Anexos

Contenido

Anexo A: Diseño de una familia de productos

Establecer una jerarquía es muy útil al momento de definir la familia de productos que se pueden fabricar para analizar qué variantes se van a utilizar y qué posibles modificaciones puede requerir el sistema, para esto ElMaraghy ha propuesto 8 niveles para realizar una jerarquía en donde se muestre la variedad de productos que se pueden obtener, estos niveles son [42]:

A.1 Características de componentes

Se deben analizar características comunes en cuanto a su geometría y sus variantes como las ranuras o agujeros y sus medidas. Toda esta información puede almacenarse mejor mediante un diseño que contenga los parámetros, restricciones, posibles variaciones y propiedades de los materiales.

A.2 Partes o componentes

Son todos los elementos que no se pueden descomponer porque perderían su utilidad, esto se refiere a ciertas piezas que si son separadas pierden sus atributos y no funcionarían de igual forma lo que requeriría un rediseño de todo el sistema.

A.3 Familias de componentes

Este nivel agrupa los productos de acuerdo a sus similitudes en cuanto a su geometría y a los procesos que requieren para su fabricación. El propósito de esto es poder reducir tiempo de diseño y aumentar la eficiencia de los procesos y de la asignación de procesos y flujos de material.

A.4 Módulos de producto y sub-ensamblajes

Aquí se pueden crear unidades independientes (módulos) donde se agrupen productos que cumplan con una especificación técnica en común. Es importante tener claros los rangos para no afectar la modularidad del sistema para que sea posible hacer las correctas combinaciones y personalización del producto. El sub-ensamblaje es la

relación estrecha que hay entre las partes, y pueden considerarse como una sola unidad estable que toma una misma dirección una vez haya sido montada o ensamblada.

A.5 Productos

Es la unión de los módulos y sub-ensamblajes que dan como resultado el producto que va a ser comercializado.

A.6 Familias de producto

En este nivel están los productos que tienen características, componentes y/o sub-ensamblajes iguales. La definición de este punto es la más importante ya que aquí la empresa tiene que cumplir con la demanda de la variedad de productos que exige el mercado. Además, se debe tener en cuenta las variantes de los productos que se adapten a los sistemas reconfigurados y ser coherente con los productos dentro de la familia, de tal forma que se pueda brindar la mayor variedad posible de productos con la menor variedad de métodos de fabricación aumentando así la competitividad.

A.7 Plataforma de productos

Es el conjunto de módulos, sub-ensamblajes, sus interfaces relacionadas e infraestructura que comparten características que sirven como eje para producir varios productos similares. El diseño de la plataforma es dinámico para que su planeación, gestión y actualización tenga una respuesta satisfactoria en la fabricación de productos variados.

Las plataformas, componentes y partes siempre son las mismas en la familia de productos.

A.8 Portafolio de productos

Es toda la oferta de productos que brinda la empresa, pueden haber productos muy diferentes unos de otros. Aquí se agrupan todas las plataformas. El portafolio es definido de acuerdo a los objetivos de la empresa y la plaza en la que desea moverse.

La selección de los productos a fabricar no debe realizarse de forma individual, si bien la empresa comienza planeando un producto para vender se debe pensar en toda la gama de productos similares o derivados que pueden fabricarse con base en el

producto inicial, y que se adapten al sistema de manufactura con el que se cuenta [42]. De esta forma se promueve la modularidad y la automatización integral del sistema de manufactura para obtener un Sistema de Manufactura Reconfigurable.

A.9 Principios para el diseño de productos

Koren propone los siguientes principios de diseño de productos personalizados y reconfigurables [42][17]:

- "*La funcionalidad esencial debe permanecer invariante en todos los productos de una familia: Las características y funcionalidades esenciales definen a la familia de productos, y si se pierden el producto es diferente."*

- *"Se deben minimizar las opciones que no aportan valor en las diferentes variantes de una familia de productos: las variaciones que no agregan valor en el producto no amplían el portafolio de productos, y es realmente un truco de publicidad para atraer clientes. Este tipo de variaciones en los productos deben traer beneficios para el cliente y no confundirlo, que es lo que sucedería con esas variaciones que no le dan valor, lo cual puede disminuir las ventas."*

- *"Las características claves deben ser incorporadas en la arquitectura del producto: características básicas de diseño que generan la rentabilidad personalizada de la producción, y los productos reconfigurables deben estar en todos los miembros de la familia de productos."*

Anexo B: Descripción de la máquina CNC a reconfigurar

Existe una gran variedad de máquinas CNC y centros de mecanizado especializados para realizar diferentes trabajos sobre madera y demás materiales pero estos son costosos. El objetivo de la reconfiguración es adaptar la funcionalidad de la máquina para reducir los tiempos de mecanizado, aumentar la producción y la variedad de productos, mediante modificaciones en la punta adaptándola para la ejecución de tres procesos: torneado, taladrado y fresado, con el fin de reducir costos. Mediante el diseño de un modelo de control distribuido se busca la optimización de una máquina CNC partiendo de que esta se desplace en los ejes de coordenadas X, Y y Z; y la adaptación del elemento final de control que actúa sobre la pieza de trabajo.

Se ha podido observar que la madera es un buen material para los procesos de mecanizado, y por esto se escogió como el material para la fabricación de los productos definidos en el caso de estudio. Las operaciones definidas para la máquina (taladrado, fresado y torneado) permiten mayor precisión de las puntas cuando se trabaja con madera, todo esto para adquirir mayor robustez y el éxito en el desarrollo de un Sistema de Manufactura Reconfigurable.

B.1 P&ID

Para tener una idea de cómo sería la máquina caso de estudio se presenta el diagrama P&ID de la RMT en la figura B.1.

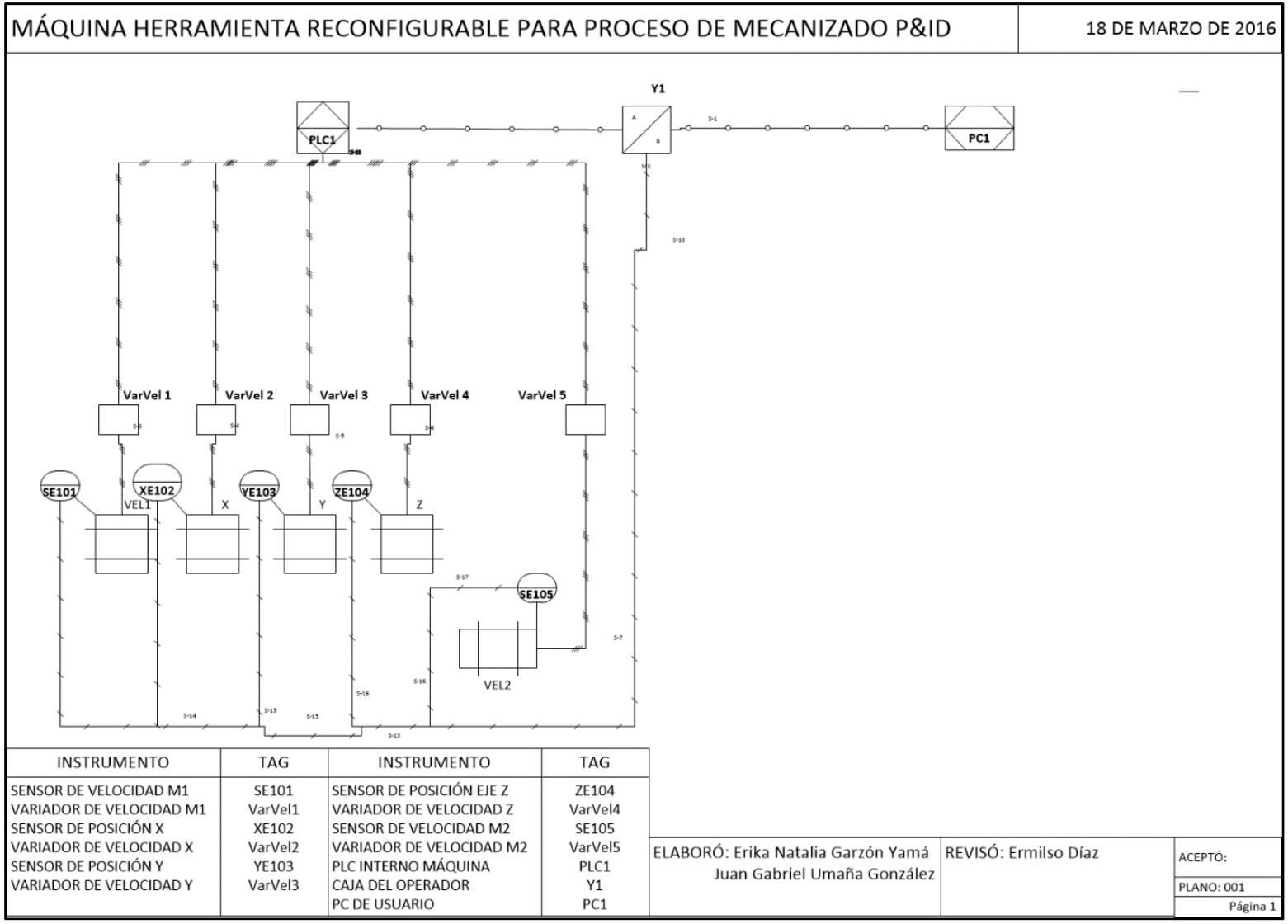


Figura B.1: Diagrama P&ID de la RMT.

B.2 Torneado

En la tabla B.1 se presentan las diferentes actividades que se pueden realizar en el torneado.

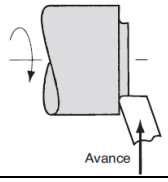
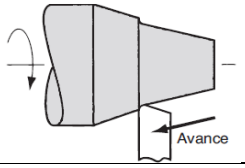
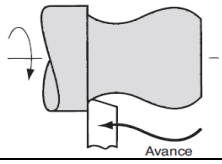
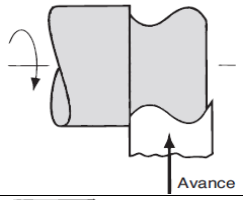
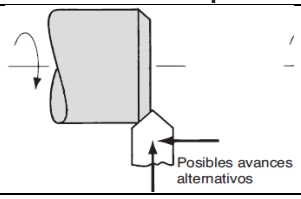
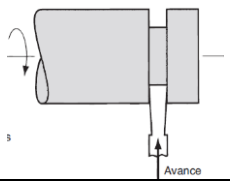
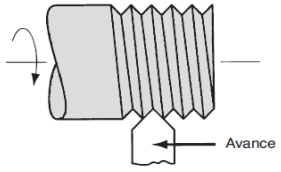
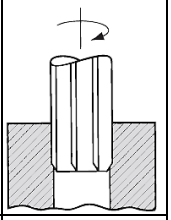
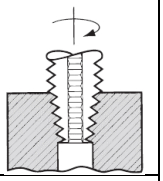
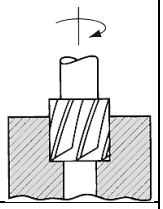
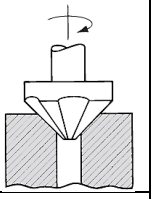
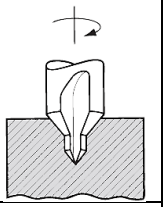
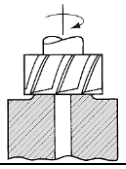
Operación	Descripción	Vista
Careado	La herramienta se alimenta radialmente sobre el extremo rotatorio para crear una superficie plana.	
Torneado ahusado o cónico	La herramienta avanza en cierto ángulo creando una forma cónica.	
Torneado de contornos	La herramienta avanza siguiendo un contorno diferente a la línea recta, creando así una forma contorneada.	
Torneado de formas	La herramienta tiene una forma que imparte al trabajo y se hunde radialmente dentro del trabajo.	
Achaflanado	El borde cortante de la herramienta se usa para cortar un ángulo en la esquina del cilindro y forma lo que se llama un "chaflán".	
Tronzado	La herramienta avanza radialmente dentro del trabajo en rotación, en algún punto a lo largo de su longitud, para trozar el extremo de la pieza.	
Roscado	Una herramienta puntiaguda avanza linealmente a través de la superficie externa de la pieza en rotación y en dirección paralela al eje de rotación, a una velocidad de avance suficiente para crear cuerdas roscadas en el cilindro.	

Tabla B.1: operaciones relacionadas con torneado [37].

B.3 Taladrado

En la tabla B.2 se presentan las diferentes actividades que se pueden realizar en el taladrado.

Operación	Descripción	Vista
Escariado	Un escariador aumenta el tamaño del agujero ligeramente, mejora la tolerancia en su diámetro y el acabado superficial.	
Roscado interior	Un machuelo corta una rosca interior en el agujero ya existente.	
Abocardado	Se va realizando un agujero de forma gradual donde un diámetro más grande sigue a uno más pequeño.	
Avellanado	Igual al abocardado pero realiza el agujero en forma de cono.	
Centrado	Realiza el agujero inicial que mejora la precisión de la siguiente operación.	
Refrenteado	Genera una superficie maquinada[1] plana en la	

[1] Maquinado: remoción de exceso de material [38].

	pieza de trabajo en el área donde se ubique.	

Tabla B.2: operaciones relacionadas con el taladrado [37].

B.4 Fresado

En la tabla B.3 se presentan las diferentes actividades que se pueden realizar en el fresado frontal que es utilizado para la RMT caso de estudio.

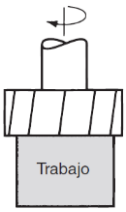
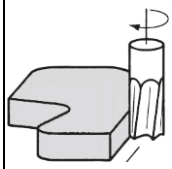
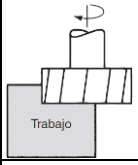
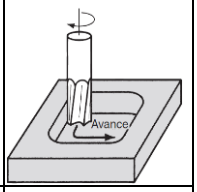
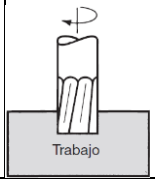
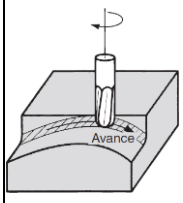
Operación	Descripción	Vista	Operación	Descripción	Vista
Frontal convencional	El diámetro de la fresa es más grande que la pieza de trabajo, sobrepasándola en ambos lados.		De perfiles	Igual que el terminal pero corta una pieza plana de la periferia.	
Frontal parcial	La fresa sobrepasa la pieza en un lado.		De Cavidades	Para cavidades poco profundas.	
Frontal terminal	El diámetro es menor que el ancho de la pieza para cortar una ranura.		De contorno superficial	Genera una superficie tridimensional mientras avanza su punta redonda.	

Tabla B.3: operaciones de fresado frontal [37]

Anexo C: Selección del a herramienta de simulación

Basado en las características de los RMS y el estándar IEC 61499 se han estudiado varias herramientas software, estas son:

C.1 FBDK

Fue la primera herramienta que se tuvo para el desarrollo de sistemas de control distribuido para el estándar IEC 61499. Permite visualizar los FBs y programar en lenguaje JAVA por lo que es beneficioso para la portabilidad pero no está de ninguna manera condicionada por las normas actuales y este lenguaje no está establecido en los estándares. FBDK es muy útil en el entorno académico para el aprendizaje del estándar IEC 61499 pero no fue desarrollado para una aplicación real en campo. Además, se pueden extraer los FB en XML y exportar e importar desde 4DIAC y NxtStudio [23]. Sin embargo, no cuenta con herramientas para el desarrollo de aplicaciones de control reconfigurable ni configuración de la comunicación [41].

C.2 IsaGraf

Es una herramienta útil de para diseño basado en el estándar IEC 61131 y se ha extendido para abordar IEC 61499. Su rendimiento es bajo en la aplicación del estándar IEC 61499 ya que es lento en la ejecución de los eventos debido a que la comunicación entre las partes del programa se lleva a cabo mediante el uso de variables de red, y es incompatible con otras plataformas desarrolladas [23].

C.3 4DIAC-IDE

Es una herramienta robusta que permite el desarrollo de sistemas de control distribuidos basados en el estándar IEC 61499. Esta permite depurar y realizar pruebas para ver el estado de las variables [23]. Además, cuenta con un soporte para el modelado de aplicaciones de reconfiguración y reconfiguración dinámica.

C.4 NxtOne

Es esta la aplicación comercial de NxtControl donde se pueden desarrollar y simular sistemas de control basados en IEC 61499. Esta herramienta posee un HMI para

visualizar el proceso a diferencia de 4DIAC, que proporciona una interfaz gráfica entre el FB y un HMI/SCADA [23]. Aunque la herramienta brinda un demo gratuito, es muy poco lo que se puede trabajar en este y acceder al paquete completo es difícil debido a su costo y la información sobre este software es muy poca.

C.5 Simatic IMAP

Esta herramienta implementa PROFINET CBA soportado en el estándar IEC 614499 que describe las tecnologías para la implementación de un sistema distribuido y modular basado en componentes. Aquí se puede realizar la configuración del sistema distribuido y la comunicación de los dispositivos de la red industrial [43]. Sin embargo, durante la exploración de la herramienta se observó que solo se muestran los componentes básicos del estándar IEC 61499 pero no permite una vista clara para el desarrollo de los modelos superiores propuestos por el estándar con el que se desea trabajar.

Anexo D: Descripción de los Bloques de Función creados

Para el desarrollo del proyecto se han creado los siguientes bloques:

D.1 Bloques de Función compuestos

Estos bloques están definidos por una red bloques conectados por datos y eventos. Los bloques de este tipo creados el trabajo son:

D.1.1 VELM1

Contiene la información de la velocidad que requiere el motor principal el cual manipula la punta que actúa sobre el producto. Se tienen dos eventos de entrada. INIT es el encargado de inicializar el FB y permite que todas las variables de entrada asociadas a este puedan muestrear los valores que entran en este, al igual que el evento REQ con sus variables asociadas. Los datos de entrada son QI que al leer un valor verdadero realiza el cálculo correspondiente para este FB, las coordenadas x, y y z son tomadas para los diferentes tipos de operación dependiendo de lo que se vaya a realizar y la variable tipo para saber si se va a taladrar o fresar definidos como 1 y 2, respectivamente. A la salida se tiene el estado del bloque y el valor de la velocidad con la que va a operar los cuales salen cuando el bloque se ha ejecutado con éxito y activa los eventos de salida indicando su finalización y su disponibilidad con INITO y CNF, respectivamente. Su interfaz externa y la red de trabajo que lo compone se muestran en la figura D.1. Los bloques de función internos se explican más adelante en la sección de D.2 de bloques de función básicos.

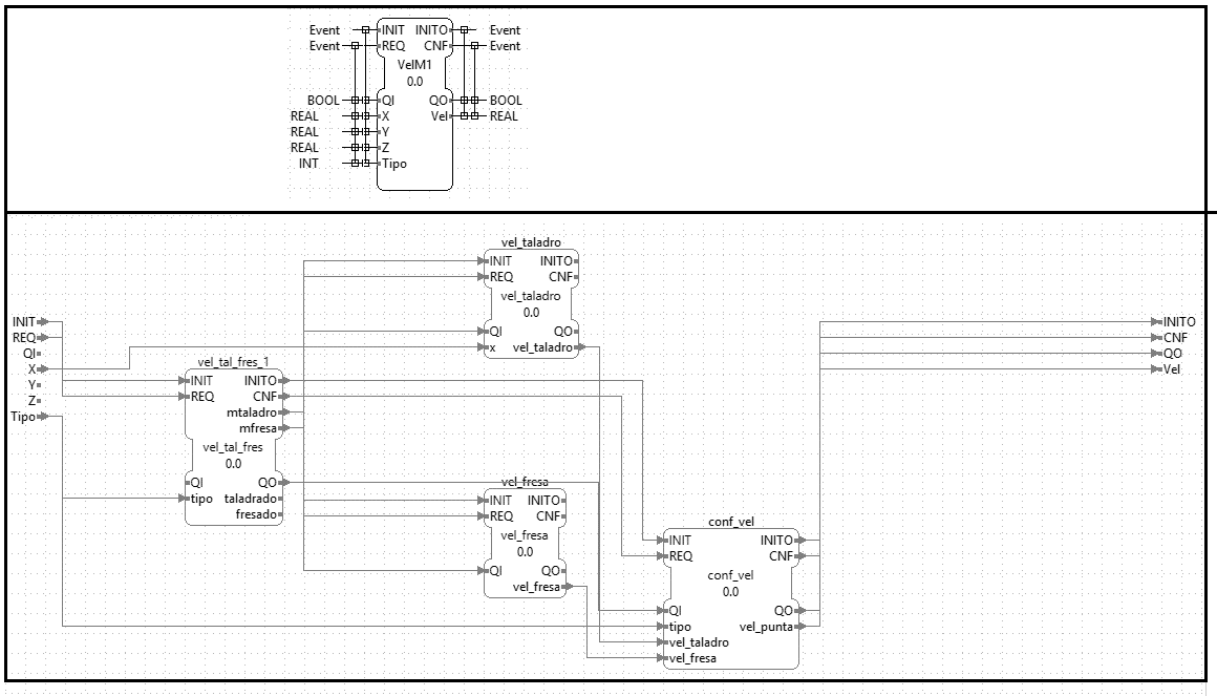


Figura D.1: FB VELM1.

D.1.2 VELM2

Este bloque define la velocidad para el motor secundario cuando es requerido en el proceso de torneado, utilizado para rotar constantemente la pieza de trabajo. Se tienen dos eventos de entrada INIT que es el encargado de inicializar el FB y permite que todas las variables de entrada asociadas a este puedan muestrear los valores que entran en este, y el evento REQ con sus variables asociadas para realizar el cálculo de la velocidad. Los datos de entrada son QI para el estado del bloque que al ser verdadero permite calcular la velocidad y la coordenada Y, que es el eje de desplazamiento sobre el cual se mueve la punta que trasforma la pieza de trabajo. A la salida se tiene el estado del proceso realizado por el bloque y *vel* que es el valor de la velocidad con la que va a operar el motor para el torneado. Con la ejecución del bloque se activan los eventos de salida asociados con sus variables de salida que son INITO indicando la finalización del cálculo realizado en la red de trabajo interna y CNF indicando que el bloque ya está disponible para un nuevo cálculo. La interfaz externa de este bloque compuesto y su red de trabajo se muestra en la figura D.2. El bloque de función interno se explica más adelante en la sección de D.2 de bloques de función básicos.

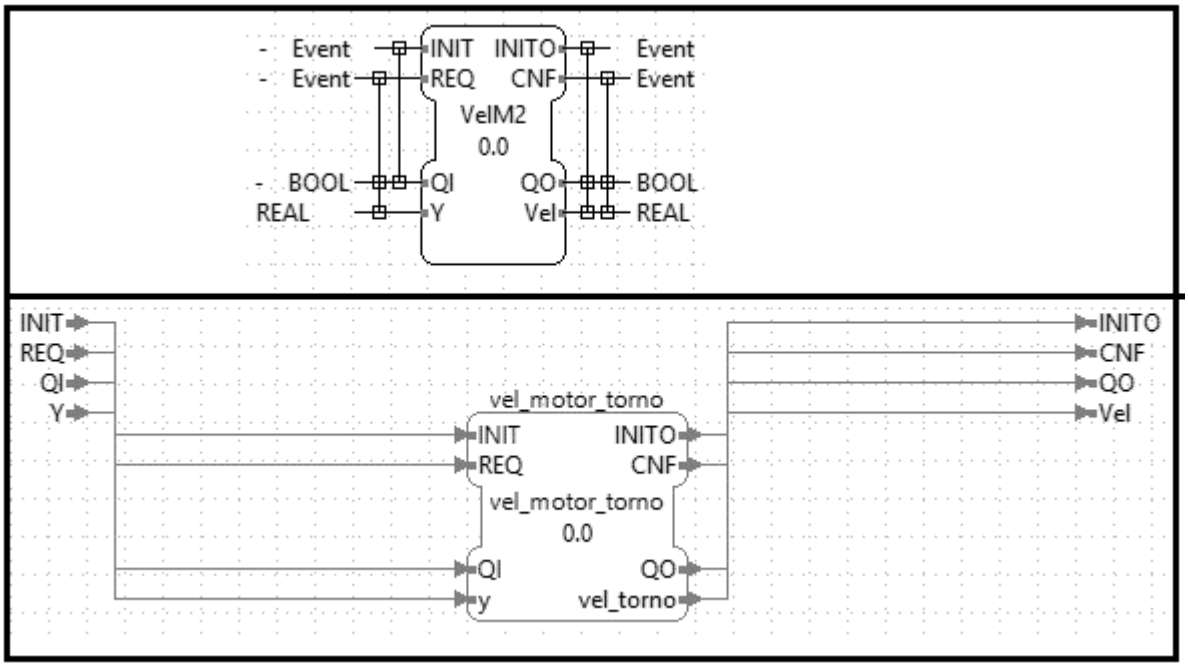


Figura D.2: FB VELM2.

D.1.3 Tipo_Operacion

Este bloque se ha creado con el fin de identificar el producto que se va a realizar y de acuerdo a esto activar las diferentes operaciones que se requieren y las piezas que se deben producir. El evento INIT inicializa el bloque de función y permite la lectura de la entrada QI que envía un estado verdadero para que se realice el proceso definido para este bloque. Se definieron dos modelos de sillas que entran por la variable de entrada *tipo_silla* que puede ser 1 o 2, la lectura de esta variable se realiza mediante la activación del evento REQ que ejecuta la red de trabajo para generar el producto y sale el tipo de producto realizado. La interfaz externa de este bloque compuesto se muestra en la figura D.3 con su red de trabajo en la parte inferior.

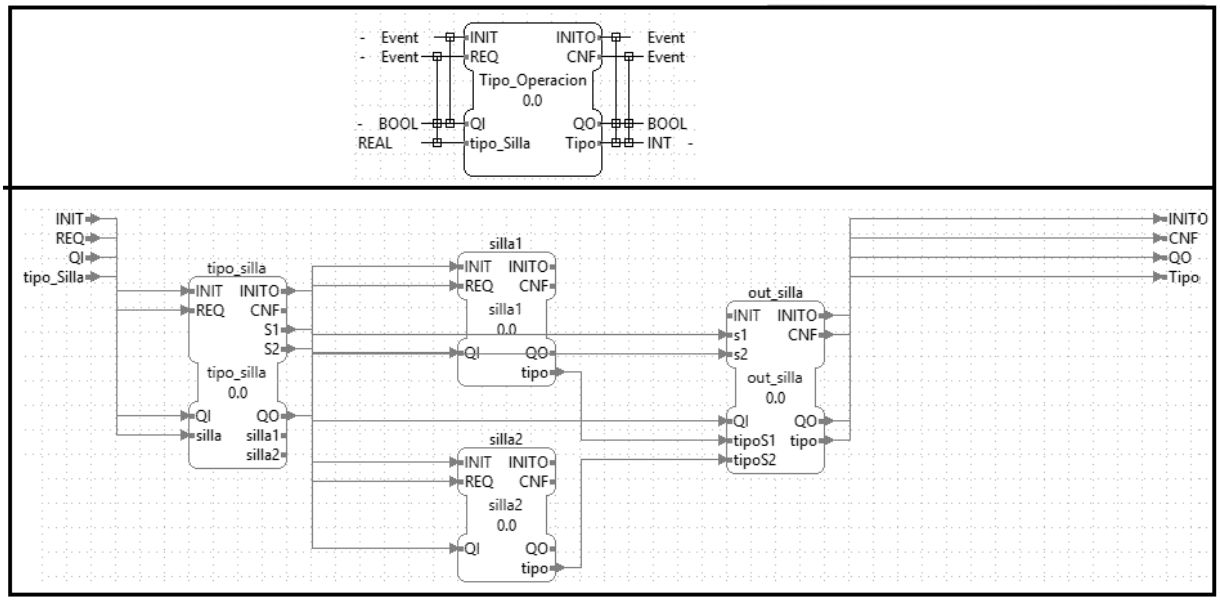


Figura D.3: FB Tipo_Operacion.

D.1.4 control_pid

El desarrollo del bloque de control PID se realiza mediante la creación de un boque compuesto, la representación de este con su red de trabajo se muestra en la figura D.4. El bloque se inicializa mediante el evento INIT y el evento REQ para la lectura de los datos de entrada. A la entrada está el estado que al ser verdadero ejecuta el control PID, se tiene el valor de referencia SetPoint y el valor real que realmente está manejando la maquina en la variable *valor_medido*, las constantes proporcional Kp, derivativa Td e integral Ti y el tipo de operación realizada en la variable *tipo*. Tras su ejecución se tiene el esfuerzo de control requerido y el estado que indica la finalización de la ejecución del bloque, con esto se activan los eventos de salida INTO y CNF que se ha finalizado la ejecución y que se puede realizar nuevamente el proceso, respectivamente.

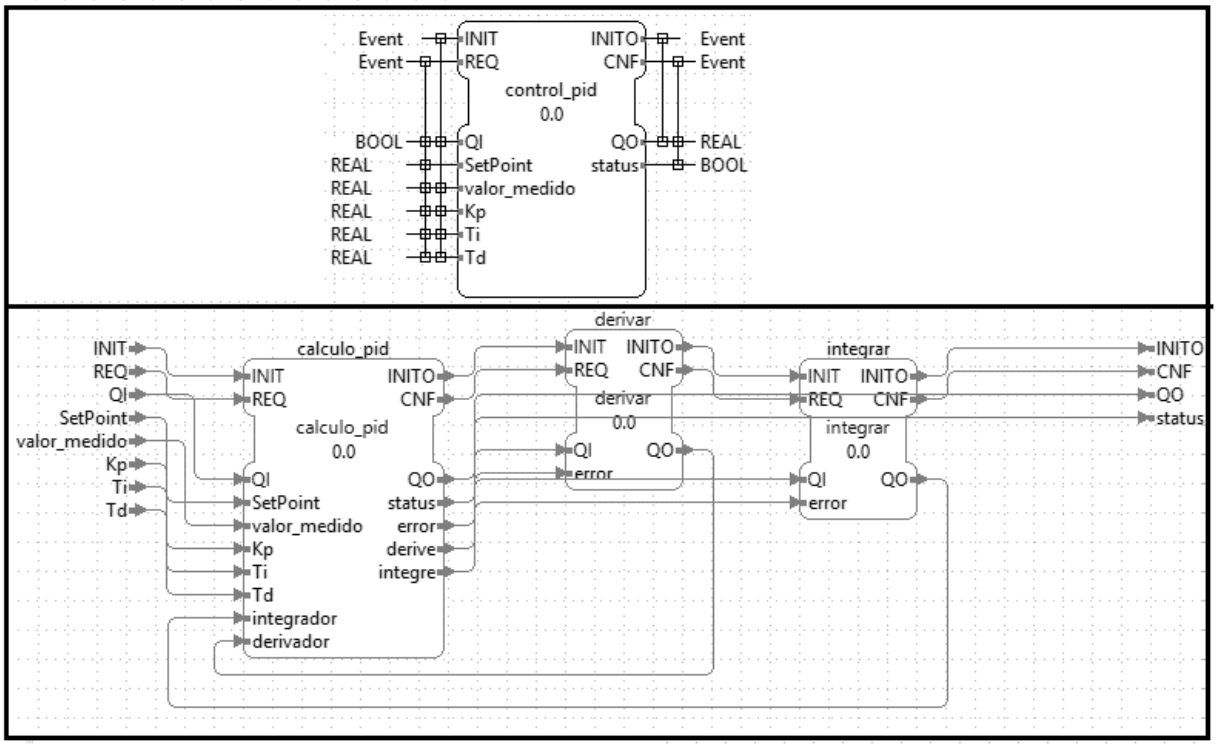


Figura D.4: FB control_pid.

D.1 Bloques de Función básicos

Estos FB básicos creados para el proyecto y utilizados en los bloque de función compuestos se presentan a continuación con sus respectivo gráfico de control de ejecución y los algoritmos realizados.

D.2.1 vel_taladro

Este bloque se encarga de calcular la velocidad de rotación que se muestra en la ecuación 3.2 del capítulo 3. La interfaz se muestra a la izquierda de la figura D.5 y su ECC a la derecha de esta. Se tiene el evento de entrada INIT para inicializar el bloque y el evento REQ que permite realizar el cálculo de la velocidad para la operación de taladrado. Entre los datos de salida se tiene QI que toma un valor TRUE para realizar el cálculo de la velocidad y está *x* para definir la dimensión del diámetro de la broca con la que se corte la pieza de trabajo. Como datos de salida se tiene QO indicando el estado actual de la variable calculada en este bloque que es la velocidad de rotación y *vel_taladro* que es el valor calculado. Por último, los eventos de salida son INITO

indicando el fin de su ejecución y CNF para indicar que el bloque ya está disponible para un nuevo cálculo.

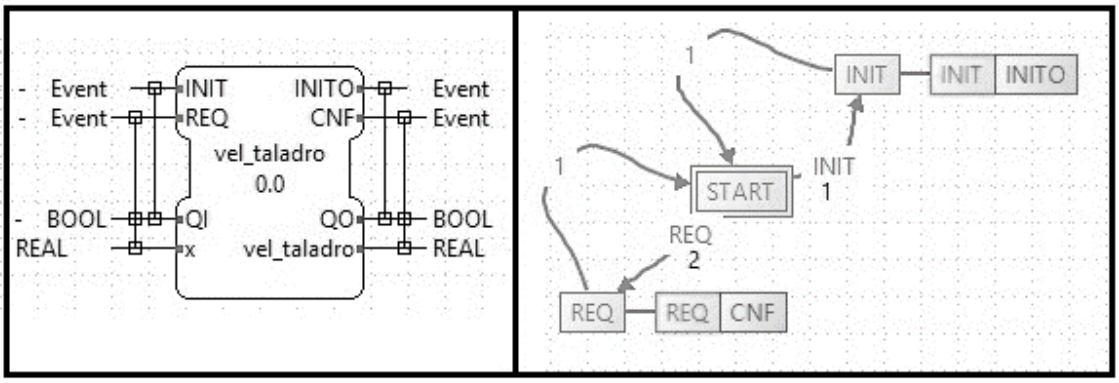


Figura D.5: FB vel_taladro.

Los algoritmos para este bloque se presentan a continuación:

Algoritmo INIT

```
vel_corte := 7500; /*mm/min*/
```

Algoritmo REQ

```
vel_taladro := vel_corte/(3.1416*x);
QO := TRUE;
```

D.2.2 vel_fresa

Este bloque se encarga de calcular la velocidad de rotación presentada en la ecuación 3.3 del capítulo 3. El FB vel_fresado se muestra en la figura D.6 cuya interfaz es está a la izquierda y el ECC está derecha. Se tiene el evento de entrada INIT para inicializar el bloque y el evento REQ que permite realizar el cálculo de la velocidad para la operación de fresado. Entre los datos de entrada está el dato de entrada QI que toma un valor TRUE para realizar el cálculo de la velocidad. Como datos de salida se tiene QO, indicando el estado actual de la variable calculada en este bloque que es la velocidad de rotación y *vel_fresa* que es el valor calculado. Por último, los eventos de salida son INITO indicando el fin de su ejecución y CNF para indicar que el bloque ya está disponible para un nuevo cálculo.

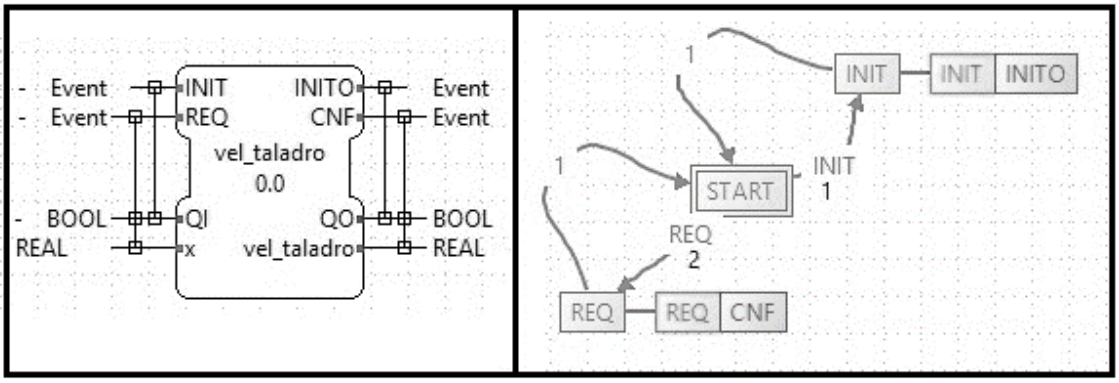


Figura D.6: FB vel_fresa.

Los algoritmos para este bloque se presentan a continuación:

Algoritmo INIT

```
vel_corte := 7500; /*mm/min*/
x := 0.02; /*m diámetro exterior de la fresa */
```

Algoritmo REQ

```
vel_fresa := vel_corte/(3.1416*x);
```

D.2.3 vel_motor_torno

Este bloque se encarga de calcular la velocidad de rotación para el motor utilizado en el torneado presentada en la ecuación 3.1 del capítulo 3. La interfaz y su ECC se pueden observar en la figura D.7. Se tiene el evento de entrada INIT para inicializar el bloque y el evento REQ que permite realizar el cálculo de la velocidad para el motor en el torneado. Está el dato de entrada QI que toma un valor TRUE para realizar el cálculo de la velocidad y la coordenada *y* que es definida de acuerdo al diseño del producto indicando el diámetro de la pieza de trabajo. Como datos de salida se tiene QO indicando el estado actual de la variable calculada en este bloque que es la velocidad de rotación y *vel_torno* que es el valor calculado. Por último, los eventos de salida son INITO indicando el fin de su ejecución y CNF para indicar que el bloque ya está disponible para un nuevo cálculo.

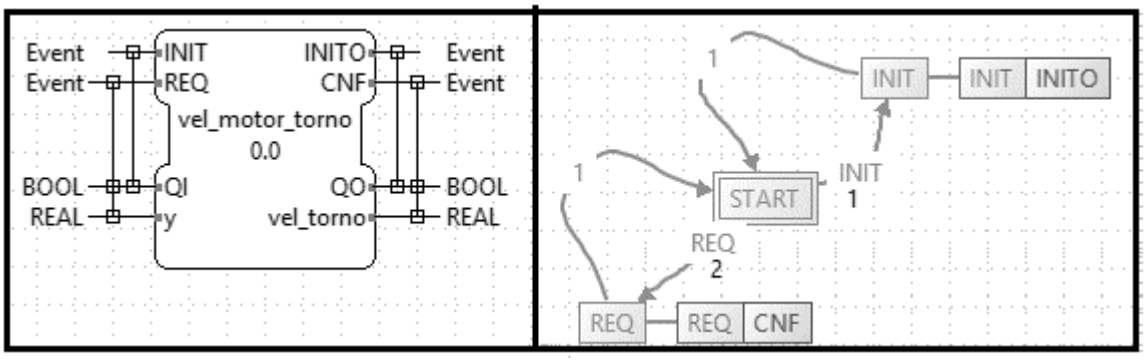


Figura D.7: FB vel_motor_torno.

Los algoritmos para este bloque se presentan a continuación:

Algoritmo INIT

```
vel_corte := 7500; /*mm/min*/
```

Algoritmo REQ

```
vel_torno := vel_corte/(3.1416*y);
QO := TRUE;
```

D.2.4 vel_tal_fres

Se encarga de tomar la decisión de activar los bloques que calculan la velocidad de rotación del motor de la punta que actúa sobre la pieza de trabajo, esto dependiendo del tipo de operación que se desee realizar (taladrado o fresado) enviando una variable TRUE para uno y FALSE para el otro. Se tiene el evento de entrada INIT para inicializar el bloque y el evento REQ que permite definir qué salida debe ser verdadera y cual falsa. Se cuenta con el dato de entrada QI que toma un valor TRUE para ejecutar el bloque y el tipo que indica la operación que se va a realizar. Como datos de salida están QO indicando el estado del FB y las variables: *taladrado* y *fresado* que son de tipo booleanas que se activan de forma negativa o positiva dependiendo la operación indicada a la máquina. Por último, los eventos de salida son INITO indicando el fin de su ejecución, CNF para indicar que el bloque ya está disponible para un nuevo cálculo y los eventos que activan los FB para el cálculo de la velocidad *mtaladro* y *mfresa*. La figura D.8 muestra a la izquierda la interfaz y a la derecha el grafico de control de ejecución de este FB.

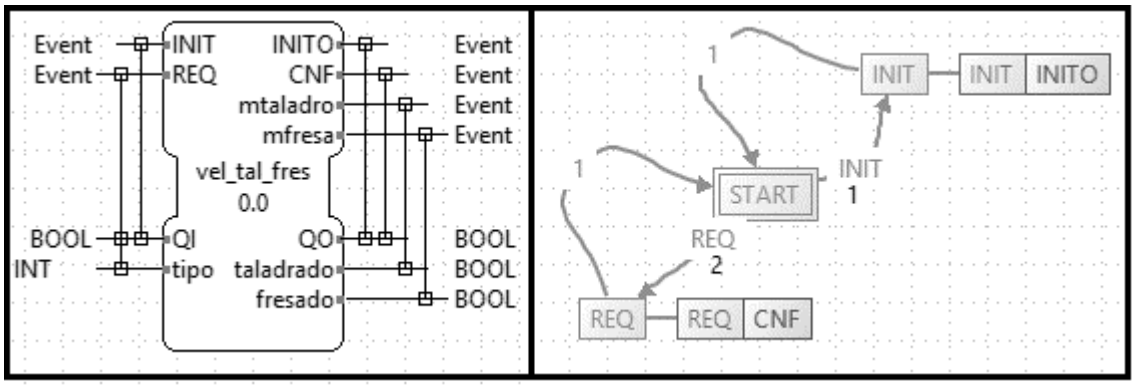


Figura D.8: FB vel_tal_fres.

Los algoritmos para este bloque se presentan a continuación:

Algoritmo INIT

```
QI := TRUE;
```

Algoritmo REQ

```
IF tipo =1 THEN
vel_punta := vel_taladro;
QO := TRUE;
ELSE
        IF tipo = 2 THEN
        vel_punta := vel_fresa;
        QO := TRUE;
        END_IF;
END_IF;
```

D.2.5 conf_vel

Recibe la velocidad calculada para los motores y la envía al motor para que actúe sobre la pieza de trabajo dependiendo de la operación que se va a ejecutar. Se cuenta con el evento de entrada INIT para inicializar el bloque y el evento REQ que permite definir realizar la función que tiene el FB. A la entrada se tiene el dato QI que toma un valor TRUE para ejecutar el bloque, el tipo que indica la operación que se va a realizar y los valores de la velocidad de rotación para taladrado y fresado: *vel_taladro* y *vel_punta,* respectivamente. Como datos de salida se tiene QO indicando el estado del FB y la velocidad de debe ser enviada al motor principal con el que trabaja la punta de la máquina. Por último, los eventos de salida que son INITO indicando el fin de su

ejecución y CNF para indicar que el bloque ya está disponible. La figura D.9 muestra a la izquierda la interfaz y a la derecha el grafico de control de ejecución de este FB.

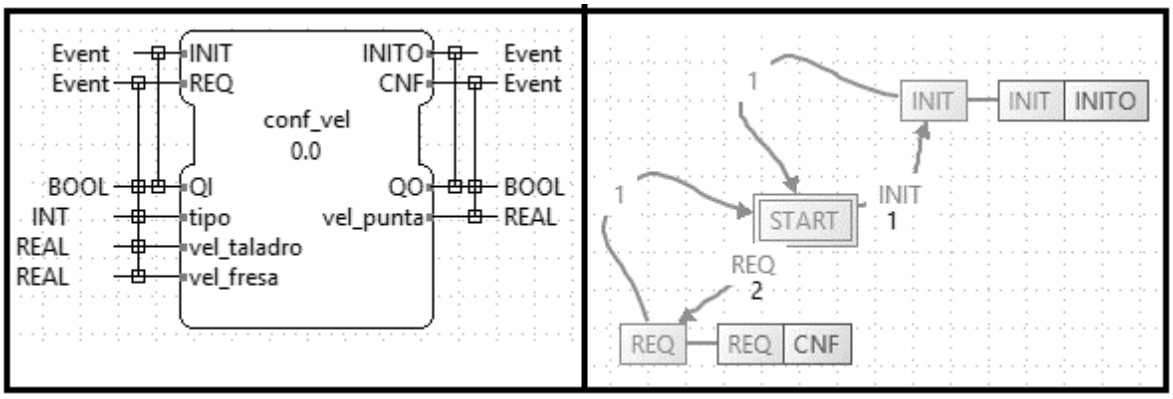


Figura D.9: FB conf_vel.

Los algoritmos para este bloque se presentan a continuación:

Algoritmo INIT

```
QI := TRUE;
```

Algoritmo REQ

```
IF tipo =1 THEN
        QO := TRUE;
        taladrado := TRUE;
        fresado := FALSE;
    ELSE
        IF tipo = 2 THEN
            QO := TRUE;
            taladrado := FALSE;
            fresado := TRUE;
        END_IF;
    END_IF;
```

D.2.6 tipo_silla

Para conocer el tipo de silla y activar el proceso para su fabricación correspondiente se cuenta con el bloque llamado *tipo_silla* que se muestra en la figura D.10 con su respectivo ECC. Se tiene el evento de entrada INIT para inicializar el bloque y el evento REQ que permite realizar su función. A la entrada está el dato de QI que toma un valor TRUE para ejecutar el bloque y silla que indica el producto que se va a fabricar, para

esto se definieron dos modelos de sillas en el capítulo 3. Como datos de salida se tiene QO indicando el estado del FB y *silla1* y *silla2* que se activan dependiendo del producto que se va a realizar como true o false. Por último, los eventos de salida que son INITO indicando el fin de su ejecución, CNF para indicar que el bloque ya está disponible y los eventos que activan el procedimiento para el producto solicitado S1 y S2.

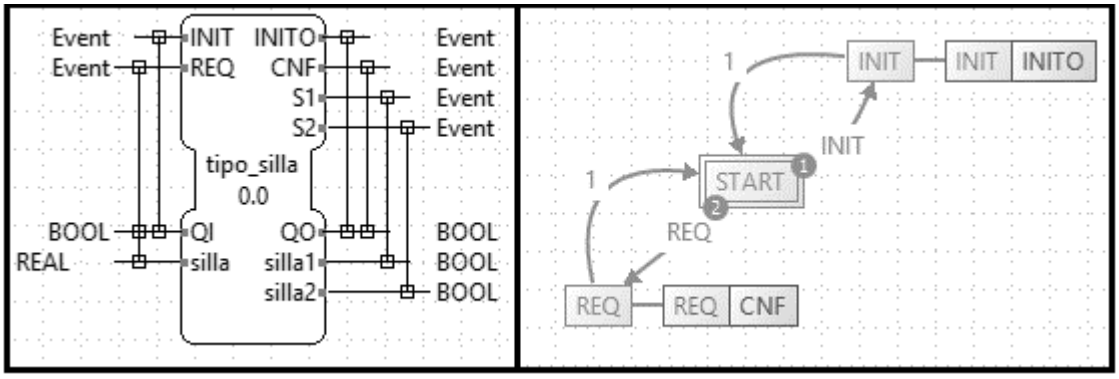


Figura D.10: FB tipo_silla.

Los algoritmos para este bloque se presentan a continuación:

Algoritmo INIT

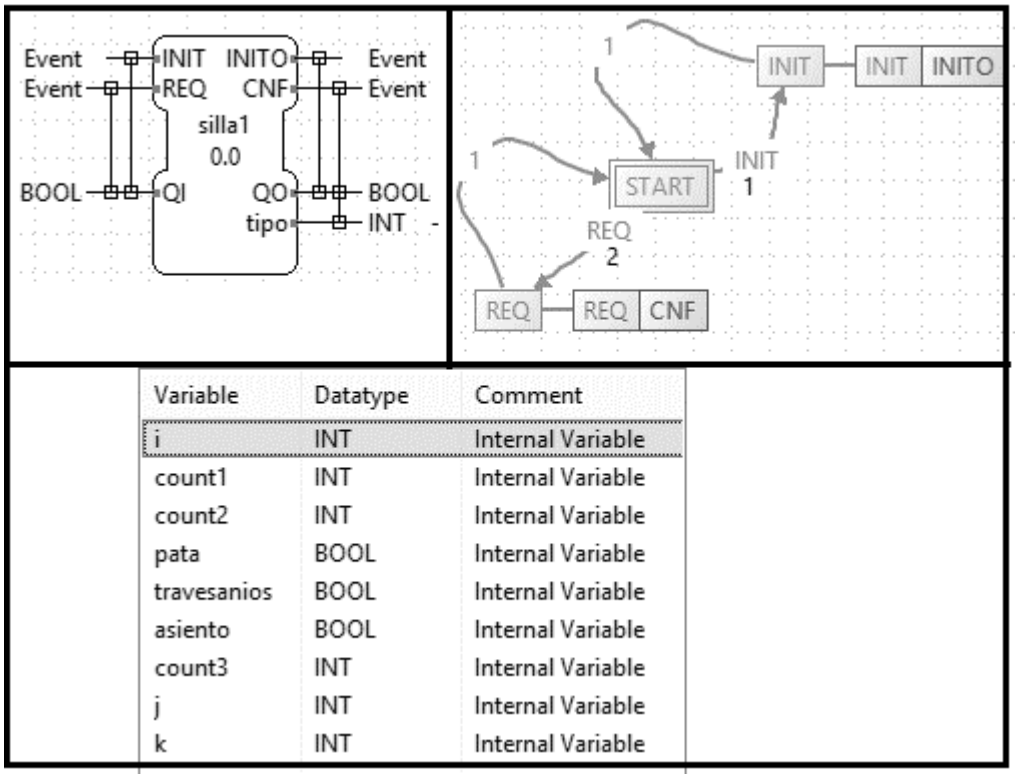
```
QI := TRUE;
```

Algoritmo REQ

```
IF silla =1 THEN
silla1 := TRUE;
silla2 := FALSE;
QO := TRUE;
ELSE
        IF silla = 2 THEN
        silla2 := TRUE;
        silla1 :=FALSE;
        QO := TRUE;
        END_IF;
END_IF;
```

D.2.7 silla1

Este contiene las operaciones que se deben realizar y el conteo de las piezas obtenidas para la fabricación del primer producto definido en la sección 3.2.1 como taburete de cuatro patas. El FB *silla1* con su respectivo ECC y las variables internas que se definieron para el conteo se muestra en la figura D.11. Se tiene el evento de entrada INIT para inicializar el bloque y el evento REQ para iniciar las operaciones para este producto. A la entrada está el dato de QI que lee un valor TRUE para ejecutar el bloque. Como datos de salida se tiene QO indicando el estado del FB y tipo con el tipo de producto fabricado. Por último, los eventos de salida que son INITO indicando el fin de su ejecución, CNF para indicar que el bloque ya está disponible.



Variable	Datatype	Comment
i	INT	Internal Variable
count1	INT	Internal Variable
count2	INT	Internal Variable
pata	BOOL	Internal Variable
travesanios	BOOL	Internal Variable
asiento	BOOL	Internal Variable
count3	INT	Internal Variable
j	INT	Internal Variable
k	INT	Internal Variable

Figura D.11: FB silla1.

Los algoritmos ejecutados en el bloque de la figura D.11 se muestran a continuación:

Algoritmo INIT

```
QI := TRUE;
```

Algoritmo REQ:

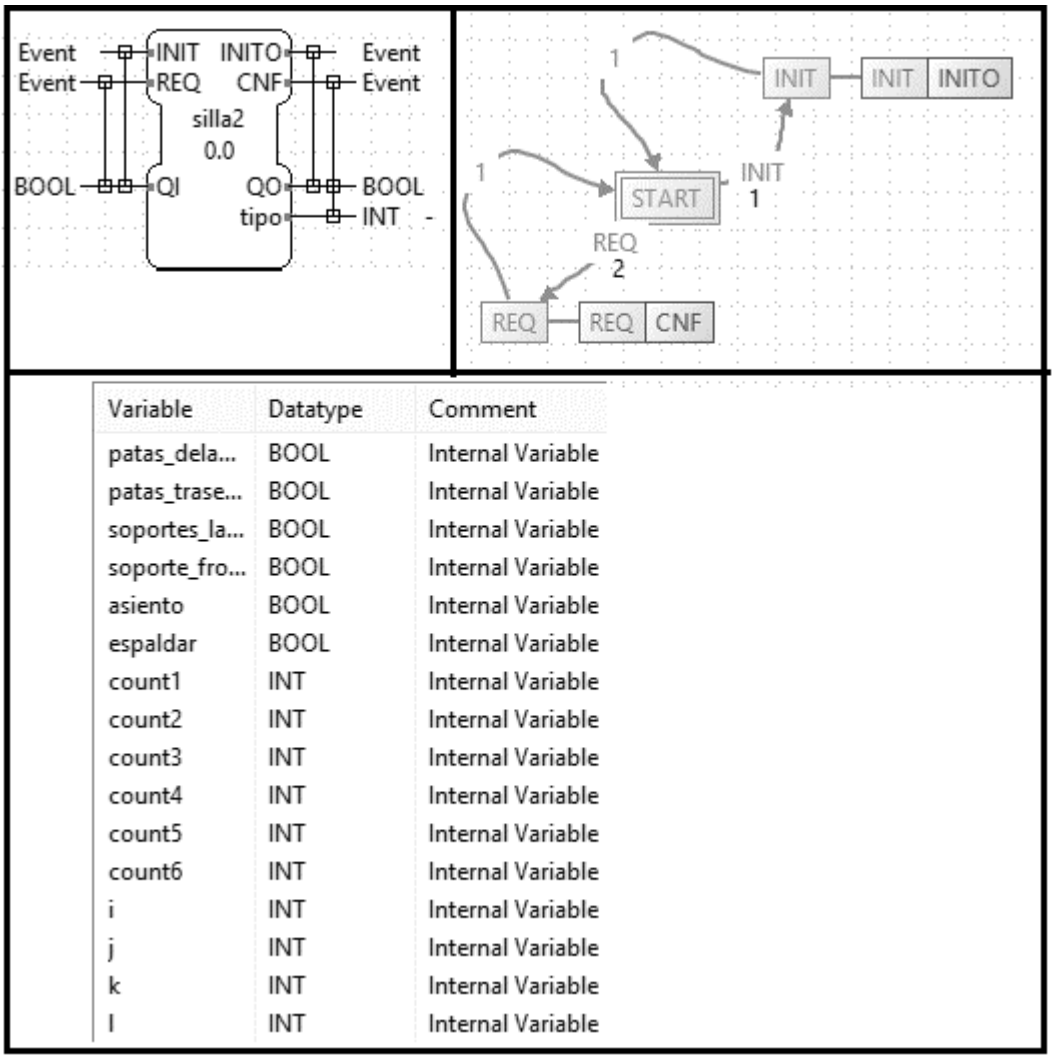
```
i :=0;
```

```
j := 0;
k := 0;
count1 := 0;
count2 := 0;
count3 := 0;
pata := FALSE;
travesanios := FALSE;
asiento := FALSE;
/*patas */
WHILE i< 4 DO
        WHILE count1 <3 DO
                IF count1 = 0 THEN
                        tipo := 3;
                        ELSE
                        IF count1 = 1 THEN
                                tipo := 1;
                        ELSE
                        IF count1 =2 THEN
                        tipo :=1;
                        END_IF;
                END_IF;
            END_IF;
            count1 := count1+1;
            END_WHILE;
        i := i+1;
        pata := TRUE;
        END_WHILE;
/*travesaño */
WHILE i< 3 DO
        WHILE count1 <3 DO
                IF count1 = 0 THEN
                        tipo := 3;
                        ELSE
                        IF count1 = 1 THEN
                                tipo := 1;
                        ELSE
                        IF count1 =2 THEN
                        tipo :=1;
                        END_IF;
                END_IF;
            END_IF;
```

```
                count1 := count1+1;
                END_WHILE;
        i := i+1;
        travesanios := TRUE;
        END_WHILE;
/*asiento */
WHILE count1 <4 DO
                        IF count1 = 0 THEN
                                tipo := 2;
                                ELSE
                                IF count1 = 1 THEN
                                        tipo := 2;
                                ELSE
                                IF count1 =2 THEN
                                tipo :=1;
                                ELSE
                                                IF count1 =3 THEN
                                                tipo :=1;
                                        END_IF;
                                END_IF;
                        END_IF;
                END_IF;
                count1 := count1+1;
                asiento := TRUE;
                END_WHILE;
```

D.2.8 silla2

Este contiene las operaciones que se deben realizar y el conteo de las piezas obtenidas para la fabricación del segundo producto definido en la sección 3.2.2 como silla de madera para adultos. El FB *silla2* con su respectivo ECC y las variables internas que se definieron para el conteo se muestra en la figura D.12. Se tiene el evento de entrada INIT para inicializar el bloque y el evento REQ para iniciar las operaciones para este producto. A la entrada está el dato de QI que lee un valor TRUE para ejecutar el bloque. Como datos de salida se tiene QO indicando el estado del FB y tipo con el tipo de producto fabricado. Por último, los eventos de salida que son INITO indicando el fin de su ejecución, CNF para indicar que el bloque ya está disponible.



Variable	Datatype	Comment
patas_dela...	BOOL	Internal Variable
patas_trase...	BOOL	Internal Variable
soportes_la...	BOOL	Internal Variable
soporte_fro...	BOOL	Internal Variable
asiento	BOOL	Internal Variable
espaldar	BOOL	Internal Variable
count1	INT	Internal Variable
count2	INT	Internal Variable
count3	INT	Internal Variable
count4	INT	Internal Variable
count5	INT	Internal Variable
count6	INT	Internal Variable
i	INT	Internal Variable
j	INT	Internal Variable
k	INT	Internal Variable
l	INT	Internal Variable

Figura D.12: FB silla2.

Los algoritmos ejecutados para este bloque se muestran a continuación:

Algoritmo INIT

```
QI := TRUE;
```

Algoritmo REQ:

```
i := 0;
j := 0;
k := 0;
l := 0;
count1 := 0;
count2 := 0;
```

```
count3 := 0;
count4 := 0;
count5 := 0;
count6 := 0;
patas_delanteras := FALSE;
patas_traseras := FALSE;
soportes_laterales := FALSE;
soporte_frontal := FALSE;
asiento := FALSE;
espaldar := FALSE;
/*patas_delanteras 2*/
WHILE i< 2 DO
        WHILE count1 <3 DO
                        IF count1 = 0 THEN
                                tipo := 2;
                                ELSE
                                IF count1 = 1 THEN
                                        tipo := 1;
                                ELSE
                                IF count1 =2 THEN
                                tipo :=1;
                                END_IF;
                        END_IF;
                END_IF;
                count1 := count1+1;
                END_WHILE;
        i := i+1;
        patas_delanteras := TRUE;
        END_WHILE;
/*patas_traseras 2*/
WHILE j< 2 DO
        WHILE count2 <4 DO
                        IF count2 = 0 THEN
                                tipo := 2;
                                ELSE
                                IF count2 = 1 THEN
                                        tipo := 2;
                                ELSE
                                IF count2 =2 THEN
                                tipo :=1;
                                ELSE
```

```
                                        IF count2 =2 THEN
                                        tipo :=1;
                                        END_IF;
                                END_IF;
                        END_IF;
                        count2 := count2+1;
                        END_WHILE;
        j := j+1;
        patas_traseras := TRUE;
        END_WHILE;
/*soportes_laterales 2*/
WHILE k< 2 DO
        WHILE count3 <3 DO
                        IF count3 = 0 THEN
                                tipo := 2;
                                ELSE
                                IF count3 = 1 THEN
                                        tipo := 1;
                                ELSE
                                IF count3 =2 THEN
                                tipo :=1;
                                END_IF;
                        END_IF;
                        count3 := count3+1;
                        END_WHILE;
        k := k+1;
        soportes_laterales := TRUE;
        END_WHILE;
/*soporte_frontal 4*/
WHILE l< 4 DO
        WHILE count4 <4 DO
                        IF count4 = 0 THEN
                                tipo := 2;
                                ELSE
                                IF count4 = 1 THEN
                                        tipo := 2;
                                ELSE
                                IF count4 =2 THEN
                                tipo :=1;
```

```
                    ELSE
                                IF count4 =2 THEN
                                tipo :=1;
                                END_IF;
                    END_IF;
          END_IF;
          END_IF;
          count4 := count4+1;
          END_WHILE;
l := l+1;
soporte_frontal := TRUE;
END_WHILE;
/*asiento  */
WHILE count5 <4 DO
          IF count5 = 0 THEN
                    tipo := 2;
                    ELSE
                    IF count5 = 1 THEN
                              tipo := 2;
                    ELSE
                    IF count5 =2 THEN
                    tipo :=1;
                    ELSE
                              IF count5 =3 THEN
                              tipo :=1;
                              END_IF;
                    END_IF;
          END_IF;
          END_IF;
          count5 := count5 +1;
          asiento := TRUE;
          END_WHILE;
/*espaldar */
WHILE count6 <4 DO
          IF count6 = 0 THEN
                    tipo := 3;
                    ELSE
                    IF count6 = 1 THEN
                              tipo := 1;
                    ELSE
                    IF count6 =2 THEN
```

```
                    tipo :=1;
                    ELSE
                        IF count6 =3 THEN
                        tipo :=1;
                        END_IF;
                    END_IF;
                END_IF;
            END_IF;
            count6 := count6 +1;
            espaldar := TRUE;
            END_WHILE;
```

D.2.9 out_silla

Encargado de leer la silla realizada y envía el dato del producto fabricado. Este bloque se muestra en la figura D.13 con su ECC. Cuenta con el evento de entrada INIT para inicializar el bloque y los eventos *s1* y *s2* para activar la lectura del tipo de silla realizada 1 o 2. A la entrada se tiene el dato de QI que toma un valor TRUE para ejecutar el bloque y el tipo de producto *tipoS1* y *tipoS2*. Como datos de salida están QO indicando el estado del FB y *tipo* indicando el producto fabricado. Por último, los eventos de salida son INITO indicando el fin de su ejecución y CNF para indicar que el bloque ya está disponible para ser utilizado nuevamente. La figura D.13 muestra a la izquierda la interfaz y a la derecha el grafico de control de ejecución de este FB.

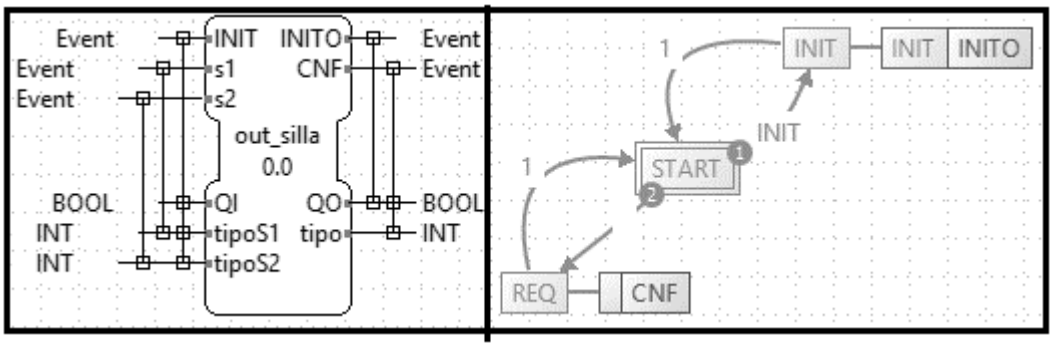


Figura D.13: FB out_silla.

Los algoritmos ejecutados en el bloque out_silla son los siguientes:

Algoritmo INIT:

QI := **TRUE**;

Algoritmo s1:

tipo := tipoS1;

QO := **TRUE**;

Algoritmo s2:

tipo := tipoS2;

QO := **TRUE**;

D.2.10 decision

El FB decision mostrado en la figura D.14 es el encargado de leer el tipo de operación que se desea realizar (torneado, taladrado o fresado) para saber si solo debe activar el motor encargado de manejar la punta o también el que mueve la pieza para tornear. Para esto se definen las tres operaciones con números enteros, donde 1 es taladrado, 2 es fresado y 3 torneado. Se tiene el evento de entrada INIT para inicializar el bloque y el evento REQ para iniciar la toma de la decisión. A la entrada están el dato de QI que lee un valor TRUE para ejecutar el bloque y el tipo de operación a realizar. Como datos de salida se cuenta con QO indicando el estado del FB, *punta* para activar el motor de la puta que actúa sobre la pieza de trabajo y *torno* para activar el motor secundario utilizado en la operación de torneado (solo en el caso del torneado se activan los dos motores como se puede ver en los algoritmos). Por último, los eventos de salida son INITO indicando el fin de su ejecución, CNF para indicar que el bloque ya está disponible y los eventos: *mpunta* y *mtorno* que activan los bloques que realizan los cálculos de la velocidad de los motores de la punta y el torno, respectivamente.

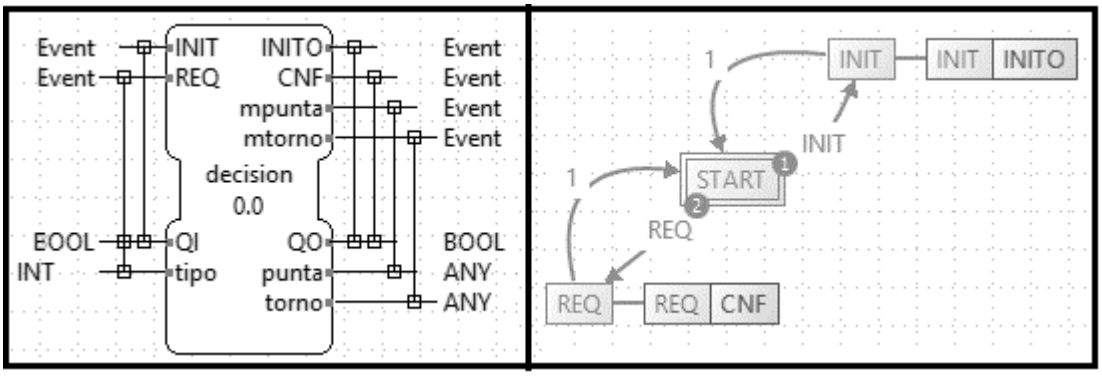


Figura D.14: FB decisión.

El algoritmo realizado para este bloque es el siguiente:

Algoritmo INIT:

```
QI := TRUE;
```

Algoritmo REQ

```
IF tipo =1 THEN
        QO := TRUE;
        punta := TRUE;
        torno := FALSE;
    ELSE
        IF tipo = 2 THEN
            QO := TRUE;
            punta := TRUE;
            torno := FALSE;
        ELSE
            IF tipo = 3 THEN
                QO := TRUE;
                punta := TRUE;
                torno := TRUE;
                END_IF;
            END_IF;
    END_IF;
```

D.2.11 derivar

El bloque derivar que se muestra en la figura D.15 realiza el cálculo de la derivada por medio del error. Se tiene el evento de entrada INIT para inicializar el bloque y el evento REQ para realizar el cálculo de la derivada. A la entrada están el dato de QI que lee un valor TRUE para ejecutar el bloque y *error* para la lectura del error. Como dato de salida se cuenta con QO con el cálculo de la derivada. Por último, los eventos de salida que son INITO indicando el fin de su ejecución y CNF para indicar que el bloque ya está disponible para un nuevo cálculo.

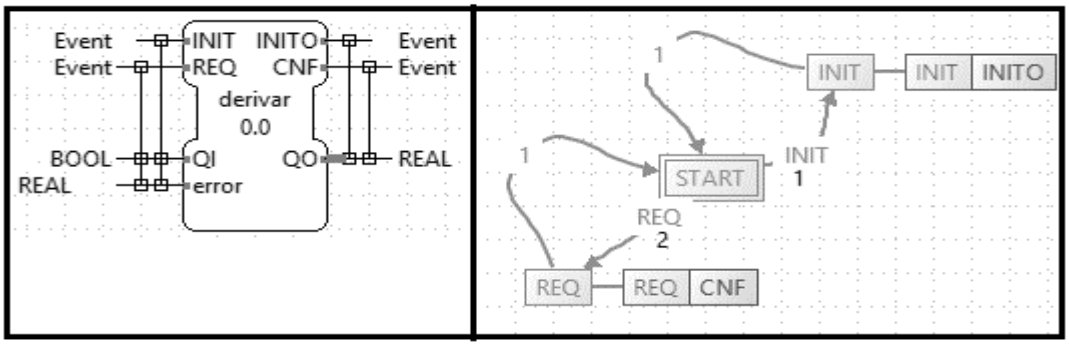


Figura D.15: FB derivar.

En el bloque derivar se define una variable interna *e* de tipo real a la que se le asigna el valor del error. Los algoritmos presentes en este bloque se presentan a continuación.

Algoritmo INIT:

e := error;

algoritmo REQ:

QO := (error-**e**)/100 /*100 es el valor del tiempo de muestreo */

e := error;

D.2.12 integrar

El bloque integrar de la figura D.16 se encarga de calcular el valor de la integral. Se tiene el evento de entrada INIT para inicializar el bloque y el evento REQ para realizar el cálculo de la integral. A la entrada están el dato QI que lee un valor TRUE para realizar el cálculo y error para la lectura del error. Como dato de salida están QO con el cálculo de la integral obtenida a partir del error. Por último, los eventos de salida son INITO indicando el fin de su ejecución y CNF para indicar que el bloque ya está disponible para un nuevo cálculo.

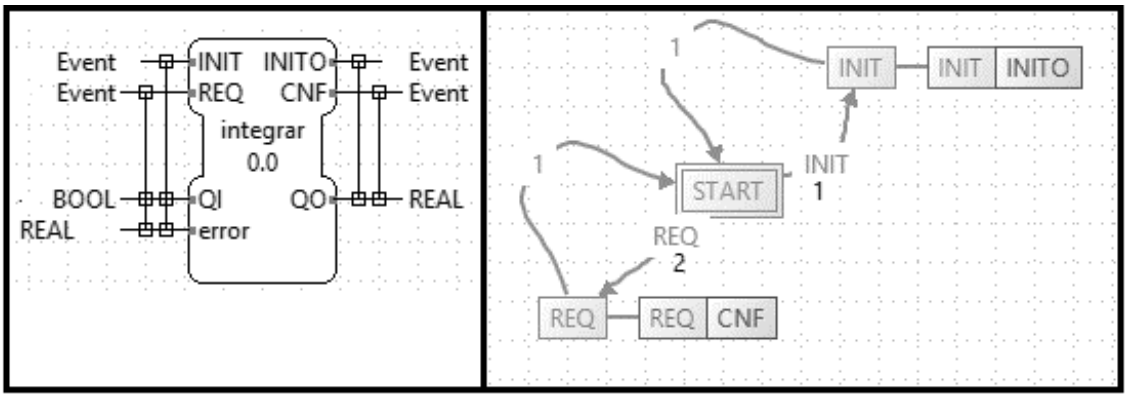


Figura D.16: FB integrar.

Los algoritmos presentes en este bloque son los siguientes:

Algoritmo INIT:

```
QI := TRUE;
```

Algoritmo REQ

```
QO := QO + error*100;
```

D.2.13 calculo_pid

Se utiliza para calcular el esfuerzo de control, se tiene el evento de entrada INIT para inicializar el bloque y el evento REQ para realizar el cálculo, a la entrada están el dato de QI que lee un valor TRUE para realizar el cálculo, *SetPoint* que es el valor de referencia, *valor_medido* dato real medidos desde el proceso, la contante proporcional *Kp*, la constante integral *Ti*, la constante de la derivada *Td*, el cálculo de la derivada como dato para *derivador* y el cálculo de la integral leído en la variable *integrador*. Entre los datos de salida están STATUS indicando el estado del bloque, QO con el esfuerzo de control calculado, *derive* para activar el bloque para un nuevo cálculo de la derivada e *integre* para activar el bloque para un nuevo cálculo de la integral. Por último, los eventos de salida que son INITO indicando el fin de su ejecución y CNF para indicar que el bloque ya está disponible para un nuevo cálculo, este bloque se muestra en la figura D.17.

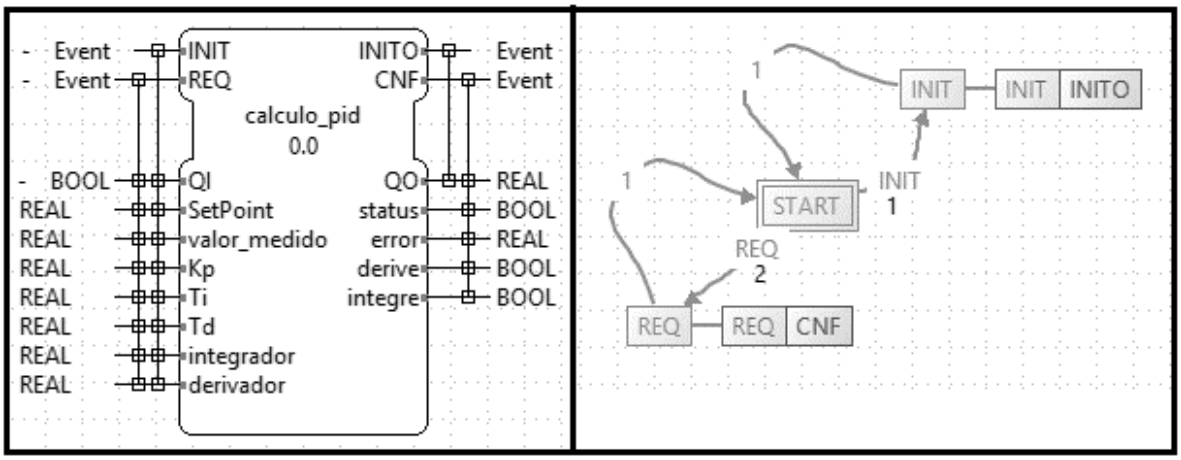


Figura D.17: FB calculo_pid.

Los algoritmos ejecutados aquí son los siguientes:

Algoritmo INIT:

```
QI := TRUE;
```

Algoritmo REQ:

```
error := SetPoint-valor_medido;
derive := TRUE;
integre  := TRUE;
QO :=(error + (1/Ti)*integrador + Td*derivador) * Kp;  /*tiempo de muestreo=100 */
status := TRUE;
```

Anexo E: Manejo de 4DIAC

Para el desarrollo del proyecto se necesitó el uso de una herramienta software de modelado con base en IEC 61499, en el anexo C se muestran los criterios por los cuales se seleccionó 4DIAC y en este anexo se mostraran algunas de sus funcionalidades básicas a través del ejemplo del Recurso COM_PC del dispositivo MAQUINA y algunos bloques propios creados para este trabajo:

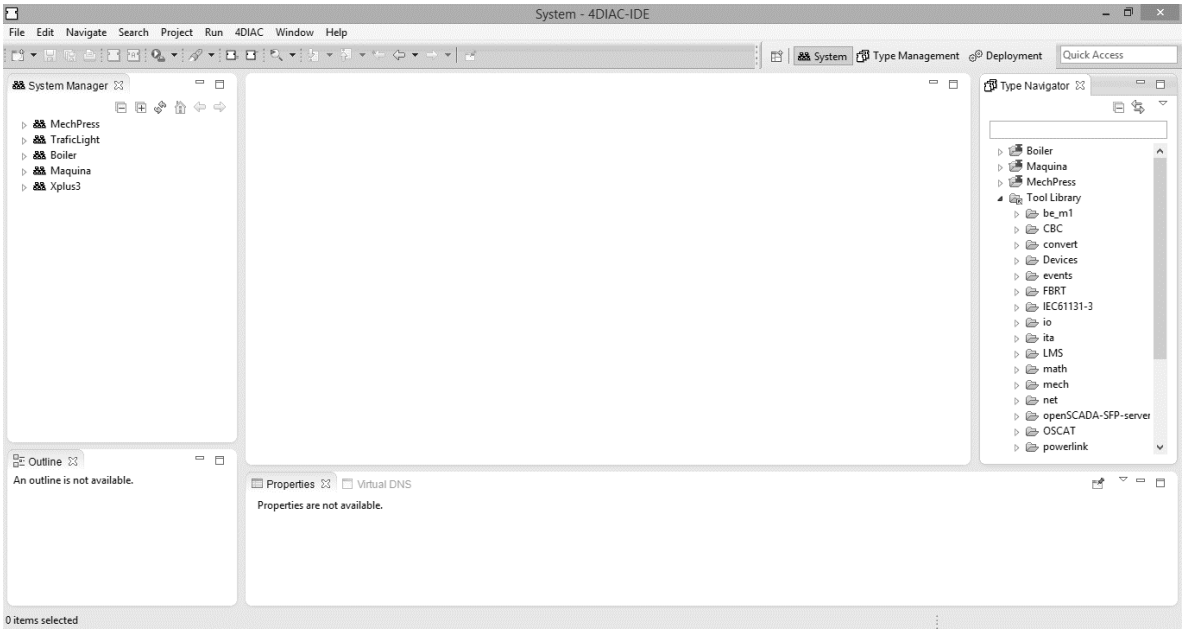


Figura E.1: Vista general del programa, columna izquierda: navegador de sistemas, central: área de trabajo, columna derecha: biblioteca de FB, Inferior: visor de propiedades.

E.1 Cómo crear un sistema

El primer paso para crear un proyecto o sistema, es dar click en el menú File-New-NewSystem, así:

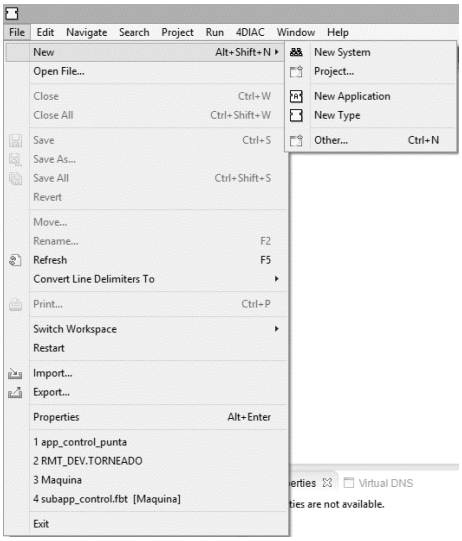


Figura E.2: Menú File, nuevo sistema.

Una vez que se da click en crear nuevo sistema se despliega la ventana de la Figura E.1.2 en donde se podrá nombrar el sistema y cambiar la direccion donde será creado:

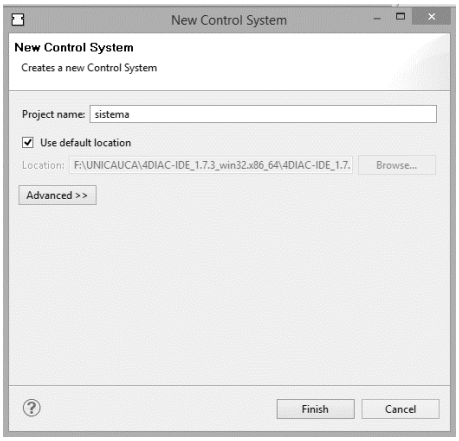


Figura E.3: Creación del nuevo sistema.

Cuando el sistema sea creado se verá de nuevo la vista general del programa Figura E.1, y el nuevo sistema aparecerá en la columna izquierda donde se podrá desplegar la opción System Configuration para empezar a crear los dispositivos que van dentro de él.

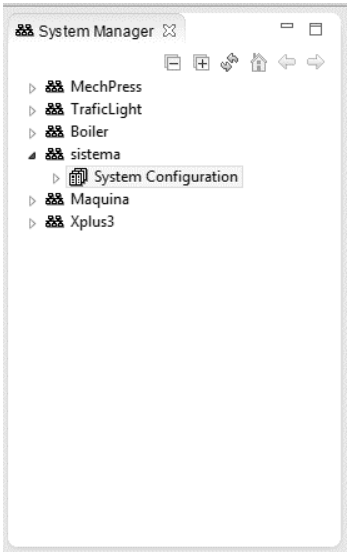


Figura E.4: Configuración del nuevo sistema.

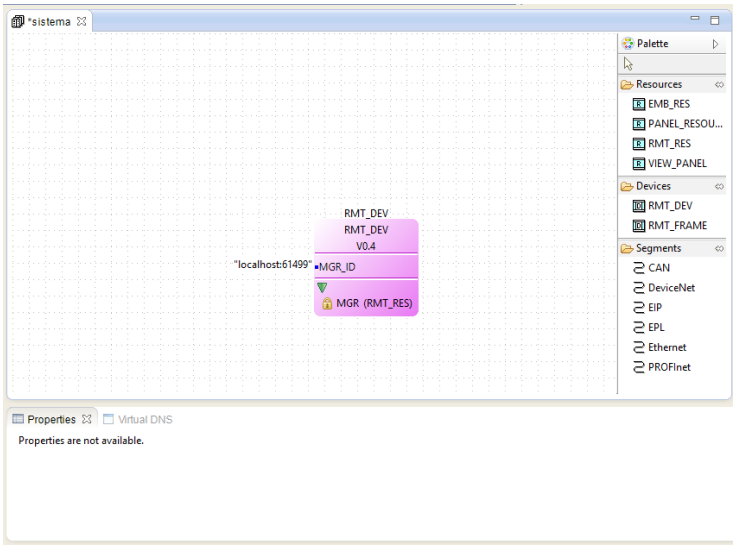


Figura E.5: Configuración del nuevo sistema, central: área de trabajo, derecho: recursos, dispositivos y segmentos de comunicación; inferior: propiedades.

Dentro de System Configuration se podrán crear los dispositivos y los segmentos de comunicación necesarios para comunicarlos arrastrándolos desde el menú de la derecha hasta el área de trabajo, una vez creados los dispositivos se podrán crear dentro de estos los recursos también arrastrándolos hasta la división inferior del bloque de dispositivo (triangulo verde).

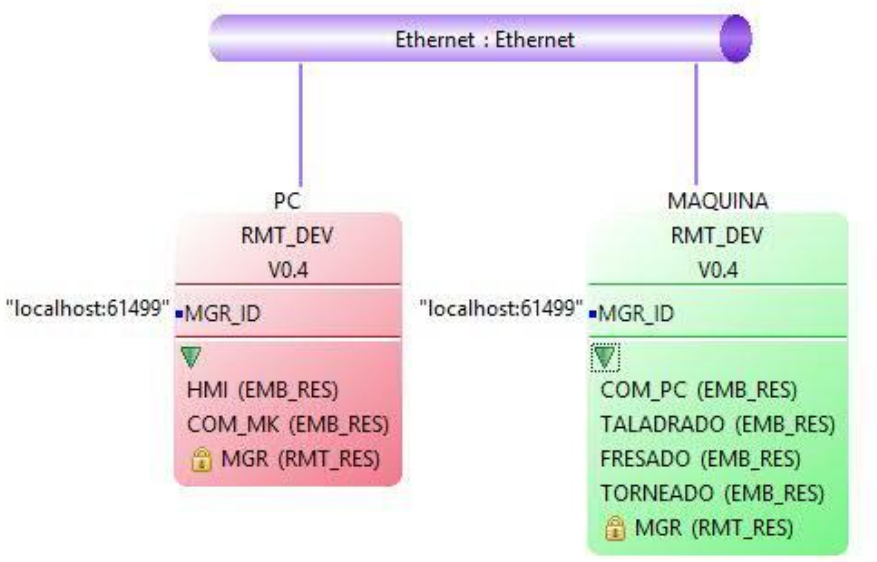


Figura E.6: Dispositivos conectados a segmento de comunicación tipo Ethernet.

Figura E.7: Dispositivo con recursos embebidos (EMB_RES).

Tanto a dispositivos como a recursos se les puede cambiar el nombre seleccionándolos y modificándolo en el área de propiedades (inferior).

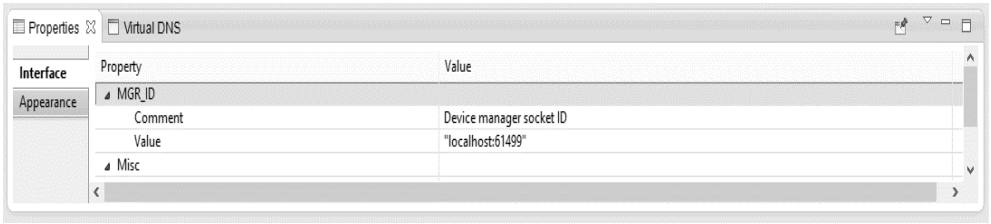


Figura E.8: Propiedades del dispositivo.

Para trabajar dentro de un recurso se abre haciendo doble click sobre él, se desplegará un área de trabajo que trae un bloque START generador de eventos por defecto y se podrán arrastrar dentro de él los bloques que sean necesarios desde la biblioteca de FB.

Figura E.9: Área de trabajo de los recursos

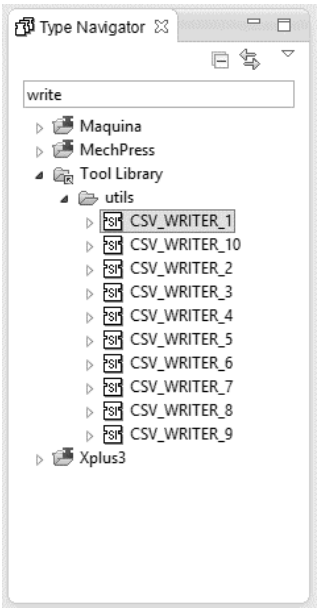


Figura E.10: Biblioteca de FB, para utilizarlos se deben seleccionar y arrastrar hasta el espacio de trabajo.

Con los bloques dispuestos en el espacio de trabajo, se procede a su interconexión seleccionando los puertos de cada bloque y arrastrando con click sostenido hasta el de los otros bloques; es importante recordar que los puertos rojos corresponden a eventos y los azules a datos, la conexión solo se creara si ambos puertos coinciden en el tipo de información que manejan y el hilo que se crea entre ambos bloques tiene flechas en sus extremos para indicar la dirección del flujo de información.

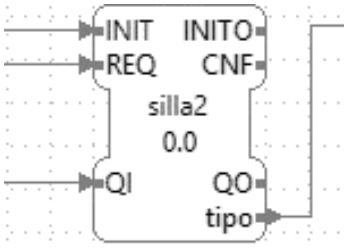


Figura E.11: FB conectado con bloques vecinos.

Cuando una aplicación sea mapeada sobre un recurso, la red de bloques mapeados aparecerá en el espacio de trabajo de este.

E.2 Cómo crear una aplicación

Para crear una aplicación, se procede de igual forma que para crear un nuevo sistema, es decir File-New-New Application. La nueva aplicación creará una ventana de dialogo como la del nuevo sistema, donde podremos nombrarla y relacionarla con el sistema creado; una vez creada aparecerá en el menú del sistema así:

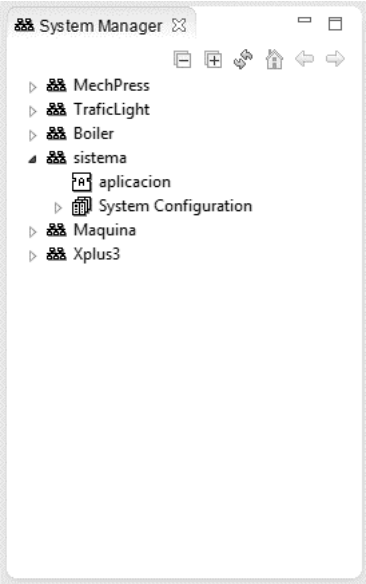


Figura E.12: Nueva aplicación relacionada con el sistema.

Desde el menú del sistema podemos abrir la aplicación para desarrollar la red de FB interna de esta, arrastrándolos al espacio de trabajo desde la biblioteca de FB (derecha), es importante recordar que solo se podrán utilizar los bloques que estén relacionados con la biblioteca del proyecto.

Cuando la red de bloques ha sido creada se procede a mapear la aplicación en los recursos que sean convenientes, para hacer una aplicación distribuida se pueden mapear partes de ellas en un dispositivo y otras en otros. Para mapear una aplicación se procede a seleccionar el o los bloques que se desean mapear y con el click izquierdo se despliega un menú, se selecciona la opción Hardware Mapping y se escoge el recurso seleccionado, los bloques mapeados adquirirán el color del dispositivo donde se encuentra el recurso en el que fueron mapeados y de igual forma aparecerán en ese recurso para que puedan ser utilizados desde allá.

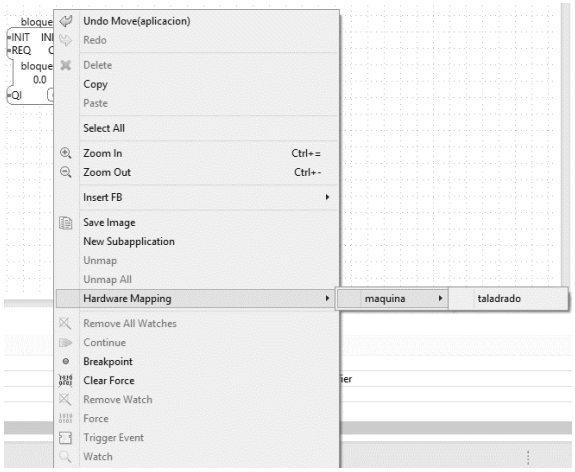


Figura E.13: Menú para mapear FB de una aplicación en un recurso.

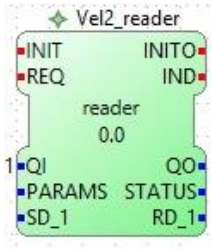


Figura E.14: FB de aplicación mapeado en un recurso.

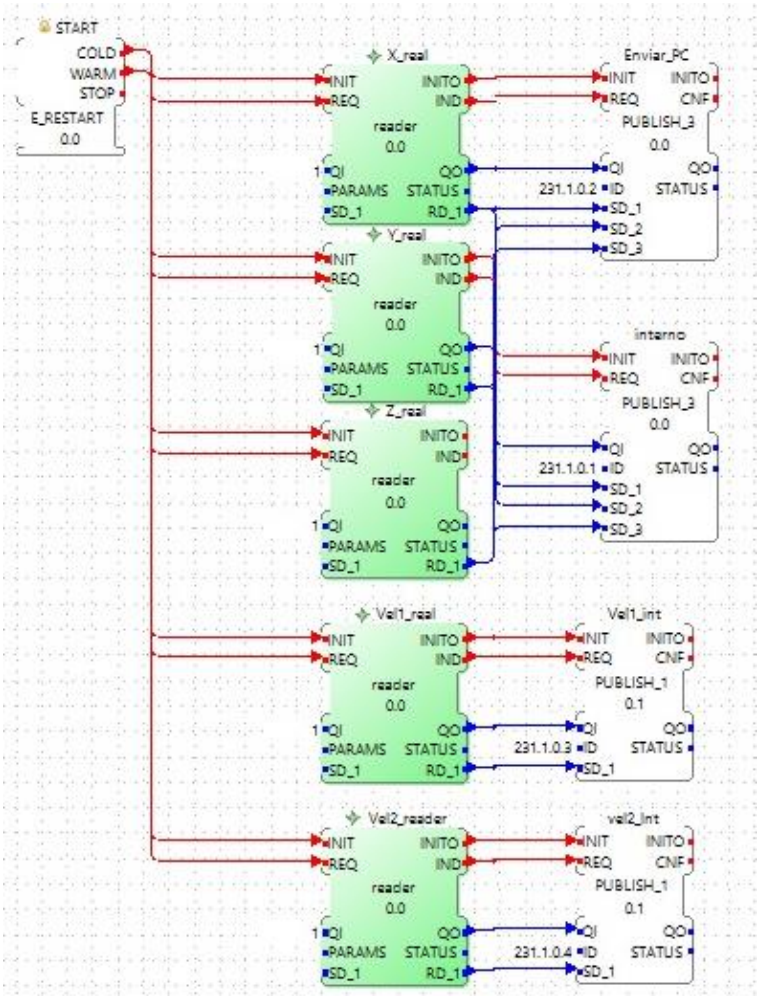


Figura E.15: Recurso datos_mk con FB mapeados desde una aplicación.

E.3 Cómo crear un FB

Para desarrollar bloques de función propios se siguen los mismos pasos para la creación de un sistema o aplicación pero en este caso se selecciona la opción New Type, se abrirá una ventana en la cual se puede relacionar el bloque con el sistema y seleccionar una carpeta en la biblioteca de FB para que lo contenga, además se podrá nombrar el bloque y escoger el tipo de bloque (Básico, de interfaz de servicio, compuesto o Subaplicación).

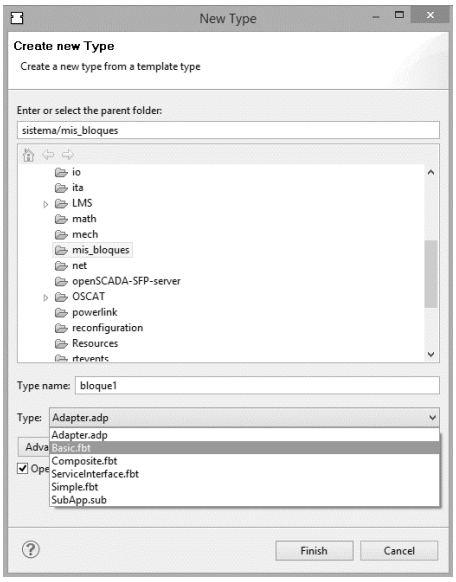


Figura E.16: Configuración de un nuevo FB.

Con todos los parametros seleccionados, se da click en finish y se abrirá la interfaz de trabajo del nuevo bloque en área de trabajo.

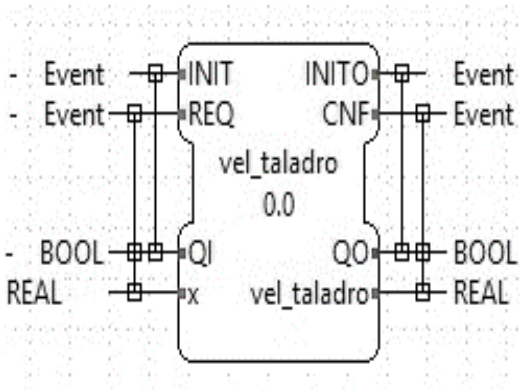


Figura E.17: Trabajo en un nuevo FB.

La interfaz del nuevo bloque trae los parámetros básicos como son eventos de inicio y datos de entrada y salida, sin embargo se podrán crear, destruir y renombrar estas entradas o salidas, es importante que todas las entradas o salidas de datos estén asociadas a los eventos correspondientes con los que interactúan, esto se hace con click sostenido arrastrándolo entre ellos o en la parte inferior (visor de propiedades). En la parte inferior del área de trabajo aparecen varias pestañas a las que se puede acceder para programar el comportamiento del bloque.

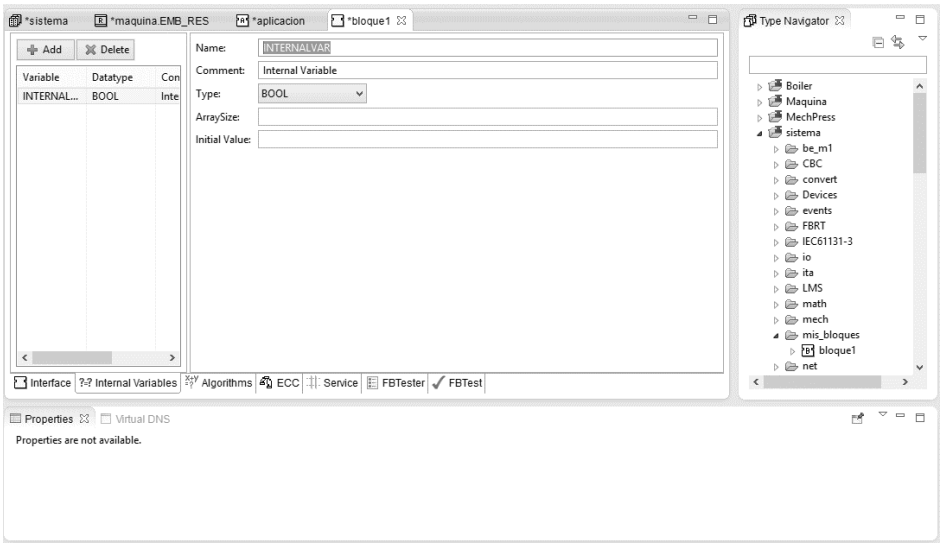


Figura E.18: Pestaña Internal Variables, en esta pestaña se configuran las variables internas de trabajo del bloque, su nombre y el tipo de variable.

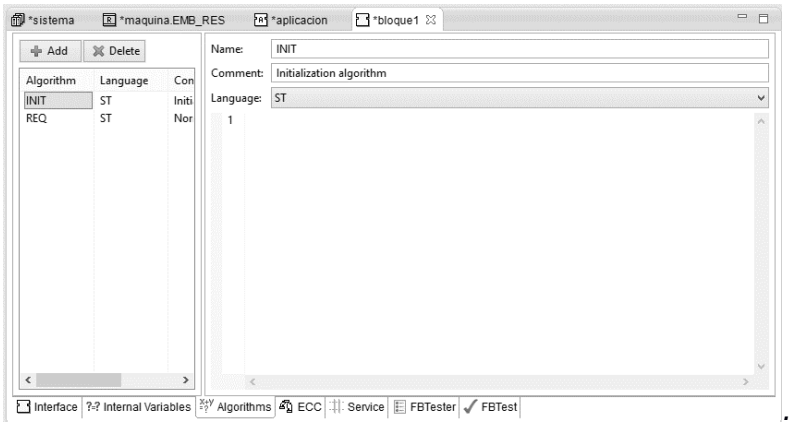


Figura E.19: Pestaña Algorithms, en esta pestaña se escribe el algoritmo interno del bloque relacionándolo con el evento que servirá de iniciador del algoritmo, se puede configurar el nombre del algoritmo y el lenguaje en que va a ser escrito.

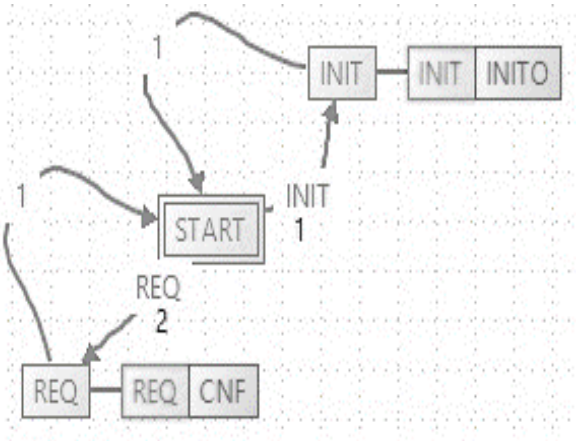


Figura E.20: Pestaña ECC, en esta pestaña se organiza la secuencia de ejecución del bloque, relacionando los eventos con los algoritmos; tiene símil con una red de Petri.

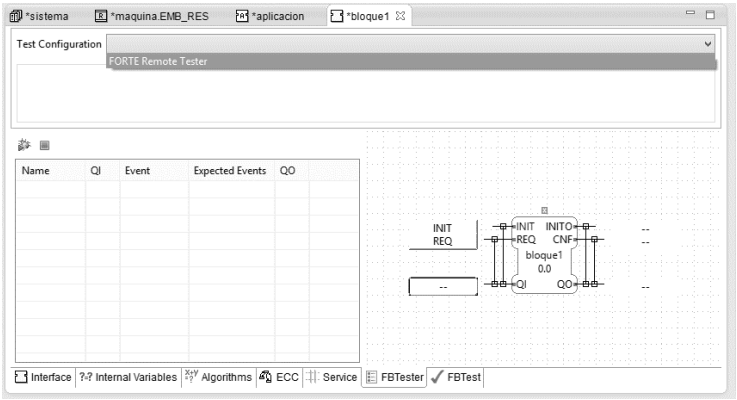


Figura E.21: Pestaña FBTester, en esta pestaña se puede hacer pruebas para comprobar que el bloque funciona correctamente.

E.4 Cómo crear una subaplicación o bloque compuesto

La creación de subaplicaciones y bloques compuestos es similar, se efectúan los mismos pasos que para la creación de un FB normal pero en la opción tipo se selecciona ya sea Composite o SubApp.

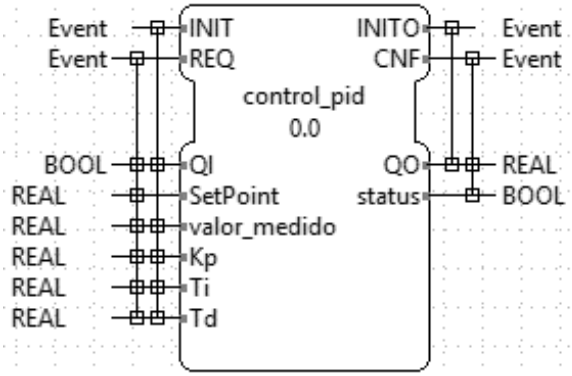


Figura E.22: Interfaz de subaplicación con red interna de FB.

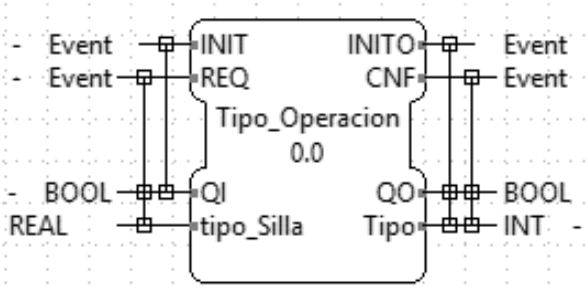


Figura E.23: Interfaz de bloque compuesto.

Del mismo modo que en un FB regular, se pueden crear nuevos eventos y datos de entrada y salida, en las pestañas de abajo ofrecen la función de FBTester igual que en un FB normal, sin embargo ahora no existen pestañas de variables, algoritmos ni ECC; en su lugar ofrecen pestañas de redes de bloques: SubApplication Network y Composite Network respectivamente, en estas pestañas se podrá crear redes internas que regulen el funcionamiento de esas subaplicaciones o bloques compuestos y que funcionan de la misma forma en que lo hace una red de bloques en cualquier otro lugar del sistema, los datos y eventos tanto de entrada como de salida aparecen en el espacio de estas redes internas para que sean utilizados por ellas, así:

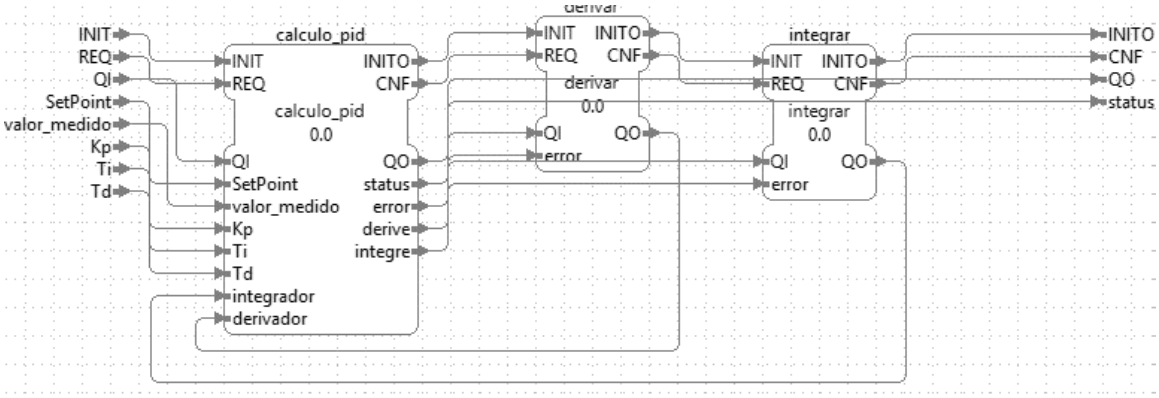


Figura E.24: Espacio de trabajo para la red interna de la subaplicación figura E.22.

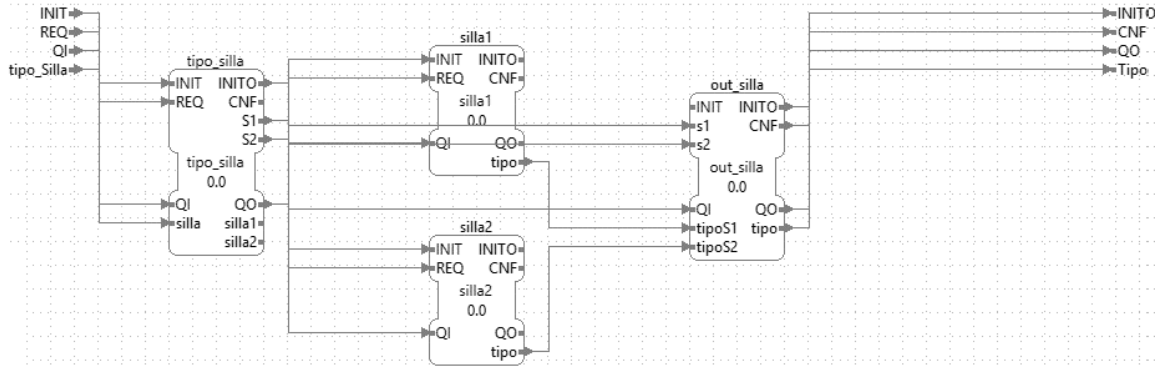


Figura E.25: Espacio de trabajo para la red interna del bloque compuesto figura E.23.

Anexo F: Representación del modelo del Sistema mediante redes de petri

F.1 Modelo de aplicación

- **Aplicación app_datosPc**

La figura F.1 muestra la aplicación encargada de los datos que entran al PC, a la derecha está la red de bloques realizada en 4DIAC-IDE y a la izquierda se encuentra la red de Petri que interpreta su comportamiento; además, la tabla F.1 explica los eventos que componen la red con su equivalente en los FB y su función.

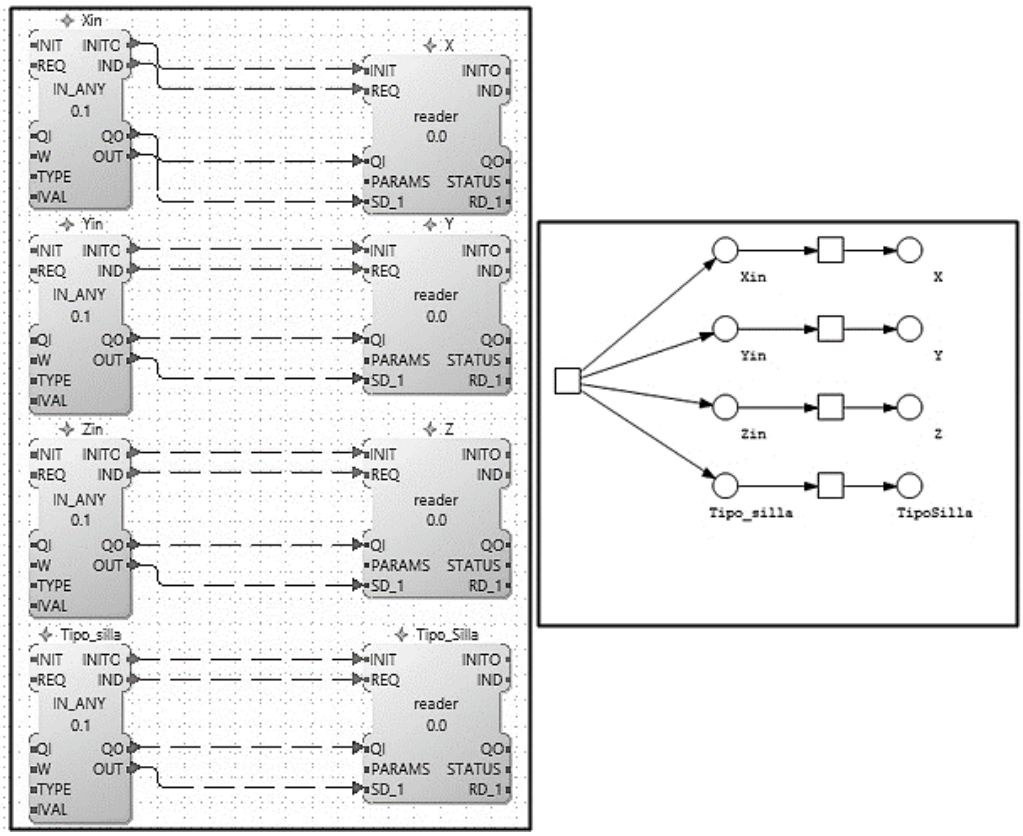


Figura F.1: modelo en red de Petri de la aplicación app_datosPc en paralelo con su modelo en 4DIAC.

Lugar	Tipo de FB	Función
Xin	IN_ANY	Permite la entrada de un dato desde el exterior del sistema, en este caso la coordenada X.
Yin	IN_ANY	Permite la entrada de un dato desde el exterior del sistema, en este caso la coordenada Y.
Zin	IN_ANY	Permite la entrada de un dato desde el exterior del sistema, en este caso la coordenada Z.
tipo_silla	IN_ANY	Permite la entrada de un dato desde el exterior del sistema, en este caso el modelo de la silla.
X	Reader	Lee el dato que es recibido en el bloque anterior, en este caso X.
Y	Reader	Lee el dato que es recibido en el bloque anterior, en este caso Y.
Z	Reader	Lee el dato que es recibido en el bloque anterior, en este caso Z.
TipoSilla	Reader	Lee el dato que es recibido en el bloque anterior, en este caso el modelo de la silla deseado.
Transiciones	Eventos	Representan a los eventos INITO y CNF, los cuales ocurren al finalizar la ejecución de los FBs y permiten la activación de los siguientes

Tabla F.1: traducción de los FB del modelo en 4DIAC a los lugares de la red de Petri para la figura F.1.

- **Aplicación app_control_punta**

En la figura F.2 se presenta la aplicación encargada del control de la punta que actúa sobre la pieza de trabajo, a la derecha está la red de bloques realizada en 4DIAC-IDE y a la izquierda se encuentra la red de Petri que interpreta su comportamiento; adicionalmente, se tiene la tabla F.2 donde se exponen los eventos que componen la red con su equivalente en los FB y su función.

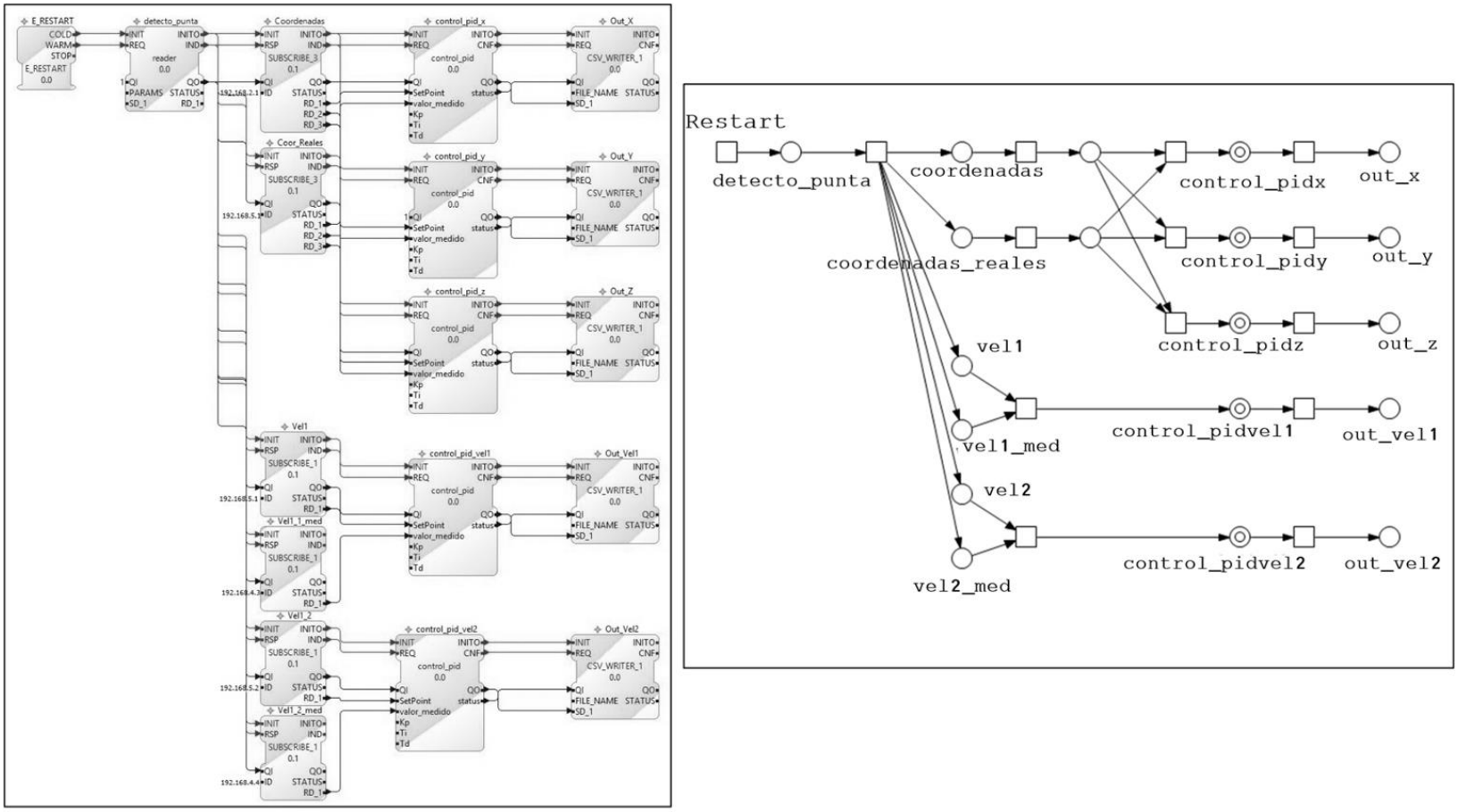


Figura F.2: modelo en red de Petri de la aplicación app_control_punta en paralelo con su modelo en 4DIAC

Lugar	Tipo de FB	Función
detecto_punta	Reader	Lee la señal del sensor que detecta la presencia de la punta de trabajo en la máquina.
coordenadas	Suscriptor	Recibe las coordenadas de la pieza desde el PC.
coordenadas_reales	Suscriptor	Recibe las coordenadas de la pieza medidas desde el área de trabajo.
vel1	Suscriptor	Recibe el dato de la velocidad calculada para el motor 1.
vel1_med	Suscriptor	Recibe el dato de la velocidad medida desde el motor 1.
vel2	Suscriptor	Recibe el dato de la velocidad calculada para el motor 2.
vel2_med	Suscriptor	Recibe el dato de la velocidad medida desde el motor 2.
control_pid x,y,z,vely y vel2	Bloque compuesto	Realiza el cálculo de un controlador PID y genera el esfuerzo de control.
out_x,y,z,vel1 y vel2	Writer	Escribe el esfuerzo de control en las salidas físicas del dispositivo.
Transiciones	Eventos	Representan a los eventos INITO y CNF, los cuales ocurren al finalizar la ejecución de los FBs y permiten la activación de los siguientes

Tabla F.2: traducción de los FB del modelo en 4DIAC a los lugares de la red de Petri para la figura F.2.

- **Aplicación app_Reconfiguracion**

La figura F.3 muestra la aplicación encargada de la reconfiguración del sistema, en la parte superior de la imagen está la red de bloques realizada en 4DIAC-IDE y abajo se encuentra la red de Petri que interpreta el comportamiento de estos; al mismo tiempo, se tiene la tabla F.3 explicando los eventos que componen la red con su equivalente en los FB y su función.

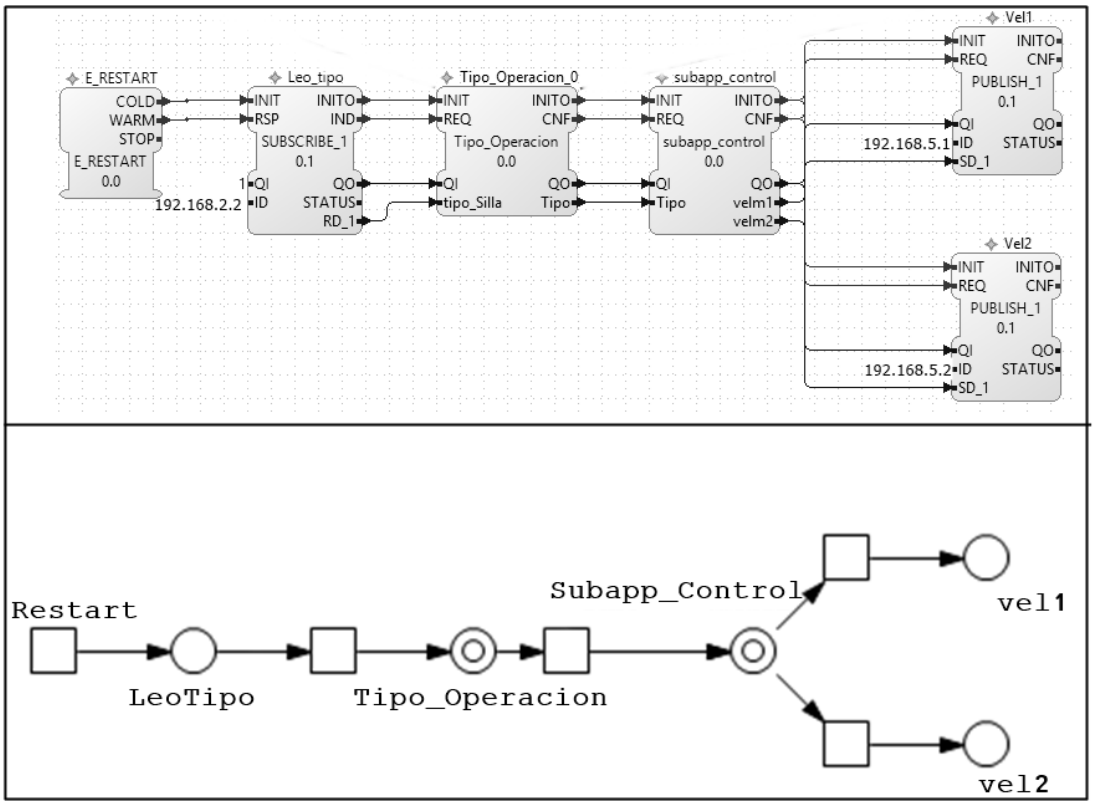


Figura F.3: modelo en red de Petri de la aplicación app_Reconfiguración en paralelo con su modelo en 4DIAC

Lugar	Tipo de FB	Función
Restart	FB de eventos	Inicializa la ejecución de la red.
LeoTipo	Suscriptor	Recibe el modelo de silla deseado desde el PC.
Tipo_Operacion	FB compuesto	Contiene la red que convierte el tipo de silla en la secuencia de tareas de mecanizado para fabricarlas.
Subapp_control	Subaplicación	Calcula las velocidades de operación de los motores de acuerdo a las tareas de mecanizado requeridas.
vel1	Publicador	Envía el dato de la velocidad calculada para el motor 1 hacia el resto del sistema.
vel2	Publicador	Envía el dato de la velocidad calculada para el motor 2 hacia el resto del sistema.
Transiciones	Eventos	Representan a los eventos INITO y CNF, los cuales ocurren al finalizar la ejecución de los FBs y permiten la activación de los siguientes.

Tabla F.3: traducción de los FB del modelo en 4DIAC a los lugares de la red de Petri para la figura F.3.

F.2 Modelo de recurso

- **Recurso: HMI**

La figura F.4 muestra el recurso HMI de la máquina caso de estudio, arriba se pueden observar los bloques que lo componen diseñado en 4DIAC-IDE y en la parte inferior se encuentra la red de Petri que interpreta su comportamiento; además, la tabla F.4 explica los eventos que componen la red con su equivalente en los FB y su función.

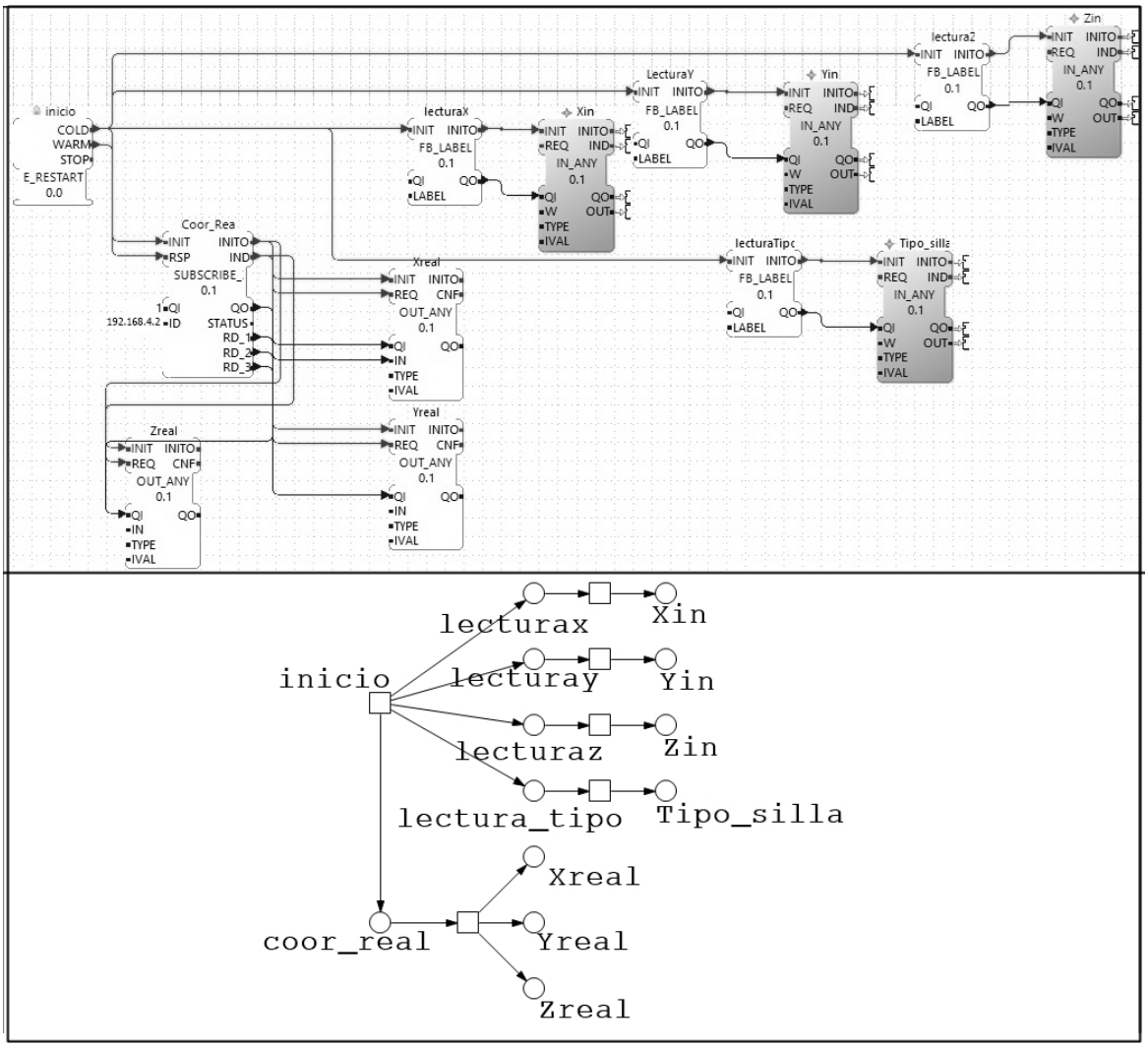


Figura F.4: modelo en red de Petri del HMI en paralelo con su modelo en 4DIAC.

Lugar	Tipo de FB	Función
Inicio	FB de eventos	Inicializa la ejecución de la red.
coor_real	Suscriptor	Recibe las coordenadas de la pieza medidas desde la máquina.
lectura x,y,z y tipo	FB_LABEL	El cual muestra las etiquitas en el HMI para que el usuario identifique cada dato.
x, y y *z real*	OUT_ANY	Imprime los datos hacia las salidas del HMI.
x,y y z in	IN_ANY	Permite el ingreso del modelo de silla deseado desde el HMI.
Tipo_silla	IN_ANY	Permite el ingreso de esos datos desde el HMI.
Transiciones	Eventos	Representan a los eventos INITO y CNF, que finalizan y activan los FBs al ser activados.

Tabla F.4: traducción de los FB del modelo en 4DIAC a los lugares de la red de Petri para la figura F.4.

- **Recurso: comunicación del PC hacia la máquina COM_MK**

La figura F.5 muestra el recurso encargado de comunicar al PC con la máquina, a la derecha muestran los bloques hechos en 4DIAC-IDE y a la izquierda está la red de Petri que representa este recurso; conjuntamente, la tabla F.5 explica los eventos que hacen parte de la red con su equivalente en los FB y su función.

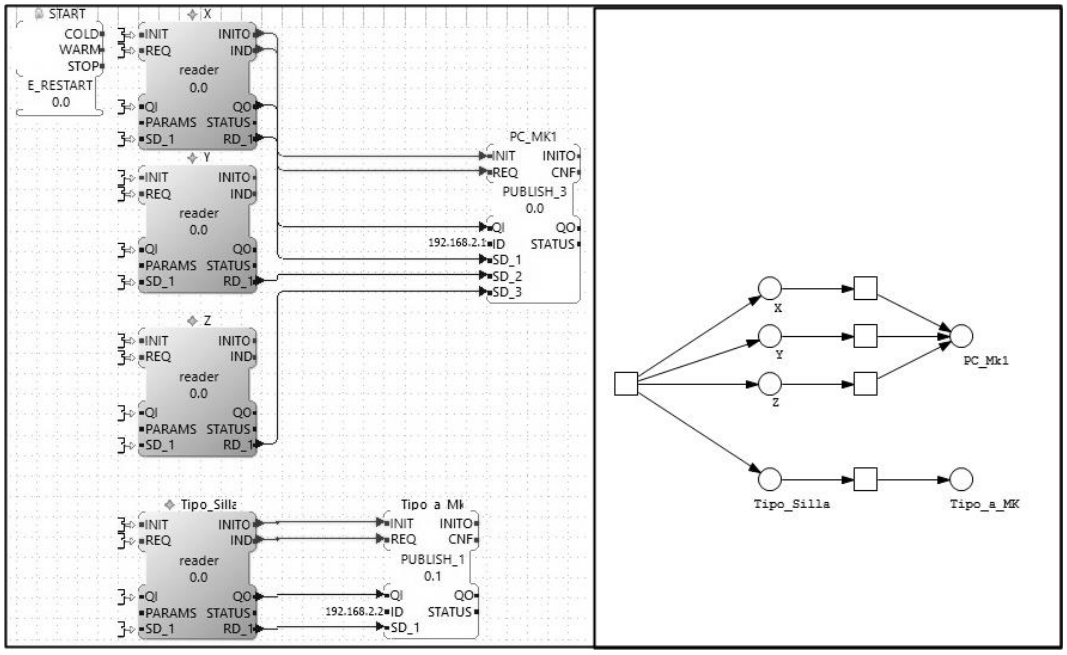


Figura F.5: modelo en red de Petri del recurso COM_MK en paralelo con su modelo en 4DIAC.

Lugar	Tipo de FB	Función
X, Y y Z	Reader	Lee las coordenadas X, Y y Z desde el HMI, gracias al mapeo distribuido de aplicaciones.
Tipo_Silla	Reader	Recibe las coordenadas de la pieza medidas desde la máquina.
Tipo_a_MK	Publicador	Envía los datos de las coordenadas hacia el dispositivo MAQUINA.
PC_MK1	Publicador	Envía el dato del modelo de silla deseado hacia el dispositivo MAQUINA.
Transiciones	Eventos	Representan a los eventos INITO y CNF, los cuales ocurren al finalizar la ejecución de los FBs y permiten la activación de los siguientes.

Tabla F.5: traducción de los FB del modelo en 4DIAC a los lugares de la red de Petri para la figura F.5.

- **Recurso: comunicación de la máquina hacia el PC COM_PC**

La figura F.6 muestra el recurso encargado de comunicar a la máquina con el PC, allí se ve el modelo en 4DIAC-IDE y su representación en redes de Petri, además, la tabla F.6 explica los eventos que hacen parte de la red con su equivalente en los FB y su función.

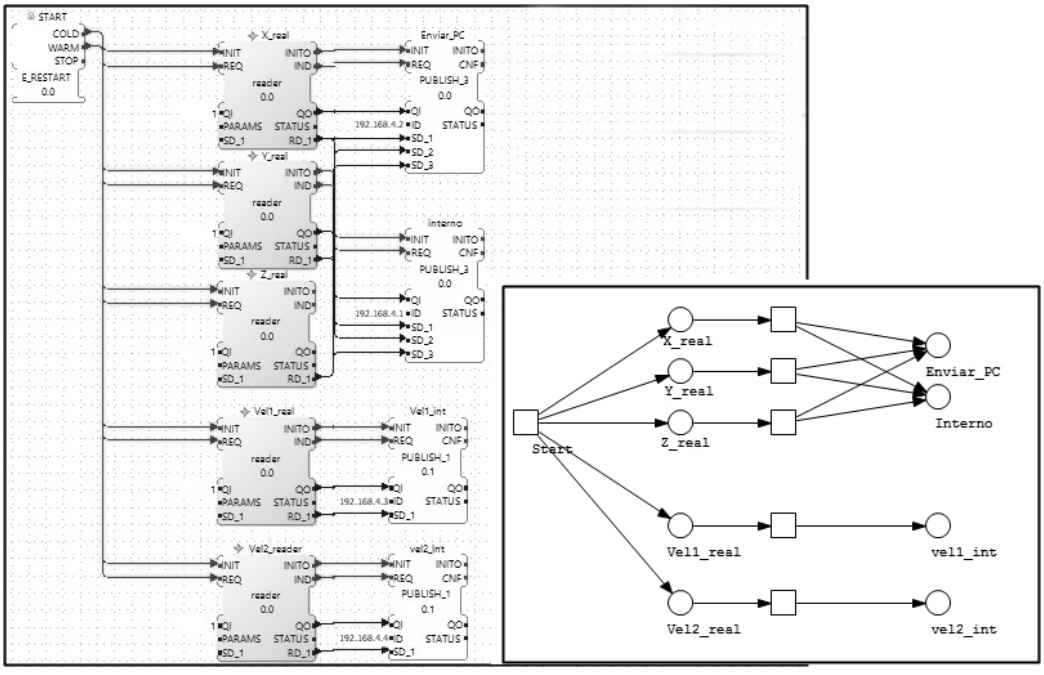


Figura F.6: modelo en red de Petri del recurso COM_PC en paralelo con su modelo en 4DIAC.

Lugar	Tipo de FB	Función
Start	FB de eventos	Inicializa la ejecución de la red.
X, Y y Z real	Reader	Lee las coordenadas de la pieza medidas desde el área de trabajo (mapeado desde la aplicación de datos de la máquina).
Vel1_real	Reader	Lee la velocidad del motor 1 medida desde el área de trabajo (mapeado desde la aplicación de datos de la máquina).
Vel2_real	Reader	Lee la velocidad del motor 2 medida desde el área de trabajo (mapeado desde la aplicación de datos de la máquina).
Enviar_PC	Publicador	Envía las coordenadas X, Y y Z desde el área de trabajo hacia el dispositivo PC.
Interno	Publicador	Envía las coordenadas X, Y y Z desde el área de trabajo hacia otras partes del sistema donde son necesarias.
vel1_int	Publicador	Envía la velocidad del motor 1 medida desde el área de trabajo hacia otras partes del sistema donde son necesarias.
vel2_int	Publicador	Envía la velocidad del motor 2 medida desde el área de trabajo hacia otras partes del sistema donde son necesarias.
Transiciones	Eventos	Representan a los eventos INITO y CNF, los cuales ocurren al finalizar la ejecución de los FB's y permiten la activación de los siguientes eventos.

Tabla F.6: traducción de los FB del modelo en 4DIAC a los lugares de la red de Petri para la figura F.6.

- **Recurso: TALADRADO**

La figura F.7 muestra el recurso para la operación de mecanizado taladrado, en la parte superior se puede observar el modelo en 4DIAC-IDE y abajo la red de Petri; para una mejor interpretación se presenta la tabla F.7 explicando los eventos que hacen parte de la red, su equivalente en los FB y su función.

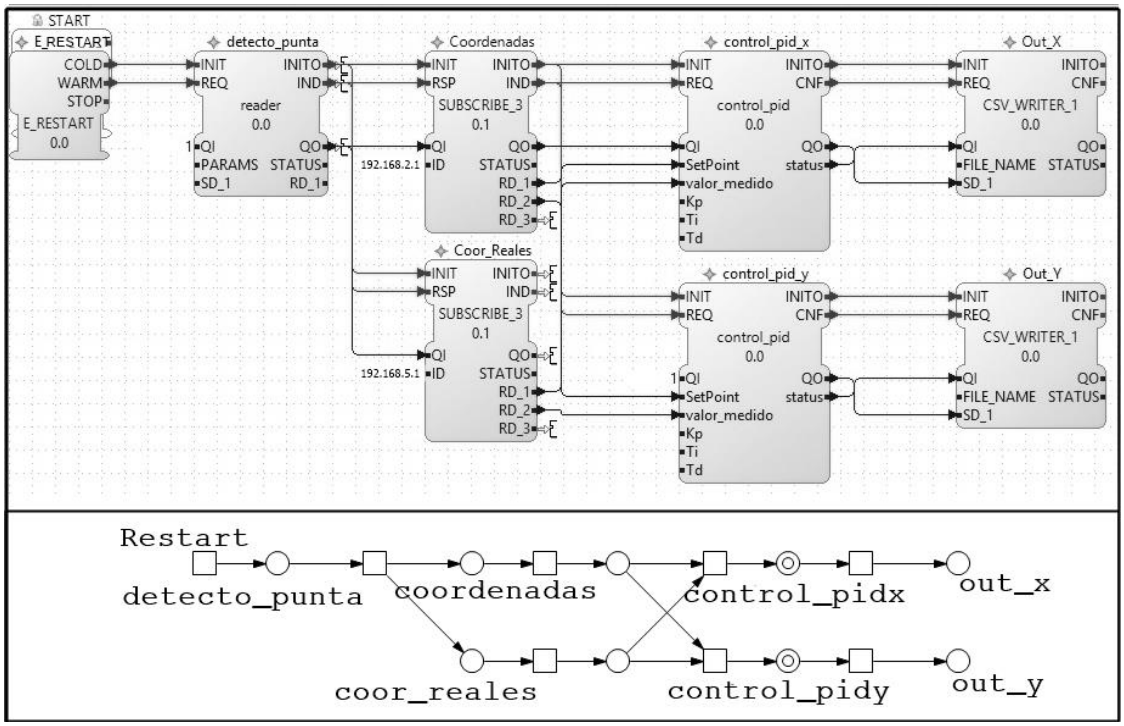


Figura F.7: modelo en red de Petri del recurso TALADRADO en paralelo con su modelo en 4DIAC.

Lugar	Tipo de FB	Función
Restart	FB de eventos	Inicializa la red de FB's.
detecto_punta	Reader	Lee la señal del sensor que detecta la presencia de la punta de trabajo en la máquina (mapeado desde aplicación control_punta).
coordenadas	Suscriptor	Recibe las coordenadas de la pieza desde el PC (mapeado desde aplicación control_punta).
coordenadas_reales	Suscriptor	Recibe las coordenadas de la pieza medidas desde el área de trabajo (mapeado desde aplicación control_punta).
control_pid x,y	Bloque compuesto	Realiza el cálculo de un controlador PID y genera el esfuerzo de control (mapeado desde aplicación control_punta).
out_x,y	Writer	Escribe el esfuerzo de control en las salidas físicas del dispositivo (mapeado desde aplicación control_punta).
Transiciones	Eventos	Representan a los eventos INITO y CNF, los cuales ocurren al finalizar la ejecución de los FB's y permiten la activación de los siguientes

Tabla F.7: traducción de los FB del modelo en 4DIAC a los lugares de la red de Petri para la figura F.7.

- **Recurso: FRESADO**

La figura F.8 muestra el recurso fresado, con la red de bloques y su equivalente en redes de Petri; y la tabla F.8 presenta los eventos que hacen parte de la red de Petri, su equivalente en los FB y su función.

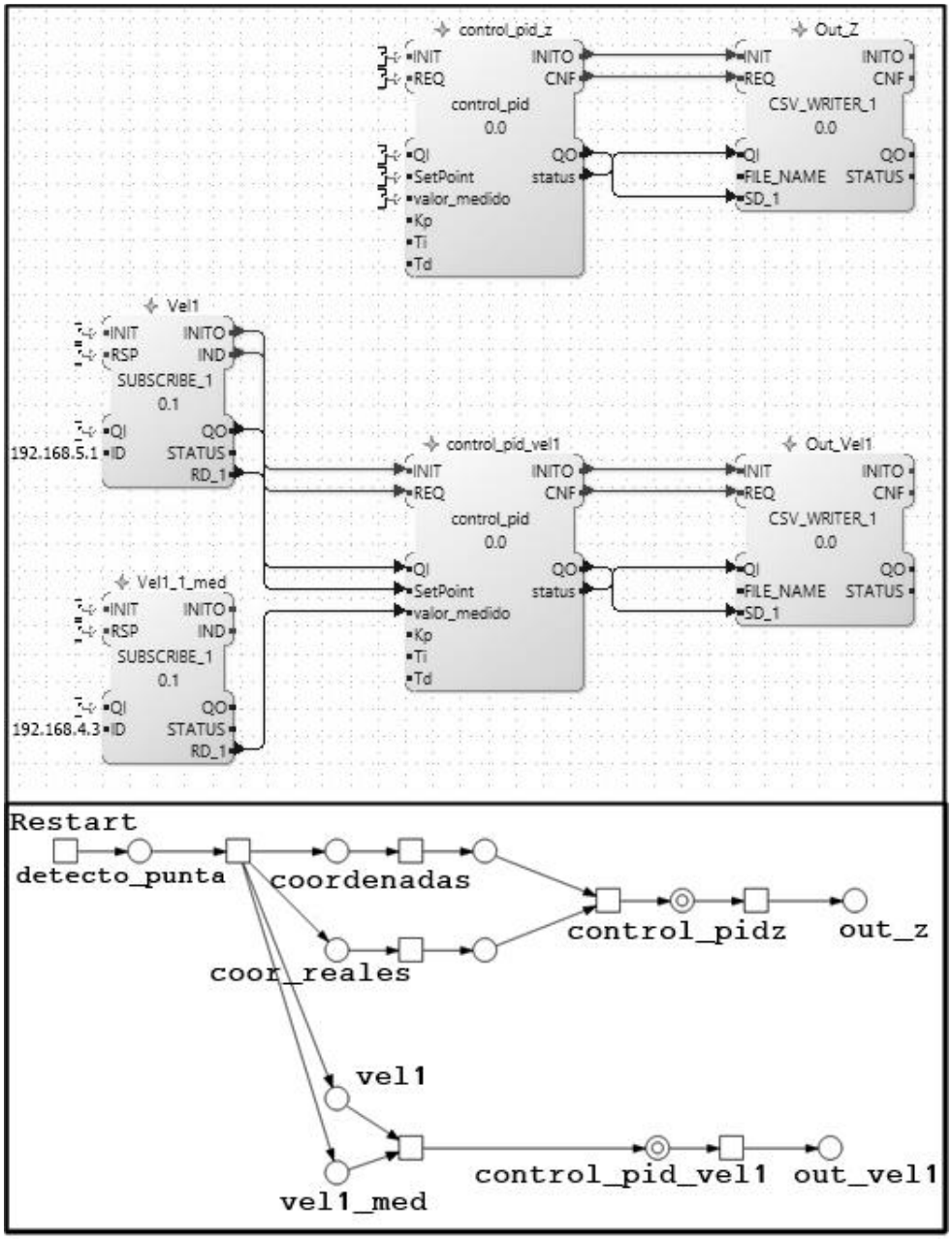


Figura F.8: modelo en red de Petri del recurso FRESADO en paralelo con su modelo en 4DIAC.

Lugar	Tipo de FB	Función
Restart	----	Inicializa la red, solo es necesaria en la Red de Petri ya que en 4DIAC está mapeado en el recurso taladrado.
detecto_punta	-----	Lee la señal del sensor que detecta la presencia de la punta de trabajo en la máquina, solo es necesaria en la Red de Petri ya que en 4DIAC está mapeado en el recurso taladrado.
coordenadas	------	Recibe las coordenadas de la pieza desde el PC, solo es necesaria en la Red de Petri ya que en 4DIAC está mapeado en el recurso taladrado.
coordenadas_reales	-----	Recibe las coordenadas de la pieza medidas desde el área de trabajo, solo es necesaria en la Red de Petri ya que en 4DIAC está mapeado en el recurso taladrado.
vel1	Suscriptor	Recibe el dato de la velocidad calculada para el motor 1.
vel1_med	Suscriptor	Recibe el dato de la velocidad medida desde el motor 1.
control_pid z y vel1	Bloque compuesto	Realiza el cálculo de un controlador PID y genera el esfuerzo de control.
out_z y vel1	Writer	Escribe el esfuerzo de control en las salidas físicas del dispositivo.
Transiciones	Eventos	Representan a los eventos INITO y CNF, los cuales ocurren al finalizar la ejecución de los FB's y permiten la activación de los siguientes.

Tabla F.8: traducción de los FB del modelo en 4DIAC a los lugares de la red de Petri para la figura F.8.

- **Recurso: TORNEADO**

La figura F.9 muestra el recurso para la operación de mecanizado taladrado, en la parte superior de la imagen está la red de bloques realizada en 4DIAC-IDE y abajo se encuentra la red de Petri que interpreta el comportamiento de estos; al mismo tiempo, se tiene la tabla F.9 explicando los eventos que componen la red con su equivalente en los FB y su función.

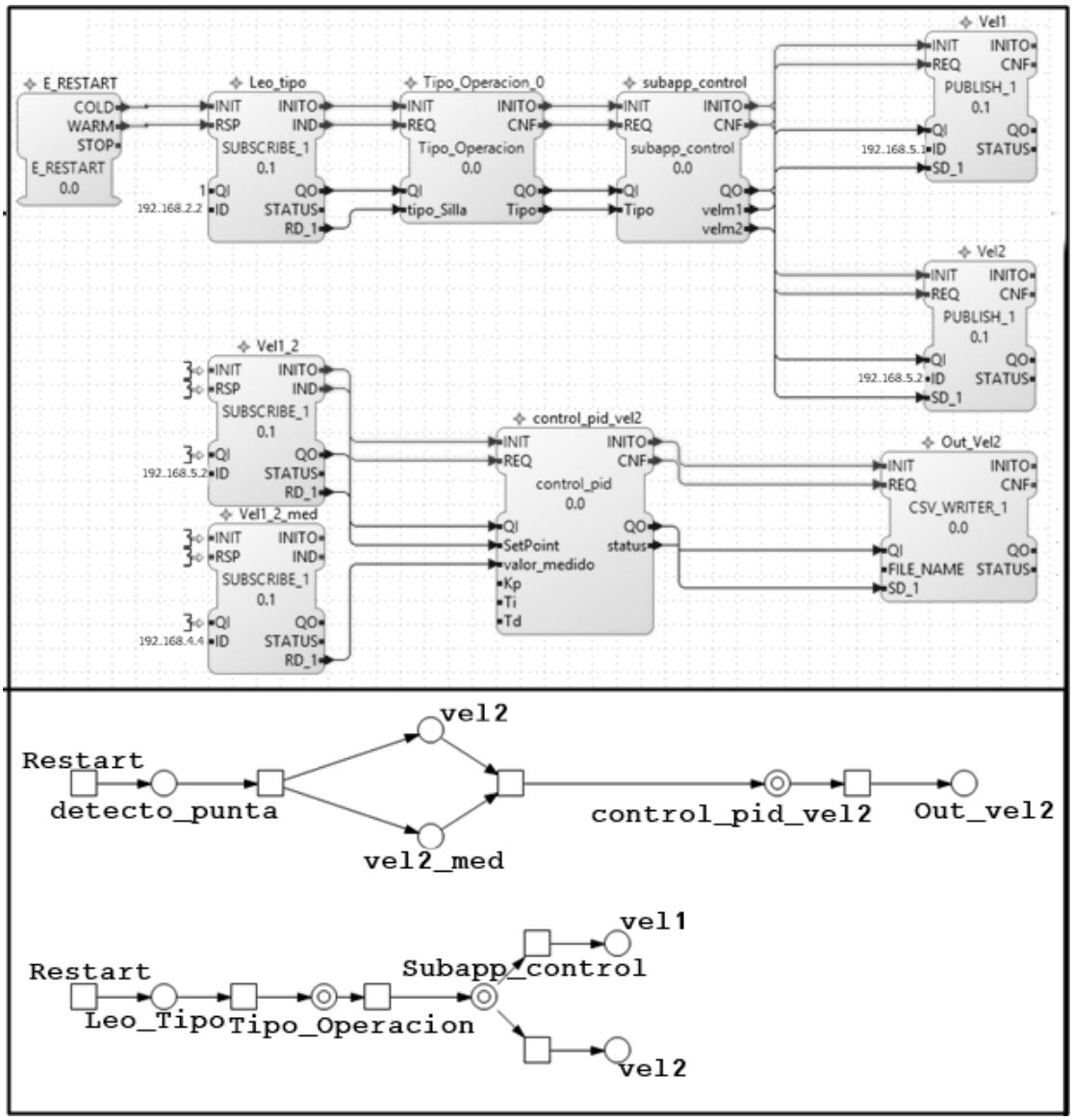


Figura F.9: modelo en red de Petri del recurso TORNEADO en paralelo con su modelo en 4DIAC.

Lugar	Tipo de FB	Función
Restart	-----	Inicializa la red.
detecto_punta	-----	Lee la señal del sensor que detecta la presencia de la punta de trabajo en la máquina, solo es necesaria en la Red de Petri ya que en 4DIAC está mapeado en el recurso taladrado.
coordenadas	-----	Recibe las coordenadas de la pieza desde el PC, solo es necesaria en la Red de Petri ya que en 4DIAC está mapeado en el recurso taladrado.
coordenadas_reales	-----	Recibe las coordenadas de la pieza medidas desde el área de trabajo.
vel2	Suscriptor	Recibe el dato de la velocidad calculada para el motor 2 (mapeado desde control_punta).
vel2_med	Suscriptor	Recibe el dato de la velocidad medida desde el motor 2(mapeado desde control_punta).
control_pidvel2	Bloque compuesto	Realiza el cálculo de un controlador PID para el motor 2 y genera el esfuerzo de control (mapeado desde control_punta).
out_ vel2	Writer	Escribe el esfuerzo de control para el motor 2 en las salidas físicas del dispositivo (mapeado desde control_punta).
LeoTipo	Suscriptor	Recibe el modelo de silla deseado desde el PC (mapeado desde la aplicación de reconfiguración).
Tipo_Operacion	FB compuesto	Contiene la red que convierte el tipo de silla en la secuencia de tareas de mecanizado para fabricarlas.
Subapp_control	Subaplicación	Calcula las velocidades de operación de los motores de acuerdo a las tareas de mecanizado requeridas (mapeado desde la aplicación de reconfiguración).
vel1	Publicador	Envía el dato de la velocidad calculada para el motor 1 hacia el resto del sistema (mapeado desde la aplicación de reconfiguración).
vel2	Publicador	Envía el dato de la velocidad calculada para el motor 2 hacia el resto del sistema.
Transiciones	Eventos	Representan a los eventos INITO y CNF, los cuales ocurren al finalizar la ejecución de los FB's y permiten la activación de los siguientes.

Tabla F.9: traducción de los FB del modelo en 4DIAC a los lugares de la red de Petri para la figura F.9.

F.3 modelo de bloque de función compuesto

- **Bloque compuesto VELM1**

En la figura F.10 puede observar el bloque encargado de la velocidad del motor uno, su modelo realizado en 4DIAC y la red de Petri que equivale a dicho modelo, adicionalmente, está la tabla F.10 explicando mejor los elementos presentes en esa figura.

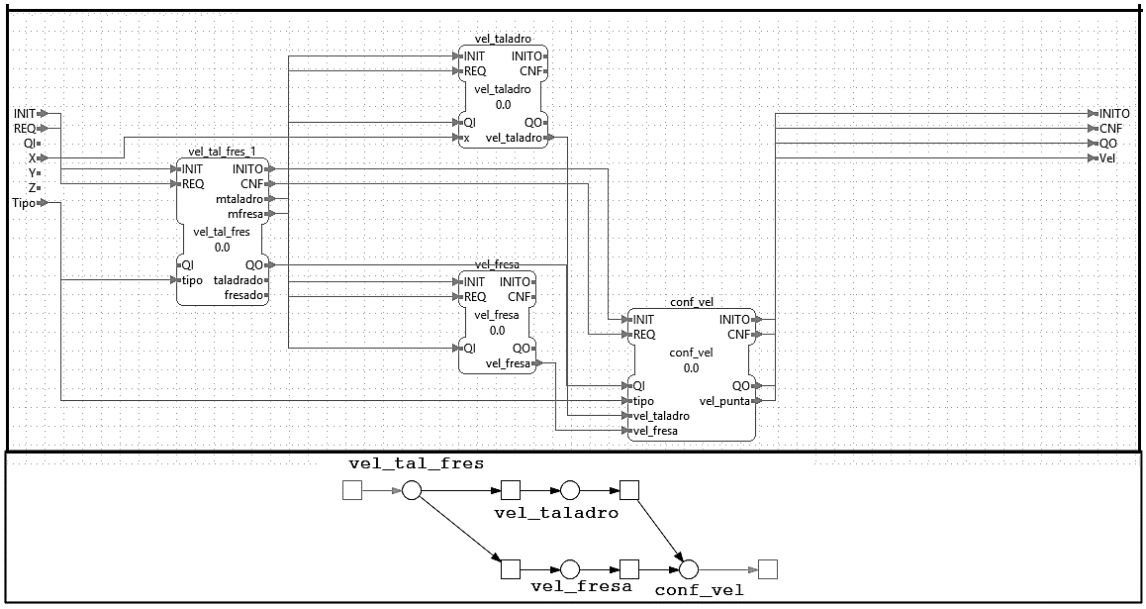


Figura F.10: modelo en red de Petri del bloque compuesto VELM1 en paralelo con su modelo en 4DIAC.

Lugar	Tipo de FB	Función
vel_tal_fres	FB básico	Decide cuál de las tareas que involucran al motor 1 debe actuar, taladrado o fresado.
vel_taladrado	FB básico	Calcula la velocidad de trabajo del motor 1 para la tarea de taladrado.
vel_fresado	FB básico	Calcula la velocidad de trabajo del motor 1 para la tarea de fresado.
conf_vel	FB básico	Decide cuál de las velocidades se debe mostrar.
Transiciones	Eventos	Representan a los eventos INITO y CNF, los cuales ocurren al finalizar la ejecución de los FB's y permiten la activación de los siguientes.

Tabla F.10: traducción de los FB del modelo en 4DIAC a los lugares de la red de Petri para la figura F.10.

- **Bloque compuesto VELM2**

La figura F.11 tiene el bloque programado para el motor 2 utilizado en la operación de torneado, en paralelo con red de Petri en la parte inferior, y la tabla F.11 para mayor información sobre la figura mostrada.

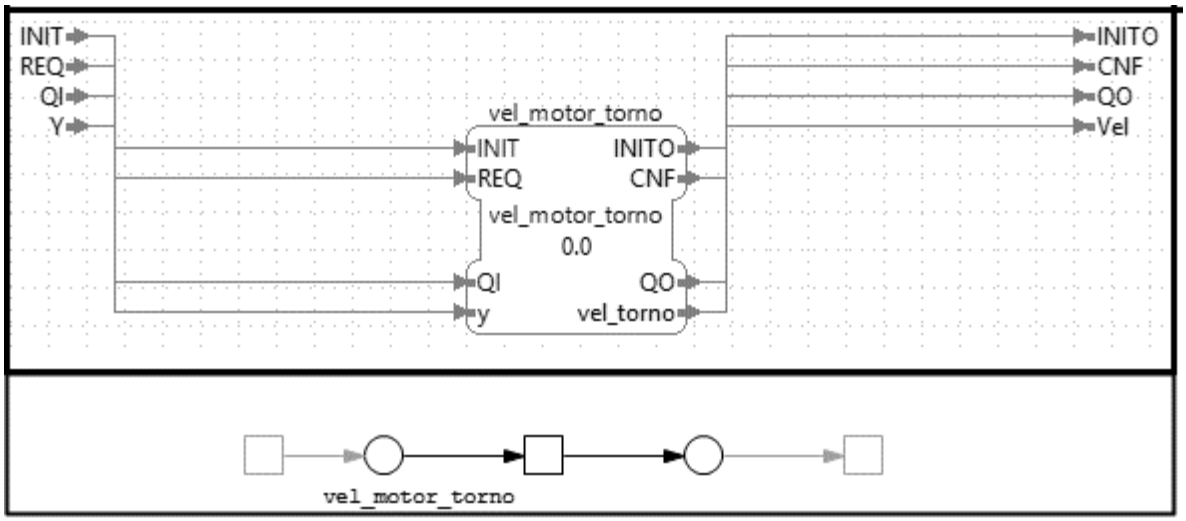


Figura F.11: modelo en red de Petri del bloque compuesto VELM2 en paralelo con su modelo en 4DIAC.

Lugar	Tipo de FB	Función
vel_taladrado	FB básico	Calcula la velocidad de trabajo del motor 2 para la tarea de torneado.
Transiciones	Eventos	Representan a los eventos INITO y CNF, los cuales ocurren al finalizar la ejecución de los FB's y permiten la activación de los siguientes.

Tabla F.11: traducción de los FB del modelo en 4DIAC a los lugares de la red de Petri para la figura F.11.