 arriba de tomar el ensamble
```

```
Pmove Point_5
Pmove Point_25 //punto 25 es el neutral del lado de las entradas
If CNCDONE=1 Then
Writeo Output_6,Off  // Apaga la CNC
Writeo Output_1,On  //Indica el fin de la rutina
endif
Pmove Point_30 //punto 30 da la vuelta para colocar la salida
Pmove Point_28 // punto 28 es arriba de la salida
Lmove Point_29 // punto 29 pone la caja a la salida
Release
Lmove Point_28
Pmove Point_30

If CNCDONE=1 Then
Writeo Output_6,Off  // Apaga la CNC
Writeo Output_1,On  //Indica el fin de la rutina
Endif
Writeo Output_10,On  //Indica el fin de la rutina
// Ms (Liberar CNC):
Else If Codigo=9 THEN
Writeo Output_12,On  //Indica que la cnc esta libre

Endif
Writeo Output_2,Off    //reinicia fin de ens.
Writeo Output_1,Off    //reinicia fin de maq.
Writeo Output_10,Off    //reinicia fin de op. rob.

Goto Inicio
```