ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Mecánica y Ciencias de la Producción

"Desarrollo de un modelo de simulación para el secuenciamiento y análisis de capacidad para una planta que fabrica Margarinas"

TESIS DE GRADO

Previo a la Obtención del Título de:

INGENIEROS INDUSTRIALES

Presentada por:

Rodrigo Alejandro Pita Bejarano

Oscar Andrés Ponce Parra

GUAYAQUIL – ECUADOR

Año: 2010

AGRADECIMIENTO

Queremos agradecer principalmente a Dios, por todas las bendiciones que recibimos de Él que nos han permitido finalizar con todo el esfuerzo y ganas en esta etapa de nuestras vidas, a nuestro Padres por su incansable apoyo e insistencia para finalizar este estudio, a todos las personas que han colaborado con sus consejos para poder terminar con éxito nuestro trabajo, a nuestras novias por su apoyo incondicional y preocupación. Finalmente al Ing. Marco Buestán por su esfuerzo, colaboración y por ser nuestro guía ante este reto.

DEDICATORIA

A NUESTROS PADRES

A MI ABUELA LA MAMAMA

A NUESTRAS NOVIAS

A NUESTROS HERMANOS

TRIBUNAL DE GRADUACIÓN

_____

Ing. Francisco Andrade S.
DECANO FIMCP

_____

Ing. Marcos Buestán B.
DIRECTOR DE TESIS

_____

Ing. Iván Silva C.
VOCAL

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesis de Grado, nos corresponden exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

(Reglamento de Graduación de la ESPOL).

_Rodrigo Alejandro Pita Bejarano

Oscar Andrés Ponce Parra

RESUMEN

Hoy en día las Empresas están exigiendo como base para mejorar sus costos y niveles de servicio la visión no solo a corto sino mediano y largo plazo de sus operaciones y cadenas de suministros. Para mantenerse a la par de la demanda y las exigencias del mercado, cuando se trata de la toma decisiones, muchas compañías han dejado a un lado los estudios técnicos previos para darle todo el trabajo a la intuición, corriendo el riesgo que tales decisiones afecten el desempeño de sus operaciones, imagen y nivel de servicio frente a sus clientes.

La presente tesis está basada en una importante empresa multinacional representante de varias categorías en el Ecuador. La empresa ha demostrado un crecimiento y aumento de la demanda en todas sus categorías, en especial las del sector alimenticio que son las más delicadas del negocio. Analizamos las proyecciones de su demanda y como estos influyen en cada una de sus operaciones de los procesos productivos.
Los análisis tradicionales, son demasiados complejos para entregar un horizonte más claro de las operaciones críticas de la empresa, estas por lo general toman demasiado tiempo en realizarse y requieren varias personas con una preparación académica de alto nivel para resolverlos e interpretarlos.
Sin embargo, la simulación es el complemento sobre el cual es posible realizar análisis del corto, mediano y largo plazo de forma rápida, sencilla (fácil de interpretar) y con bajos costos de preparación, integrando una herramienta confiable y sofisticada para la toma de decisiones de una organización.

Para los respectivos análisis e identificación de problemas, riesgos e impactos, se usó un modelo de simulación, el cual se validó y nos permitirá identificar las alternativas de mejora y propuestas orientadas a sustentar el crecimiento de la demanda.

Al concluir este trabajo, se presentaron varias propuestas u opciones que servirán para la toma de decisión de la compañía, se realizó un análisis donde se mostraron los beneficios que la empresa puede lograr implementando las propuestas planteadas.

ÍNDICE GENERAL

Pág.

RESUMEN…… II
ÍNDICE GENERAL……. IV
ABREVIATURAS……. VII
ÍNDICE DE FIGURAS…… VIII
ÍNDICE DE TABLAS..…… X

ÍNDICE DE PLANOS…… XIII
INTRODUCCIÓN…..…….. 1

CAPÍTULO 1
GENERALIDADES……..... 3

1.1 Antecedentes……..…….. 3

1.2 Objetivo General...……. 6

1.3 Objetivos Específicos……...……. 6

1.4 Metodología Utilizada en la Tesis…….. 7

CAPÍTULO 2

2 DESCRIPCIÓN GENERAL DE LA EMPRESA…… 9

2.1 Descripción de los Productos……. 9

2.2 Descripción del Proceso Productivo……..... 11

2.3 Descripción Física de la Planta…….. 16

CAPÍTULO 3 19

3 DESCRIPCIÓN DEL MODELO DE SIMULACIÓN……..

19
3.1 Creación de Entidades…….

21
3.2 Creación de Locaciones……..…….……...

25
3.3 Creación de Macros…….

31
3.4 Creación de Arribos……..……...

31
3.5 Descripción de Simulación con Tanques…..……... 37
3.6 Descripción del Modelo de Simulación…….

CAPÍTULO 4

47
4 VALIDACIÓN DEL MODELO DE SIMULACIÓN…….

47
4.1 Tipo de Simulación……..……

49
4.2 Selección de Variables de Respuesta…..…… 52
4.3 Selección de Indicadores…….

52
4.4 Comparación de Datos entre la Simulación y lo Real…….

CAPÍTULO 5

60
5 PLANTEAMIENTO Y VALIDACIÓN DE CAMBIOS….……...

60
5.1 Modelación de Cambios…..……..……

5.1.1 Evaluación de Diferentes Métodos de Secuenciamiento….. 61
5.1.1.1 Cambios Incorporados en el Modelo de Simulación……. 65
5.1.1.2 Evaluación de Resultados…….. 68
5.1.2 Evaluación de Desempeño de Cada Operación a Largo Plazo……..…… 74
5.1.2.1 Cambios Incorporados en el Modelo de Simulación……. 79
5.1.2.2 Evaluación de Resultados…….. 80

CAPÍTULO 6

6 ANÁLISIS DE LA INVERSIÓN FINANCIERA……..…….. 85

6.1 Cuantificación del Impacto de las Alternativas de Mejora……. 85

6.2 Evaluación de la Inversión Financiera……..…… 89

CAPÍTULO 7

7 CONCLUSIONES Y RECOMENDACIONES……. 92

APÉNDICES

BIBLIOGRAFÍA

ABREVIATURAS

SKU (Stock Keeping Unit) Número de Referencia
MPU Unidad de Proceso de Margarinas
TON Toneladas
VS Versus
TPM (Total Preventive Maintance) Mantenimiento Total Preventivo EM Envasadora
B1 Envasadora
B2 Envasadora
OEE Eficiencia Operacional

ÍNDICE DE FIGURAS

Pág.

Figura 1.1 Venta y proyectados (Ton) de margarinas…….....5
Figura 2.1 Diagrama de flujo de proceso de fabricación……12
Figura 2.2 Diagrama del proceso de fabricación de margarinas…...16
Figura 3.1 Flujo de proceso del modelo de simulación……..…39
Figura 4.1 Diagrama de cajas para la línea B1…….54
Figura 4.2 Diagrama de cajas comparación de lo simulado y planificado B1……...55
Figura 4.3 Diagrama de cajas para la línea B2……...57
Figura 4.4 Diagrama de cajas comparación de lo simulado y planificado B2……...58
Figura 5.1 Secuenciamiento para el 1er modelo propuesto……...66
Figura 5.2 Secuenciamiento para el 2do modelo propuesto……..67
Figura 5.3 Secuenciamiento para el 3er modelo propuesto……..…67
Figura 5.4 Secuenciamiento para el 4to modelo propuesto……...67
Figura 5.5 Secuenciamiento para el 5to modelo propuesto……68
Figura 5.6 Secuenciamiento para el 6to modelo propuesto……68
Figura 5.7 Resultados del desperdicio y costos generados por la famila D……...70
Figura 5.8 Resultados del desperdicio y costos generados por los escenarios de la familia b.……73
Figura 5.9 Resultados de la situación actual para la línea B1……75
Figura 5.10 Resultados de la situación actual para la línea B2 y Em……...77
Figura 5.11 Resultados de la situación actual para todas las líneas de producción.……78
Figura 5.12 Resultados de la situación actual para todas las líneas de producción.……84

ÍNDICE DE TABLAS

Pág.

Tabla 3.1 Entidades del modelo de simulación…….....20
Tabla 3.2 Locaciones del modelo de simulación……..…...22
Tabla 3.3 Macros del modelo de simulación…….....26
Tabla 4.1 Resultado para la pruebas de hipótesis para la línea B1…….53
Tabla 4.2 Comparación de la hora de salida entre el plan real y el plan simulado……56
Tabla 4.3 Resultado para la pruebas de hipótesis para la línea B2……59
Tabla 5.1 Combinación de los cambios de formato entre los Skus……64
Tabla 5.2 Resultados del 1er modelo de secuenciamiento propuesto……69
Tabla 5.3 Resultados del 2do modelo de secuenciamiento propuesto……69
Tabla 5.4 Resultados del 3er modelo de secuenciamiento propuesto……70
Tabla 5.5 Resultados del 4to modelo de secuenciamiento propuesto……71
Tabla 5.6 Resultados del 5to modelo de secuenciamiento propuesto……72
Tabla 5.7 Resultados del 6to modelo de secuenciamiento propuesto……72
Tabla 5.8 Tabla de resultados de la situación actual para la línea B1……75
Tabla 5.9 Tabla de resultados de la situación actual para la línea B2 y Em……..76
Tabla 5.10 Resultados del porcentaje de utilización al 2012 de todas las operaciones del proceso productivo con el escenario actual……..81
Tabla 5.11 Resultados del porcentaje de utilización al 2012 de todas las operaciones del proceso productivo con los cambios incorporados ……..82
Tabla 5.12 Comparación de resultados del porcentaje de utilización de cada operación del proceso productivo entre el escenario actual y el propuesto……...83
Tabla 6.1 Cuantificación del desperdicio generado al no estar secuenciados inmediatamente los skus Bc250 y BC500……..…88
Tabla 6.2 Tabla resumen del costo total para la instalación del kit de 500gr……89
Tabla 6.3 Tabla resumen del costo total para la instalación del kit de 500gr……89
Tabla 6.4 Tabla resumen de los escenarios comparados incluyendo el desperdicio del material exclusivo……90
Tabla 6.5 Tabla de resultados del producto terminado que se dejará de vender por falta de capacidad ……91

ÍNDICE DE PLANOS

Pág.

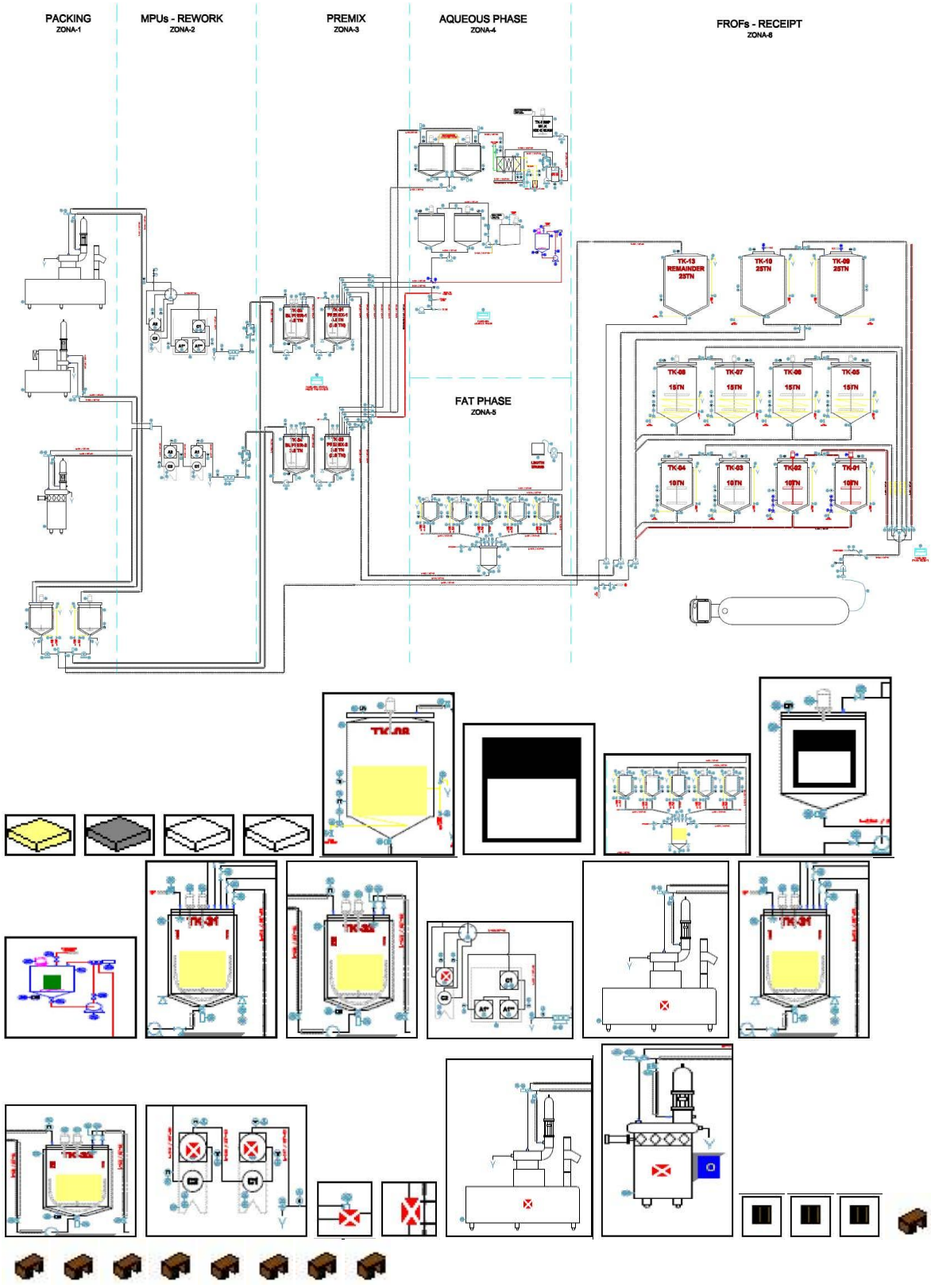
Plano 1.1 Plano Instalaciones de la Fábrica de Margarinas…….....95

INTRODUCCIÓN

No existen muchas técnicas para obtener la utilización que tendrá cada una de las operaciones de un proceso de manufactura proyectando las ventas estimadas.

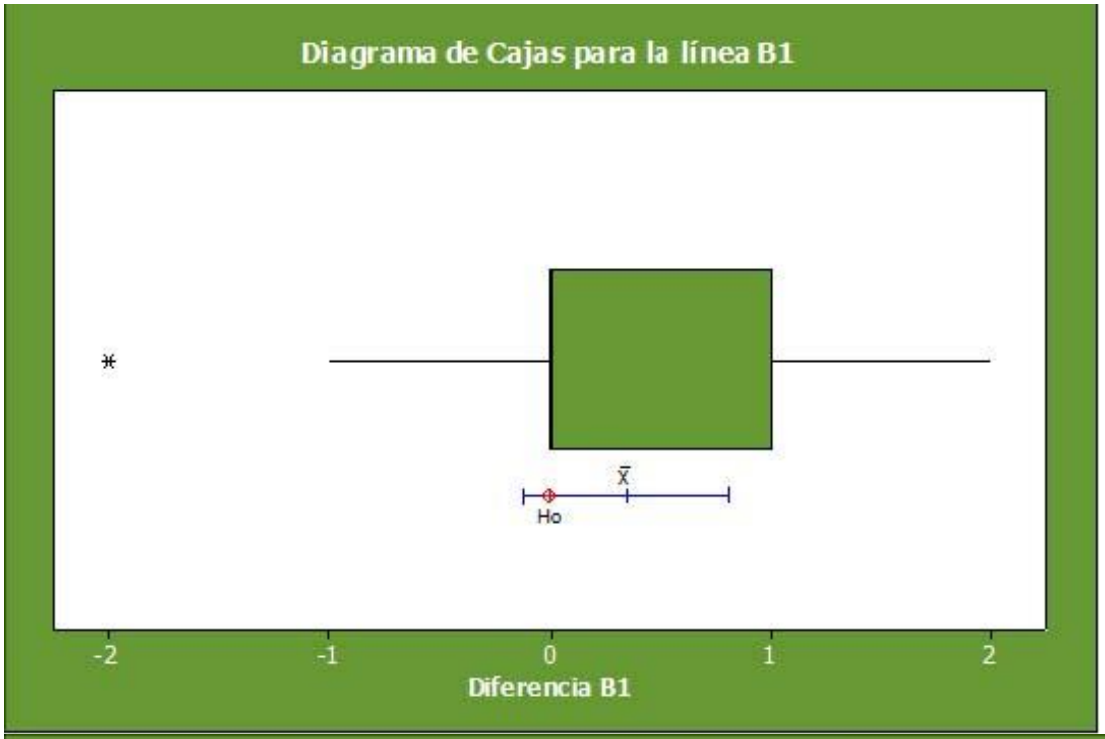
0%
5%
10%
15%
20%
25%
30%
35%
40%
45%
50%
0,0
2500,0
5000,0
7500,0
10000,0
12500,0
15000,0
2007
2008
2009
2010
2011
2012
Estimados (Ton)
Ventas (Ton)
% Crecimiento

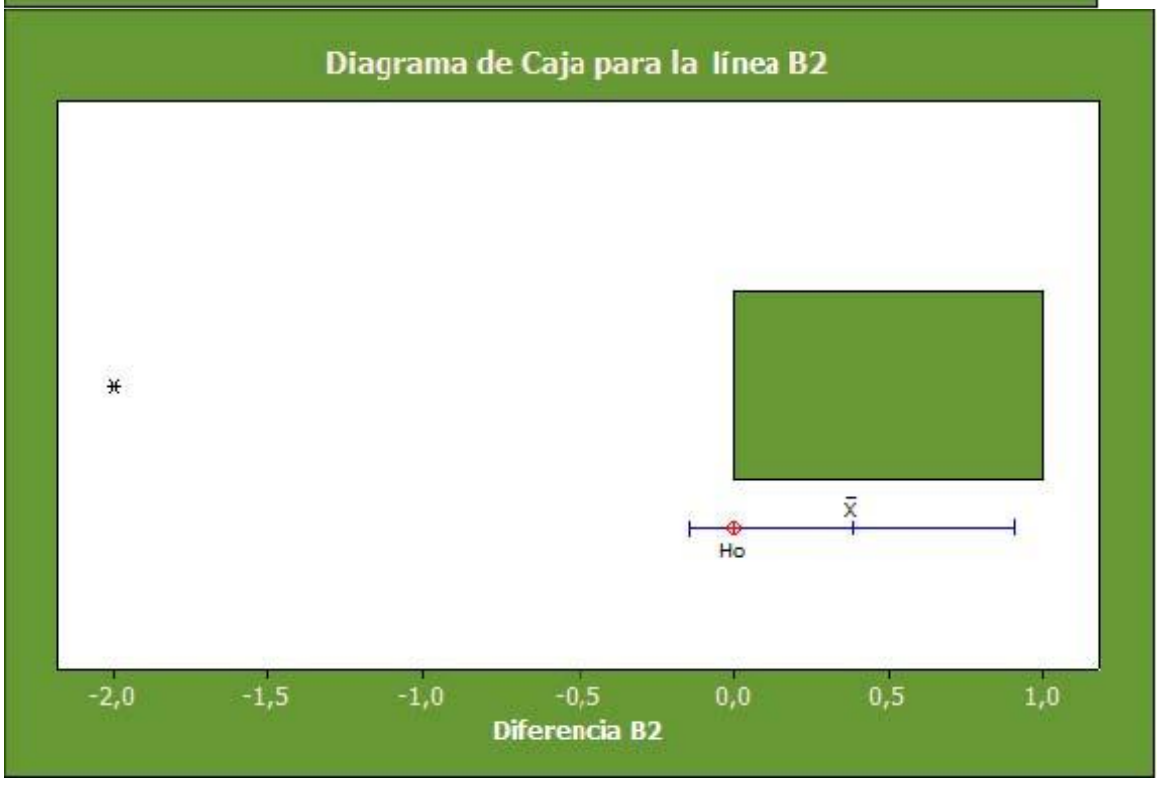
Muchos de los análisis tradicionales se han vuelto obsoletos debido a su complejidad, cantidad de tiempo y recursos necesarios a invertir para obtener lo deseado. Hoy en día existen varias formas que nos permiten obtener la información que la empresa requiere referente a la capacidad de cada una de sus operaciones.

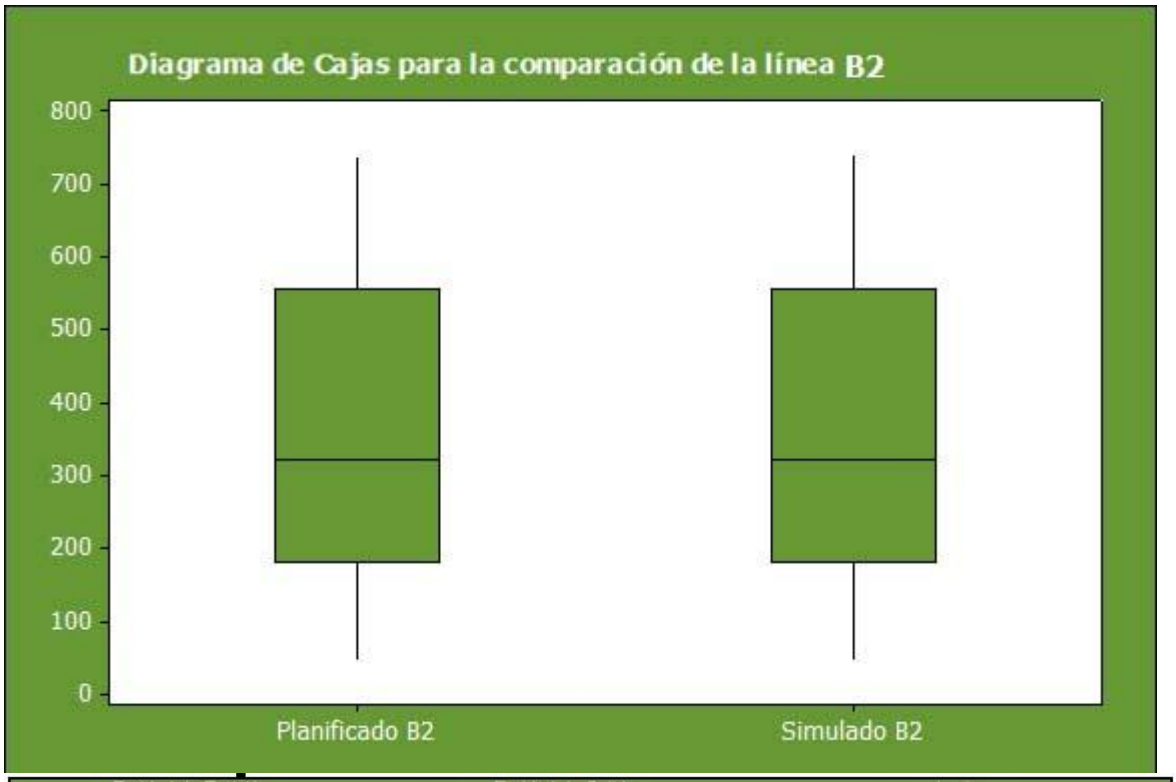


La simulación mediante PROMODEL ha probado ser no sólo una excelente herramienta estratégica para las empresas y sino que también puede ser utilizada

como una herramienta táctica en el día a día de cualquier compañía, así mismo el programa MINITAB 14 que aporta con análisis estadísticos a un conjunto de datos obteniendo información de manera fácil, rápida y asequible. Esta combinación de herramientas permite a las empresas obtener suficiente información fundamentada en datos reales y estudios estadísticos para poder tomar una decisión con fundamentos.







Entidad de Entrada	Entidad de Salida	Logica
B_V500	B_V250	wait 120 min□□
B_L250	B_V250	wait 30 min
B_L500	B_V250	wait 120 min
B_C250	B_V250	wait 30 min
B_C500	B_V250	wait 120 min
D_R250	B_V250	DesperdicioBl = DesperdicioBl + 200
D_R500	B_V250	DesperdicioBl = DesperdicioBl + 200
D_L250	B_V250	DesperdicioBl = DesperdicioBl + 200
D_L500	B_V250	DesperdicioBl = DesperdicioBl + 200
B_L500	B_L250	wait 120 min
B_C250	B_L250	wait 30 min
B_C500	B_L250	wait 120 min
D_R250	B_L250	DesperdicioBl = DesperdicioBl + 200
D_R500	B_L250	DesperdicioBl = DesperdicioBl + 200
D_L250	B_L250	DesperdicioBl = DesperdicioBl + 200
D_L500	B_L250	DesperdicioBl = DesperdicioBl + 200
B_V250	B_L250	DesperdicioBl = DesperdicioBl + 150
B_V500	B_L250	wait 120 min
B_C500	B_C250	wait 120 min
D_R250	B_C250	DesperdicioBl = DesperdicioBl + 200
D_R500	B_C250	DesperdicioBl = DesperdicioBl + 200
D_L250	B_C250	DesperdicioBl = DesperdicioBl + 200
D_L500	B_C250	DesperdicioBl = DesperdicioBl + 200
B_V250	B_C250	wait 30 min
B_V500	B_C250	wait 120 min
B_L250	B_C250	wait 30 min
B_L500	B_C250	wait 120 min
D_R500	D_R250	wait 120 min
D_L250	D_R250	wait 30 min
D_L500	D_R250	wait 120 min
B_V250	D_R250	wait 30 min
B_V500	D_R250	wait 120 min

$ 143,92
$ 191,89
$ 335,81
$ 0,00
$ 50,00
$ 100,00
$ 150,00
$ 200,00
$ 250,00
$ 300,00
$ 350,00
$ 400,00
0,00
100,00
200,00
300,00
400,00
500,00
600,00
700,00
800,00
900,00
1000,00
1
2
3
ESCENARIOS
Desperdicio Total (Kg)
CostoTotal ($)

$ 668,34
$ 639,75
$ 665,88
$ 625,00
$ 630,00
$ 635,00
$ 640,00
$ 645,00
$ 650,00
$ 655,00
$ 660,00
$ 665,00
$ 670,00
$ 675,00
0,00
100,00
200,00
300,00
400,00

500,00
600,00
700,00
800,00
900,00
1000,00
4
5
6
ESCENARIOS
Desperdicio Total (Kg)
CostoTotal ($)

74,3%
78,7%
83,1%
88,5%
0,0%
25,0%
50,0%
75,0%
100,0%
0,0
2000,0
4000,0
6000,0
8000,0
10000,0
2009
2010
2011
2012
Volumen
Capacidad Utilizada

47,5%
48,4%
49,4%
50,4%
0,0%
25,0%
50,0%
75,0%
100,0%
0,0
2000,0

4000,0
6000,0
8000,0
10000,0
2009
2010
2011
2012
Volumen
Capacidad Utilizada

47%
48%
49%
50%
74%
79%
83%
88%
0,0
2000,0
4000,0
6000,0
8000,0
10000,0
0,0%
25,0%
50,0%
75,0%
100,0%
2009
2010
2011
2012
Volumen Total (Em/B2)
Volumen Total (B1)
Capacidad Utilizada (Em/B2)
Capacidad Utilizada (B1)

48%

48%
61%
68%
74%
79%
77%
79%
0,0
2000,0
4000,0
6000,0
8000,0
10000,0
0,0%
25,0%
50,0%
75,0%
100,0%
2009
2010
2011
2012
Volumen Total (Em/B2)
Volumen Total (B1)
Capacidad Utilizada (Em/B2)
Capacidad Utilizada (B1)

La presente tesis está basada en una importante empresa multinacional representante de varias categorías en el Ecuador. La empresa ha demostrado un crecimiento y aumento de la demanda en todas sus categorías, en especial las del sector alimenticio que son las más delicadas del negocio. Buscaremos analizar las proyecciones de su demanda y como estos influyen en el desempeño de cada una de sus operaciones de los procesos productivos; utilizando para ello las herramientas de PROMODEL y MINITAB 14.

El procedimiento de este estudio tiene tres puntos principales; el primero de ellos es crear y validar el modelo de simulación del proceso productivo. El segundo paso es en la identificación del mejor escenario de secuenciamiento de los productos. El tercer y último paso es la identificación de cuellos de botellas al corto, mediano y largo plazo de todas sus operaciones del proceso productivo. Para poder identificar los posibles inversiones requeridas para satisfacer a la demanda y las exigencias del mercado.

CAPÍTULO 1

1. GENERALIDADES

1.1. Antecedentes

La empresa multinacional en la cual se desarrollo este estudio a la cual se llamará Producto Varios S.A. ingresó al Ecuador en los años 90 adquiriendo varias corporaciones con el objetivo de satisfacer al mercado de productos de cuidado personal, salud y alimentación.

La compañía antes mencionada en búsqueda de productos saludables consideró reemplazar la mantequilla por ser un producto que contiene altos porcentajes de grasa dañinos para la salud. A escala de la región esta exigencia comenzó hace 10 años. En los primeros años del siglo XXI con los objetivos antes mencionados esta organización introdujo al mercado ecuatoriano la margarina.

La margarina está elaborada con aceites vegetales, líquidos a temperatura ambiente (maíz, girasol, soya o coco), que se someten a un proceso industrial de endurecimiento para adquirir su consistencia sólida y untable, a diferencia de la mantequilla que es la grasa de la leche, que pasa por un proceso de pasteurización, centrifugación y sedimentación en frío, para obtener un producto graso.

A escala regional, el Ecuador se convirtió en el tercer consumidor de importancia en el consumo de margarinas con ventas anuales que superan los $15 millones. Entre las marcas de mayor relevancia del mercado ecuatoriano están: Regia, Klar, Bonella y Dorina.

En cuestión de tendencias, la mayoría de los consumidores de este alimento, que contiene vitamina E y capacidad antioxidante, aún opta por la margarina regular. Ipsa Group, con un estudio pormenorizado, determina que el 90% de los consumidores aún lo prefiere así. Entre los consumidores más frecuentes se encuentran los niños. Solo el 10% de los clientes buscan algo más ligero y con grasas insaturadas, necesarias para la alimentación diaria. En el Ecuador el consumo per cápita es de 1040 gramos.

El Untar margarina sobre el pan o disfrutarla en algún postre o comida de sal es un gustito que a diario crece y que en el año 2006 representó un consumo de

9500 toneladas métricas. A partir del 2007 el crecimiento se ha visto impactado por doble dígito, debido a que este producto está siendo utilizado en mayor cantidad por el sector pastelero.

A continuación se muestra como ha evolucionado el crecimiento de ventas de la empresa Productos Varios S.A. a través de los años, sus diferentes estrategias para obtener la mayor participación en el mercado en el Ecuador están dando resultados y muestra que por el momento no se detendrá su crecimiento. Estos antecedentes hacen que sea de vital importancia conocer si sus equipos podrán sostener el crecimiento deseado.

FIGURA 1.1 VENTA PROYECTADAS (TON) DE MARGARINAS

1.2. Objetivo General

Determinar mediante un modelo de simulación el nivel de utilización de todo el proceso productivo de la planta de margarinas considerando los volúmenes de ventas actuales y proyectadas al 2012. Con el fin de identificar el secuenciamiento más adecuado y potenciales restricciones.

1.3. Objetivos Específicos

- Realizar un modelo de simulación del proceso productivo de la fábrica de margarinas.

- Analizar las diferentes opciones de secuenciamiento en la línea de envasado e identificar la más adecuada considerando para su evaluación costos y nivel de desperdicio.

- Identificar los porcentajes de utilización actual de las líneas de envasado como el de cada una de las operaciones del proceso productivo.

- Evaluar económicamente las propuestas planteadas.

1.4. Metodología utilizada

En el capítulo inicial de este trabajo, se detalla los antecedentes de la empresa donde es realizado este estudio describiendo el sector a analizar y como este ha evolucionado a través del tiempo, identificando el núcleo de nuestro trabajo. Adicional a esto se detalla nuestro objetivo general como también los específicos que se busca en el transcurso y finalización de la tesis.

En nuestro Capítulo 2, se iniciará detallando los productos finales y sus características, se continúa describiendo el proceso productivo, comentando la función y objetivo de cada operación del flujo, finalmente describiendo los recursos y equipos utilizados que intervienen en cada etapa u operación.

En la Descripción del Modelo de Simulación, Capítulo 3, se presentarán las principales asunciones que fueron empleadas, la programación realizada y las variables utilizadas para poder modelar y plasmar con mayor exactitud el proceso productivo. También detallará toda la información recolectada sobre los parámetros requeridos para simular una planta de procesos continuos.

En el Capítulo 4, se procederá a la Validación del Modelo de Simulación para esto se plantearán y las variables de respuesta que serán utilizadas con el fin de comparar los datos reales con respecto a lo presentado por el modelo.

Persiguiendo los objetivos se entra al Capítulo 5, donde se planteará y describirá los cambios a realizarse en el modelo de simulación con el fin de evaluar las diferentes formas de secuenciación, las mismas que serán programadas y evaluadas. Posteriormente se presentarán los resultados obtenidos de cada una de las alternativas planteadas con el fin de analizar los resultados obtenidos.

Seguido de los cambios se entra al Capitulo 6, donde se valorará los resultados obtenidos y sus impactos.

Por último, se presenta el Capítulo 7 llamado Conclusiones y Recomendaciones, donde se detallarán las respuestas a nuestros objetivos planteados al inicio del estudio. Como también las conclusiones y recomendaciones a la empresa donde se realizó el trabajo.

CAPÍTULO 2

2. DESCRIPCIÓN GENERAL DE LA EMPRESA

2.1. Descripción de los Productos

La presente empresa elabora una gama de productos que satisfacen múltiples requerimientos del mercado ecuatoriano, entre estas tenemos las margarinas. Existen varios tipos de margarinas que se las diferencian por la composición de la mezcla de grasa y por el proceso de fabricación utilizado. En cuanto a sus características se las diferencia básicamente por sabor, esparcibilidad y dureza (Temperatura).
Las margarinas elaboradas por esta empresa pueden ser clasificados en 13 diferentes productos finales (skus), la cuales están caracterizados por su tamaño y producto.

Por guardar la confidencialidad, los skus tendrán una nomenclatura diferente a la que realmente se utiliza, los productos finales que tiene esta empresa son:

A) BV250

B) BV500
C) BL250
D) BL500
E) BC250
F) BC500
G) DR250
H) DR500
I) DL250
J) BV1000
K) BV3000
L) BV50
M) DR1000

Referente a su tamaño:

En su portafolio la compañía contiene varias presentaciones, estas son: tarrinas de 250, 500 y 1000 gramos con una lámina de papel aluminio en su interior que separa la margarina de la tapa del envase, tachos de 3000 gramos y lámina para

envasar de 50 gramos; todos estos materiales son impermeables a sustancias aromáticas y a la humedad, de lo contrario la superficie de la margarina se secaría y las capas exteriores adquirirían un color más amarillo que en la parte interior.

Referente a Productos:

El portafolio cuenta con 2 tipos o familias de productos, los productos B representan el 90% de su volumen anual y fueron creados para satisfacer el mercado desde el medio – alto hasta el bajo nivel socio – económico. La segunda familia de producto es el D (Inicia con la letra D) la cual representa el porcentaje restante de volumen anual de la compañía, el 10%; este producto está dirigido principalmente al estrato socio – económico alto.

2.2. Descripción del Proceso Productivo

El proceso de fabricación de las margarinas sigue el diagrama de flujo que se muestra a continuación:

Recepción MaterialesPreparación de EmulsiónAceitesFase AcuosaFase GrasaTanque pulmónMPUsTanque de ReWorkEnvasado

FIGURA 2.1 DIAGRAMA DE FLUJO DEL PROCESO DE FABRICACIÓN

Recepción de Material

El proceso inicia con la recepción de los materiales, los cuales están compuestos en un gran porcentaje por aceites líquidos y micro ingredientes como vitaminas, emulsificantes, etc. Los aceites son recibidos a granel en tanques que tienen desde 10 hasta 25 toneladas de
almacenamiento; para los micros ingredientes son utilizadas bodegas secas y/o refrigeradas dependiendo de la especificación del material.

Proceso de limpieza

El proceso de limpieza consiste en varias etapas, inicia con la purga de agua caliente, paso y enjuague del detergente caustico, paso y enjuague de detergente ácido, desinfección, preservación y por último el enjuague final. Este proceso es indispensable para la remoción de micro organismos en las líneas del proceso y para el aumento operativo del equipo.

Proceso de Mezclado y Homogenización

El objetivo de este proceso es que se dividan finamente los glóbulos de grasa en la mezcla para conseguir una suspensión permanente, evitando que la grasa se

separe del resto de los componentes y ascienda hacia la superficie.

Aquí se mezclan los aceites hidrogenados, los materiales hidrosolubles (Fase acuosa), los materiales Liposolubles (Fase grasa) y los micros ingredientes como las vitaminas necesarias para elaborar la emulsión.

El proceso de mezclado se lo realiza por lotes, se utilizan depósitos báscula para medir el peso provisto de un potente agitador. Se dispone de un tanque para realizar la emulsión y otro tanque que sirve de pulmón para que no afecte a la producción pues mientras se produce utilizando la emulsión del tanque pulmón, se inicia la preparación del otro lote. Para este proceso continuo se utilizan bombas dosificadoras para todos los ingredientes, ya que la emulsión contiene una ruta de preparación la cual no puede ser alterada.

La mayoría de los equipos utilizados en el proceso son de acero inoxidable ya que este metal combinado con los emulsificantes, favorecen la solubilización de los glóbulos grasos.

Proceso de Enfriamiento y Cristalización

La cristalización es la formación de una red de cristales sólidos de grasa que sostendrá todo el conjunto de componentes e ingredientes de la margarina.

Este proceso es realizado en unos equipos llamados MPUs, que tienen como objetivo transformar una emulsión líquida inestable en un producto cremoso, esparcible y estable. Las operaciones de este equipo son de enfriamiento, cristalización, trabajo y emulsificación.

Estos equipos tienen como objetivo el enfriamiento y movimiento continuo de la emulsión grasa proporcionando una gran superficie de intercambio térmico para una pequeña cantidad de producto en un espacio limitado. Dentro del aparato existe un eje con cuchillas que gira dentro del tubo, la suspensión grasa pasa a través de la zona anular formada por el eje y el tubo de intercambio térmico. El enfriamiento se efectúa mediante la expansión de amoniaco que circula por el tubo intermedio. El conjunto se aísla para reducir al mínimo las pérdidas de energía y como protección para las personas. El exterior es de acero inoxidable para proteger el aislamiento. En la operación, las cuchillas del eje rotativo raspan continuamente la película del producto que se va formando en la superficie de intercambio térmico, con lo que se consigue aumentar este intercambio de calor además de agitar el producto generando una mezcla homogénea.

Proceso de Envasado

Los métodos utilizados para el transporte de margarina hasta la máquina de empaquetado es continua, en este caso la margarina se bombea hasta la envasadora. .

En esta empresa la margarina se presentan en distintos envases. Dependiendo del tipo de envase, se utilizan distintos tipos de máquinas.
Todas las máquinas son automáticas excepto una que es manual. Los envases llenos son colocados en cajas selladas y son trasladadas mediante una banda para ser estibadas respetando las normas de apilamiento y finalmente almacenadas.

FIGURA 2.2 DIAGRAMA DEL PROCESOS DE FABRICACION DE MARGARINAS

2.3. Descripción Física de la Planta

La empresa en estudio está formada por:

• Recepción y almacenamiento de aceites
Cuenta con tanques desde 10 hasta 25 toneladas, estos tanques son especiales y acondicionados acordes a las condiciones ambientales requeridas para cada uno de los aceites.

• Fase acuosa
La Fase acuosa cuenta en pequeños tanques de 1200 kg que contienen los ingredientes hidrosolubles de la emulsión.

• Preparación
Esta es la sección de preparación de la emulsión, cuenta con tanques mezcladores básculas de 3800 y 2800 kg con sus respectivos tanques pulmones de la misma capacidad de los mezcladores.

• MPUs y Envasadoras
Cuenta con 2 MPUs, y el área de envasado en cambio está conformada por 3 líneas de las cuales salen varias presentaciones

• Paletizado y Almacenado
Esta área es la encargada del paletizado de las cajas provenientes de las 3 líneas de envasado siguiendo por la bodega pulmón refrigerada hasta finalmente ser enviada al Centro Nacional de Distribución.

Adicionalmente, existen otros sistemas auxiliares dentro de la planta
los cuales son los que suministran los diversos servicios generales que se requieren en una planta de producción, como son el vapor, aire, energía eléctrica, frío y tratamiento de agua, los cuales mencionamos a continuación:

- Sistema de Generación de Vapor
- Sistema de Suministro de Aire Comprimido
- Sistema de Suministro de Amoniaco

- Planta de Potabilización de agua
- Planta de Tratamiento de agua residual
- Sub estación eléctrica

CAPÍTULO 3

3. DESCRIPCIÓN DEL MODELO DE SIMULACION

Para la creación de un modelo de simulación se inicia con la definición de todos los elementos que están involucrados en el proceso real como son entidades, locaciones, macros, atributos, variables, subrutinas, etc. En ciertos casos no son identificados al inicio de la construcción del modelo sino en el transcurso de este.

3.1. Creación de Entidades

Iniciamos con la creación de Entidades, las cuales pueden ser reales y que son representativas al proceso en estudio. En otros casos son entidades ficticias que únicamente soportan el modelo.

Dado que el modelo maneja simulación continua y discreta se crearon entidades para realizar la conversión desde las locaciones que manejan la parte continua del modelo (Emulsión manejadas por los tanques) hasta las locaciones que manejan la parte discreta del mismo.

Adicionalmente se crearon entidades para el control de: transferencia, preparación y vaciado de los tanques en base a una planificación
ingresada por el usuario. Estas entidades circulan por locaciones ficticias que en la realidad son operaciones manejadas por el personal de planta o de manera automática por la maquinaria. Por ejemplo: abrir válvulas, controlar la cantidad de los ingredientes, realizar la limpieza, etc.

En la tabla 3.1, se muestra con mayor detalle las entidades creadas para el modelo y con la descripción de sus usos.

B_V250	Entidad que representa 1 kg. de producto de BV 250
B_V500	Entidad que representa 1 kg. de producto de BV 500
B_L250	Entidad que representa 1 kg. de producto de BL 250
B_L500	Entidad que representa 1 kg. de producto de BL 500
B_C250	Entidad que representa 1 kg. de producto de BC 250
B_C500	Entidad que representa 1 kg. de producto de BC 500
B_V1000	Entidad que representa 1 kg. de producto de BV 1000
B_V50	Entidad que representa 1 kg. de producto de BV 50
D_R250	Entidad que representa 1 kg. de producto de DR 250
D_R500	Entidad que representa 1 kg. de producto de DR 500
D_L250	Entidad que representa 1 kg. de producto de DL 250
D_L500	Entidad que representa 1 kg. de producto de DL 500
D_R1000	Entidad que representa 1 kg. de producto de DR 1000

B_V3000	Entidad que representa 1 kg. de producto de BV 3000
PlanDiario	Entidad de Inicio que ingresa al modelo por medio del archivo de arribos.
OrdenProduccionB1	Entidad que produce el producto en el tanque de capacidad máxima 3,8 ton.
OrdenProduccionB2	Entidad que produce el producto en el tanque de capacidad máxima 2,8 ton.

TABLA 3.1
ENTIDADES DEL MODELO DE SIMULACIÓN

3.2. Creación de Locaciones

Las locaciones creadas en este modelo como tanques son usadas para almacenaje de ingrediente, almacenaje de producto (Para esperar a ser procesados), para cristalización de la emulsión y para envasado.

En el modelo se crearon locaciones reales que representan a las máquinas donde el producto es procesado como también existen locaciones ficticias que se crearon con el fin facilitar ciertas actividades propias de la simulación continua (No cumplen una regla de ruteo) que no interfieren en los resultados finales.

En la tabla 3.2, se muestra con mayor detalle las locaciones creadas para el modelo y con la descripción de sus usos.

I1	50 ton	Tanque del ingrediente 1.
I2	20 ton	Tanque del ingrediente 2.
I3	20 ton	Tanque del ingrediente 3.
I4	30 ton	Tanque del ingrediente 4.
I5	30 ton	Tanque del ingrediente 5.
I6	50 ton	Tanque del ingrediente 6.
I7	100 kg	Tanque del ingrediente 7.
I8	4.4 ton	Tanque del ingrediente 8.
I9	2.4 ton	Tanque del ingrediente 9.

	I10	1 ton	Tanque del ingrediente 10.
	Premix1	3.8 ton	Tanque de mezcla de ingredientes y preparación de la emulsión.

Pulmon1	3.8 ton	Tanque que tiene como función alimentar al MPU Kombinator mientras el Premix1 prepara otro lote de producción.
MPU Kombinator	1	Máquina que tienen como función el enfriamiento y cristalización de la emulsión.
B1	1	Máquina envasadora, para los productos terminados en la presentación de 250 y 500gr.

Premix2	2.8 ton	Tanque de mezcla de ingredientes y preparación de la emulsión.
Pulmon2	2.8 ton	Tanque que tiene como función alimentar al MPU Votator mientras el Premix2 prepara otro lote de producción.
Votator	1	Máquina que tienen como función el enfriamiento y cristalización de la emulsión.
recirculacionB2	infinita	Bodega ficticia para el manejo de la recirculación
ventanaB2	1	Locación ficticia para el ruteo de las líneas Em y B2
B2	1	Máquina envasadora, para los productos terminados en la presentación de 1000gr.

Em	1	Máquina envasadora, para los productos terminados en la presentación de 50 y 3000gr.
BPTB1	infinita	Bodega de salida de la envasadora B1
BPTB2	infinita	Bodega de salida de la envasadora B2
BPTEm	infinita	Bodega de salida de la envasadora Em
Cola_SepOrd	infinita	Locación para recepción del plan máster de producción
Separacion_Ordenes	1	Locación para separación del plan máster, en un número de órdenes de producción iguales o menores al contenido máximo de los tanques de preparación.
Cola_PlanificacionB1	infinita	Locación de espera para la ordenes generadas en la locación Separacion_Ordenes para la línea B1
Cola_PlanificacionB2	infinita	Locación de espera para la ordenes generadas en la locación Separacion_Ordenes para la línea B2
PlanificacionB1	1	Locación de decisión, ¿Si el tanque Premix1 está lleno? La orden espera - caso contrario la envía a la siguiente locación
ProduccionB1Tanques	1	Locación usada para la transferencia de ingredientes al tanque Premix1, seguido de la preparación de la emulsión en dicho tanque.
VaciadoTanqueB1	1	Locación de decisión, ¿Si el tanque Pulmon1 contiene el mismo producto que el Premix1? Este es vaciado al Pulmon1 - caso contrario lo hace esperar.

ProduccionB1	1	Locación que discretisa el contenido del tanque Pulmon1 creando una entidad de 1kg en el MPU Kombinator
PlanificacionB2	1	Locación de decisión, ¿Si el tanque Premix2 está lleno? La orden espera - caso contrario la envía a la siguiente locación
ProduccionB2Tanques	1	Locación usada para la transferencia de ingredientes al tanque Premix2 seguido de la preparación de la emulsión en dicho tanque.
VaciadoTanqueB2	1	Locación de decisión, ¿Si el tanque Pulmon2 contiene el mismo producto que el Premix2? Este es vaciado al Pulmon2 - caso contrario lo hace esperar.
ProduccionB2Tanques	1	Locación usada para la transferencia de ingredientes al tanque Premix2, seguido de la preparación de la emulsión en dicho tanque.
ProduccionB1	1	Locación que discretisa el contenido del tanque Pulmon2 creando una entidad de 1kg en el MPU Kombinator

TABLA 3.2
LOCACIONES DEL MODELO DE SIMULACIÓN

3.3. Creación de Macros

Las macros son usadas cuando existen expresiones, funciones y un grupo de declaraciones que se repiten varias veces en el modelo [1]. El programa Promodel viene preparado para la simulación continua, es por esto que

al crear el primer tanque automáticamente se crean las macros para el control de los estados de los mismos. Fue necesario la creación de macros para la trasferencia y preparación de las diferentes emulsiones.

En la tabla 3.3, se muestra con mayor detalle las Macros creadas para el modelo y con la descripción de sus usos

Macros usadas por el programa para controlar el estado de los tanques.	Tank_Idle	0	Tanque ocioso
	Tank_Operation	1	Tanque en operación
	Tank_Setup	2	Tanque en cambio de formato/setup
	Tank_Filling	3	El tanque se está llenado
	Tank_Emptying	4	El tanque se está vaciando
	Tank_Blocked	5	Tanque bloqueado
	Tank_Down	6	Tanque en mantenimiento
	Tank_Loop	While 1 do	Para uso de los bucles en los tanques

Tank_TimeStep	0,2	Tiempo que toma entre cada llenado/vaciado de un tanque
Tank_Stop	-1	El tanque detiene cualquier operación que esté realizando
Tank_LongestIdle	1	Estado que define que el tanque es el de mayor tiempo ocioso
Tank_LongestBlocked	2	Estado que define que el tanque es el de mayor tiempo bloqueado
Tank_InOrder	3	

Macros usadas para la preparaciones de las emulsiones	Emulsion3800B1_1	// BV // Macro para preparación = 3800
		Tank_Transfer(I1, Premix1, 1201.58, 97.7+ L(29.4,25.5), 0, 0)
		Tank_Transfer(I2, Premix1, 345.80, 1+L(251,112), 0, 0) Tank_Transfer(I3, Premix1, 292.6, 1+L(251,112), 0, 0)
		Tank_Transfer(I4, Premix1, 425.6, 1+L(248,144), 0, 0) Tank_Transfer(I5, Premix1, 399, 1+L(248,144), 0, 0)
		Tank_Transfer(I6, Premix1, 598.61, 309 + L(614,951), 0, 0) wait 1 min
		Tank_Inc(Premix1,0.23)
		Tank_Dec(I7,0.23) //Tank_Transfer(I7, Premix1, 0.23, N(0.305,0.152), 0, 0)
		Tank_Transfer(I8, Premix1, 475, 245+L(131,536), 0, 0)
		Tank_Transfer(I9, Premix1, 42.22,20.6+L(78.8,354) , 0, 0)
		Tank_Transfer(I10, Premix1, 19.36,1+L(1.63,0.644), 0, 0)
		Tank_DoOperation(Premix1, 10)
			Macro usada para la preparación de la emulsión definida para el producto BV en el tanque Premix1

Emulsion3800B1_2	// BL // Macro para preparación = 3800 Tank_Transfer(I1, Premix1, 952.869, 97.7+ L(29.4,25.5), 0, 0) Tank_Transfer(I2, Premix1, 271.70, 1+L(251,112), 0, 0) Tank_Transfer(I3, Premix1, 229.9,1+L(251,112), 0, 0) Tank_Transfer(I4, Premix1, 334.4, 1+L(248,144), 0, 0) Tank_Transfer(I5, Premix1, 313.5, 1+L(248,144), 0, 0) Tank_Transfer(I6, Premix1, 1211.869, 309 + L(614,951), 0, 0) wait 1 min Tank_Inc(Premix1,0.228) Tank_Dec(I7,0.228) //Tank_Transfer(I7, Premix1, 0.228, N(0.305,0.152), 0, 0) Tank_Transfer(I8, Premix1, 323, 245+L(131,536), 0, 0) Tank_Transfer(I9, Premix1, 137.222, 20.6+L(78.8,354) , 0, 0) Tank_Transfer(I10, Premix1, 25.312, 1+L(1.63,0.644), 0, 0) Tank_DoOperation(Premix1, 10) // Operación de mezcla	Macro usada para la preparación de la emulsión definida para el producto BL en el tanque Premix1
Emulsion3800B1_3	// BC // Macro para preparación = 3800 Tank_Transfer(I1, Premix1, 1117 , 97.7+ L(29.4,25.5), 0, 0) Tank_Transfer(I2, Premix1, 321.1, 1+L(251,112), 0, 0) Tank_Transfer(I3, Premix1, 271.7, 1+L(251,112), 0, 0) Tank_Transfer(I4, Premix1, 370.5, 1+L(248,144), 0, 0) Tank_Transfer(I5, Premix1, 395.2, 1+L(248,144), 0, 0) Tank_Transfer(I6, Premix1, 788.797, 309 + L(614,951), 0, 0) wait 1 min Tank_Inc(Premix1,0.228) Tank_Dec(I7,0.228) //Tank_Transfer(I7, Premix1, 0.228, N(0.305,0.152), 0, 0) Tank_Transfer(I8, Premix1, 475, 245+L(131,536), 0, 0) Tank_Transfer(I9, Premix1, 42.222, 20.6+L(78.8,354) , 0, 0) Tank_Transfer(I10, Premix1, 18.253,1+L(1.63,0.644), 0, 0) Tank_DoOperation(Premix1, 10) // Operación de mezcla	Macro usada para la preparación de la emulsión definida para el producto BC en el tanque Premix1

Emulsion3800B1_4	// DR // Macro para preparación = 3800 Tank_Transfer(I1, Premix1, 2337, 97.7+ L(29.4,25.5), 0, 0) Tank_Transfer(I2, Premix1, 0, 1+L(251,112), 0, 0) Tank_Transfer(I3, Premix1, 0, 1+L(251,112), 0, 0) Tank_Transfer(I4, Premix1, 0, 1+L(248,144), 0, 0) Tank_Transfer(I5, Premix1, 239, 1+L(248,144), 0, 0) Tank_Transfer(I6, Premix1, 720, 309 + L(614,951), 0, 0) wait 1 min Tank_Inc(Premix1,89) Tank_Dec(I7,89) //Tank_Transfer(I7, Premix1, 89, N(0.305,0.152), 0, 0) Tank_Transfer(I8, Premix1, 399, 245+L(131,536), 0, 0) Tank_Transfer(I9, Premix1, 0, 20.6+L(78.8,354) , 0, 0) Tank_Transfer(I10, Premix1, 16, 1+L(1.63,0.644), 0, 0) Tank_DoOperation(Premix1, 10) // Operación de mezcla	Macro usada para la preparación de la emulsión definida para el producto DR en el tanque Premix1

Emulsion3800B1_5	// DL // Macro para preparación = 3800 Tank_Transfer(I1, Premix1, 1840.3387, 97.7+ L(29.4,25.5), 0, 0) Tank_Transfer(I2, Premix1, 0, 1+L(251,112), 0, 0) Tank_Transfer(I3, Premix1, 0, 1+L(251,112), 0, 0) Tank_Transfer(I4, Premix1, 0, 1+L(248,144), 0, 0) Tank_Transfer(I5, Premix1, 188.1, 1+L(248,144), 0, 0) Tank_Transfer(I6, Premix1, 1286.9477, 309 + L(614,951), 0, 0) wait 1 min Tank_Inc(Premix1,69.9516) Tank_Dec(I7,69.9516) //Tank_Transfer(I7, Premix1, 69.9516, N(0.305,0.152), 0, 0) Siempre demora 1 min Tank_Transfer(I8, Premix1, 399, 245+L(131,536), 0, 0) Tank_Transfer(I9, Premix1, 0, 20.6+L(78.8,354) , 0, 0) Tank_Transfer(I10, Premix1, 15.662, 1+L(1.63,0.644), 0, 0) Tank_DoOperation(Premix1, 10) // Operación de mezcla	Macro usada para la preparación de la emulsión definida para el producto DL en el tanque Premix1
Emulsion2800B2_1	//BV // Macro para preparación = 2800 Tank_Transfer(I1, Premix2, 885.374, 97.7+ L(29.4,25.5), 0, 0) Tank_Transfer(I2, Premix2, 254.8, 1+L(251,112), 0, 0) Tank_Transfer(I3, Premix2, 215.6, 1+L(251,112), 0, 0) Tank_Transfer(I4, Premix2, 313.6, 1+L(248,144), 0, 0) Tank_Transfer(I5, Premix2, 294, 1+L(248,144), 0, 0) Tank_Transfer(I6, Premix2, 441.083, 309 + L(614,951), 0, 0) wait 1 min Tank_Inc(Premix2,0.168) Tank_Dec(I7,0.168) //Tank_Transfer(I7, Premix2, 0.168, N(0.305,0.152), 0, 0) Tank_Transfer(I8, Premix2, 350, 245+L(131,536), 0, 0) Tank_Transfer(I9, Premix2, 31.111,20.6+L(78.8,354) , 0, 0) Tank_Transfer(I10, Premix2, 14.264,1+L(1.63,0.644), 0, 0) Tank_DoOperation(Premix2, 10)	Macro usada para la preparación de la emulsión definida para el producto BV en el tanque Premix2

	// Operación de mezcla	
Emulsion2800B2_4	// DR // Macro para preparación = 2800 Tank_Transfer(I1, Premix2, 1721.767, 97.7+ L(29.4,25.5), 0, 0) Tank_Transfer(I2, Premix2, 0, 1+L(251,112), 0, 0) Tank_Transfer(I3, Premix2, 0, 1+L(251,112), 0, 0) Tank_Transfer(I4, Premix2, 0, 1+L(248,144), 0, 0) Tank_Transfer(I5, Premix2, 176.4, 1+L(248,144), 0, 0) Tank_Transfer(I6, Premix2, 530.506, 309 + L(614,951), 0, 0) wait 1 min Tank_Inc(Premix2,65.543) Tank_Dec(I7,65.543) //Tank_Transfer(I7, Premix2, 65.543, N(0.305,0.152), 0, 0) Tank_Transfer(I8, Premix2, 294, 245+L(131,536), 0, 0) Tank_Transfer(I9, Premix2, 0, 20.6+L(78.8,354) , 0, 0) Tank_Transfer(I10, Premix2, 11.784, 1+L(1.63,0.644), 0, 0) Tank_DoOperation(Premix2, 10) // Operación de mezcla	Macro usada para la preparación de la emulsión definida para el producto DR en el tanque Premix2

TABLA 3.3
MACROS DEL MODELO DE SIMULACIÓN

3.4. Creación de Arribos

Los arribos nos permiten ingresar las entidades al sistema, estos contienen la planificación de un periodo. El periodo tomado a consideración es el mes de Abril del 2009. En nuestro caso la planificación es un archivo en Excel preparado para que el sistema lo pueda reconocer y leer, esto permite tener mayor flexibilidad y agilidad para realizar cambios.

3.5. Descripción de Simulación con Tanques

Los tanques, como ya se lo mencionó anteriormente, son simplemente locaciones en las cuales Promodel asocia un nivel de volumen. Usando tanques se puede modelar el fluido continuo de líquidos u otras sustancias que entran o salen de diferentes procesos. Esto combinado con la simulación discreta permite modelar de manera efectiva el intercambio de un fluido continuo con entidades discretas [2].
Para un apropiado funcionamiento, todos los modelos de tanques deben incluir el sub-modelo tanque (TANK-SUB.MOD). Este contiene subrutinas y elementos (arrays y macros) que se explicará más adelante. Este sub-modelo se carga de manera automática al crear el primer tanque [2].

Conceptos básicos de tanques

Debido a que los tanques no procesan entidades discretas, no se pueden definir rutas ni desde ni hacia los tanques. Para controlar el nivel de un tanque, Promodel provee de subrutinas predefinidas para llenar, vaciar y transferir el contenido del mismo. Para monitorear los niveles de los tanques e iniciar el flujo del contenido, se deben definir subrutinas de control. Para llamar a estas subrutinas se usa el comando ACTIVATE. Para modelar de manera efectiva los tanques, se deben conocer los siguientes conceptos.

Tank Level (Nivel del Tanque)

Promodel registra el nivel del tanque en un array previamente definido llamado Tank_Level.
The Flow Time Step

Para modelar fluido continuo, Promodel usa la macro Tank_TimeStep. Esta macro es el tiempo que transcurre entre cada llenado/vaciado de tanques.

Rate of Flow (Caudal)

Para hacer uso del caudal apropiadamente se deben definir las unidades (litros, kilos, etc.) por una unidad de tiempo. Al momento de llenar, vaciar o transferir se debe especificar el caudal.

Tank States (Estado de los Tanques)

Los tanques usan los estados para probar y registrar estadísticas. Promodel cambia el estado del tanque automáticamente al hacer uso de las subrutinas predefinidas para el control de los tanques. El detalle de los estados se lo puede

observar en la tabla de macros.

Subrutinas Predefinidas

Para el control de los niveles en los tanques Promodel proporciona diferentes subrutinas predefinas. Para el entendimiento completo del modelo se explicaran a detalle las subrutinas usadas en el mismo:
TANK_TRANSFER (<FROM Tank ID>, <TO Tank ID>, <Transfer Quantity>, <FROM Rate>, <TO Rate>, <Resume Level>)

Esta subrutina transfiere una cantidad específica de un tanque a otro. Se detallan los campos:
<FROM Tank ID>; el nombre del tanque de donde sale el contenido.
<TO Tank ID>; el nombre del tanque a donde ingresa el contenido.
<Transfer Quantity>; La cantidad total a ser transferida.
<FROM Rate>; caudal del tanque de salida.
<TO Rate>; caudal del tanque de entrada.
<Resume Level>; si el tanque de entrada se llena, este debe esperar a que llegue al nivel especificado en este campo para que reanude la transferencia.

TANK_INC (<Tank ID>, <Quantity>)
Incrementa instantáneamente el nivel del tanque en la cantidad especificada.

<Tank ID>; nombre del tanque.
<Quantity>; cantidad a incrementar.
TANK_DEC (<Tank ID>, <Quantity>)
Disminuye instantáneamente el nivel del tanque en la cantidad especificada.

<Tank ID>; nombre del tanque.
<Quantity>; cantidad a incrementar.
TANK_DOOPERATION (<Tank ID>, <Operation time>)

Cambia el estado del tanque a Tank_Operation y espera el tiempo de operación especificado.

<Tank ID>; nombre del tanque
<Operation time>; el tiempo de duración de la operación.
TANK_DOPREP (<Tank ID>, <Prep time>)

Cambia el estado del tanque a Tank_Setup y espera el tiempo de setup especificado. Esta función se la usó para la limpieza de los tanques.

<Tank ID>; nombre del tanque
<Prep time>; el tiempo de duración del setup.

Con esta información se pretende dar una introducción para realizar modelos continuos. Se detallaron las subrutinas que fueron usadas en el modelo.

Consideraciones del Modelo de Simulación

Para poder representar de la manera más realista posible la planta productiva de margarinas se realizó un diagrama de flujo de proceso; cabe resaltar que se tomaron todas las operaciones del proceso real y se tuvo que incluir otras operaciones ficticias, como ya anteriormente mencionamos.

La información utilizada para modelar el proceso fue obtenida por 3 fuentes:

1.- Datos históricos ya registrados por la planta.
2.- Datos Obtenidos por toma de tiempos de transferencia de ingredientes al tanque preparador.
3.- Planes de Producción utilizados para cumplir la demanda mensual.

La planta de margarinas tiene y practica la filosofía TPM (Total Preventive Maintenance) y 5S, por ende los registros y la administración de estos son la base para resolver cualquier problema que se presenta. Los datos que obtuvimos son tasas de la envasadora y el MPU, nominales y efectivas, nominales, como las tasas de fábrica del equipo y las efectivas, de registros históricos en base a las diferentes pérdidas que afectan a la eficiencia operacional de los equipos.
La eficiencia operacional (OEE) está definida por TPM como un indicador para medir la eficiencia desde el tiempo de carga hasta el tiempo de operación valuado.
A continuación se mencionará las pérdidas más comunes:
Arranque/parada
Pequeñas paradas
Cambio (change over)
Gerenciamiento
Mov. Operacionales
Falla de equipo (breakdown)
Velocidad (speed)
Organización de línea
Logística
Defectos
Re-trabajo, medición y ajuste.

3.6. Descripción del Modelo de Simulación

Dado que el modelo de simulación de la planta de margarinas tiene como inicio de sus procesos tanques preparadores y tanques buffers se vio la necesidad de realizar un pequeño estudio de tiempos para la transferencia de los ingredientes al tanque de preparación. Estos tanques como ya se mencionó anteriormente son simplemente locaciones en las cuales Promodel asocia un nivel de volumen a diferencia del ruteo de una entidad.
Usando tanques se puede modelar el fluido continuo de líquidos u otras sustancias que entran o salen de los tanques. Esto combinado con la simulación discreta

permite modelar de manera efectiva el intercambio de un fluido continuo con entidades discretas [2].

La programación de la producción utilizada para la simulación de modelo consideró todos los flujos obtenidos para la preparación de los tanques, las pérdidas afectadas por la Eficiencia operacional y la secuencia del plan de un mes donde se obtuvo en la categoría de margarinas un nivel de servicio excelente con los indicadores productivos y financieros muy buenos.

En el siguiente diagrama se muestra el flujo y secuencia de las operaciones programadas en el modelo.

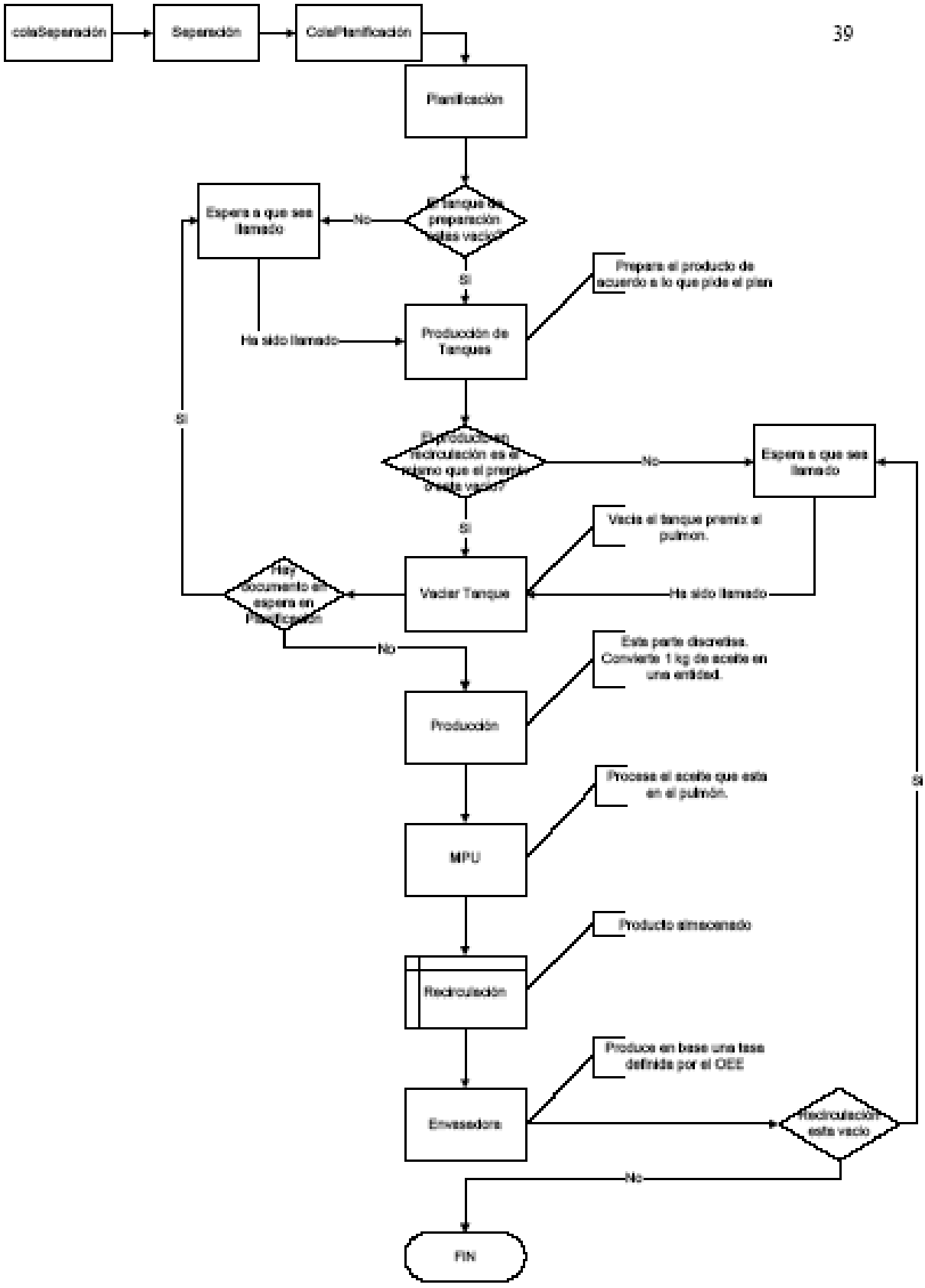

colaSeparación
Separación
ColaPlanificación
Planificación
El tanque de preparación esta vacio?
No
Espera a que sea llamado
Si
Prepara el producto de acuerdo a lo que pide el plan
Ha sido llamado
Producción de Tanques
El producto en recirculación es el mismo que el premix o esta vacio?
No
Espera a que sea llamado
Si
Vacia el tanque premix al pulmon.
Hay documento en espera en PlanificaciÓn
Vaciar Tanque
Ha sido llamado
Si
No
Esta parte discretiza. Convierte 1 kg de aceite en una entidad.
Producción
Procesa el aceite que esta en el pulmón.
MPU
Producto almacenado
Recirculación
Produce en base una tasa definida por el OEE
Envasadora
Recirculación esta vacio
Si
No
FIN

FIGURA 3.1
FLUJO DE PROCESOS DEL MODELO DE SIMULACIÓN

En resumen debido a la complejidad del modelo se vio la necesidad de buscar la manera ordenada de programar este modelo cumpliendo con todo el proceso productivo de las margarinas y sus limitantes. Se hizo uso del diagrama de flujo para identificar las operaciones de toma de decisiones y procesos que siguen la línea de producción. Se detalla la operación realizada, la entidad de entrada/salida y en que locación se realiza este proceso.

Cola de Separación
Entidad de Ingreso/Salida: PlanDiario
Locación: Cola_SepOrd
En esta locación no se realiza ninguna operación solo recibe las entidades generadas por el archivo de arribos ingresadas por el usuario.

Separación
Entidad de Ingreso: PlanDiario
Entidad de Salida: OrdenProduccionB1, OrdenProduccionB2
Locación: Separacion_Ordenes
En esta locación se realiza la separación del PlanDiario en órdenes de producción. En esta operación las órdenes se separarán en lotes de 3.8 ton (OrdenProduccionB1) o 2.8 ton (OrdenProduccionB2). Con esto se controla que los tanques no sobrepasen la capacidad máxima en el tanque de preparación (Premix1 y Premix2) y de almacenaje (Pulmón1 y Pulmón2). Se crea el número de órdenes de producción necesarias para cumplir con la cantidad definida en el atributo de la entidad de ingreso PlanDiario.

Cola de Planificación
Entidad de Ingreso/Salida: OrdenProduccionB1, OrdenProduccionB2
Locación: Cola_PlanificacionB1, Cola_PlanificacionB2
En esta locación no se realiza ninguna operación solo recibe las entidades y las mantiene en cola hasta que haya capacidad en la locación siguiente.

Planificación
Entidad de Ingreso/Salida: OrdenProduccionB1, OrdenProduccionB2
Locación: PlanificacionB1, PlanificacionB2
En esta locación se realizan dos operaciones. La primera por medio de atributo limpiezaB1 y limpiezaB2, evalúa si se requiere la limpieza de la línea y la segunda operación por medio de un filtro “if” verifica que el tanque de preparación esté disponible.
Si el tanque de preparación está disponible este continuará con la siguiente operación y si está ocupado, este hará esperar la entidad para luego ser llamada por el comando send.

Producción de Tanques
Entidad de Ingreso/Salida: OrdenProduccionB1, OrdenProduccionB2
Locación: ProduccionB1Tanques, ProduccionB2Tanques

En esta locación se hace la preparación de los taques. Dependiendo de la fórmula transfiere todos los ingredientes al tanque de preparación (Premix1, Premix2) y realiza la operación de mezcla. Luego de esto se
transfiere al tanque de almacenaje (Pulmón1, Pulmón2). Para que la emulsión sea transferida al tanque de almacenaje se deben cumplir una de dos condiciones: que el tanque este vacío o que el tanque contenga la misma emulsión que el tanque de almacenaje. Esto se realiza por medio de un filtro "Atributo Producto" comparado con las variables ValorActB1 y ValorActB2 que son los productos en el sistema. En el caso que ninguna de las dos condiciones se cumplan, la entidad espera hasta ser llamada por el comando send.

Vaciar de Tanque

Entidad de Ingreso/Salida: OrdenProduccionB1, OrdenProduccionB2

Locación: VaciadoTanqueB1, VaciadoTanqueB2

La primera operación que realiza esta locación es cambiar a la variable ValorActB1 o ValorActB2, esto sirve para llevar el control de la emulsión que circula por el sistema. Luego de esto corre un bucle hasta que el tanque de preparación este vacío.

Por último revisa si existen órdenes pendientes de atender en la operación Planificación, de ser así se la envía por medio del comando send desde la locación PlanificacionB1 y PlanificacionB2 hasta la
locación ProduccionB1Tanques y ProduccionB2Tanques para que se realice la preparación dado que el tanque de preparación se vació.

Producción

Entidad de Ingreso/Salida: OrdenProduccionB1, OrdenProduccionB2

Locación: ProduccionB1, ProduccionB2

Esta locación realiza la operación donde la emulsión de producto se convierte en entidades, básicamente convierte 1 kilo de emulsión en una entidad (Se convierte de continua a discreta). En el momento que se discretisa la emulsión pasa del Tanque Pulmón hasta el MPU. Luego crea la entidad por medio del comando ORDER en la locación Kombinator/Votator (MPU).

Esta es la última locación que maneja la parte continua del modelo luego de esto se comporta de manera discreta en entidades de 1 kilo de producto.

MPU

Entidad de Ingreso/Salida: Entidades de Producto

Locación: Kombinator, Votator

Se realiza la operación de cristalización que la cual demora 1.5 min para 1 kilo de producto.

Recirculación

Entidad de Ingreso/Salida: Entidades de Producto

Locación: RecirculacionB1, RecirculacionB2

Locación de almacenaje la cual mantiene en cola el producto hasta que pueda ser atendido por la envasadora.

Envasadora

Entidad de Ingreso/Salida: Entidades de Producto

Locación: B1, B2 y Em

Se realiza la operación final de envasado. El tiempo de envasado se lo obtuvo por

medio de las tasas manejadas al momento de la planificación afectadas por la eficiencia operacional. Esta ultima locación registra en
un array las horas en que salen los lotes completos esta información luego es exportada. Con esta información se realizó la validación.
Para mayor información de la programación del modelo de simulación de la planta de margarinas, se puede consultar el Apéndice donde se encuentra la programación escrita en el lenguaje y formato empleado por ProModel.

CAPÍTULO 4

4. VALIDACIÓN DEL MODELO DE SIMULACIÓN

4.1. Tipo de Simulación

Los tipos de modelos de simulación pueden ser clasificados en 2 grandes grupos iniciales, físicos y matemáticos. El modelo en estudio es de tipo matemático ya que utiliza ecuaciones y lógica en representación al proceso real.

Los modelos matemáticos pueden ser clasificarlos como: determinísticos o estocásticos y discretos o continuos. El modelo en estudio tiene una combinación de todos los tipos mencionados, ya que emplea variables determinísticas como aleatorias y representan tanto procesos discretos como continuos.

El modelo presentado de forma general es de tipo determinístico, ya que no posee es su mayoría variables aleatorias en sus operaciones, a excepción del tiempo de transferencia de los ingredientes al tanque preparador, el cual tuvo que ser modelado de acuerdo a una variable aleatoria ya que la empresa no tiene registros de este tiempo.

Para la operación de trasferencia se realiza una medición del flujo de cada material logrando obtener mediante un análisis estadístico de datos usando el programa STAT FIT, la distribución a la que pertenece cada ingrediente. Por ende la transferencia de aceites al tanque preparador es la operación que trabaja de manera estocástica en el modelo, las demás operaciones trabajan de manera determinística.

La simulación con tanques tiene como parámetro la tasa del flujo del líquido, esto hace que se tenga una simulación continua, pero se transforma en un modelo discreto al pasar a las siguientes operaciones.

La validación del modelo tiene como objetivo asegurar que al momento de correr y arrojar los resultados, se comporten de acuerdo al proceso real.

Este proceso es un ejercicio de prueba y error desarrollando a lo largo de la modelación, realizando ajustes a la lógica, agregando datos, incluso haciendo grandes cambios que pueden generar la transformación del modelo de simulación.

4.2. Selección de Variables de Respuesta

El objetivo de un estudio de simulación es, en definitiva, adquirir mayor conocimiento sobre el comportamiento de su sistema gracias a la exploración de cada una de las operaciones del proceso y el de las alternativas de mejora generadas por un conjunto restringido de factores, En particular nos interesará determinar el rendimiento global del sistema y de algunos de sus componentes a fin de establecer los criterios que nos conducirán a tomar una decisión determinada.

En la selección de variables de respuestas se trabajará con diferentes tipos de medidas, en los siguientes apartados se discutirá algunas de las variables seleccionadas a los objetivos planteados al inicio.

En este estudio nos referimos a diversos tipos de problemas, como la identificación de cuellos de botella en el proceso y su influencia en el rendimiento a largo plazo, y la comparación de alternativas propuestas de secuenciación y su influencia en el rendimiento del sistema.

Las variables de Respuesta e indicadores de desempeño seleccionados fueron:
- Tiempos de salida obtenidos por el modelo de simulación de cada una de las órdenes de producción en el mes a estudiar.

Esta variable permitirá comparar el tiempo de salida real de cada una de las cantidades fabricadas versus los tiempos de salida arrojados por el modelo. La comparación de estos valores permitirá validar si efectivamente el proceso en el modelo de simulación refleja los tiempos reales empleados durante la producción.

Para obtener esta variable de respuesta el modelo de simulación utilizó el atributo índice, este atributo sirve para identificar la salida de cada entidad que representa 1kg de cualquier producto, la función de este atributo es sumar cada entidad hasta completar la cantidad total
planificada, dando como respuesta la hora en que este plan se completa. El atributo índice es colocado en el archivo de arribos ingreso por el usuario.

- Porcentaje de Utilización de cada operación del proceso.

La utilización mide el tiempo ocupación de cada operación (Tanque de Preparación, Tanque Pulmón, MPU y Envasadora) versus el tiempo efectivo (tiempo total disponible para la producción sin considerar mantenimientos o

paradas planeadas).

Los porcentajes de utilización son arrojados por el programa al momento de realizar la corrida de todo el mes en estudio.

- La cantidad de desperdicio por cada cambio de formato, de acuerdo a las diferentes secuencias utilizadas en el estudio.

Esta es una de las variables de respuesta que permiten evaluar nuestras alternativas de mejora referentes a los cambios de secuenciación y no será utilizada para la validación del modelo. La
empresa en estudio tiene procedimientos para los cambios de formato de cada producto, estos cambios generan un costo los cuales serán evaluados con el objetivo de determinar la alternativa más eficiente para la compañía.

4.3. Selección de Indicadores

Los indicadores que se utilizaran son:
- Porcentaje de Utilización de cada operación del proceso.
- Diferencia de los tiempos de salida reales y los que obtuvieron por el modelo de simulación.

4.4. Comparación de Datos entre la Simulación y lo Real

Se realizó la simulación de un mes en particular de la empresa en estudio, para esto se obtuvo como información inicial la secuencia semanal de producción, las respectivas cantidades y los indicadores de productividad obtenidos en dicho mes (Eficiencia y Cumplimiento de planes de producción), cabe indicar que durante este mes fueron muy satisfactorios e incluso el nivel de servicio fue uno de los mejores del 2009.
La comparación de los datos obtenidos se los analizó en el programa MINITAB 14, a continuación se mostrará la tabla con los resultados.

Dado que se quiere demostrar que la simulación representa de manera confiable las operaciones de la planta se plantean las siguientes hipótesis para la línea B1:

H_0: No existe diferencia significativa entre los tiempos de salida promedio reales de la línea B1 y los obtenidos por el modelo.
H_1: La diferencia de las medias de los datos de salida reales de la línea B1 y los obtenidos por el modelo es diferente de cero.
$\mu_1 - \mu_2 = 0$ $\mu_1 - \mu_2 \neq 0$

One-Sample T: Diferencia B1

```
Test of mu = 0 vs. not = 0


Variable         N      Mean     StDev    SE Mean           95% CI             T
Diferencia B1   23  0.347826  1.070628  0.223241  (-0.115148, 0.810801)  1.56

Variable          P
Diferencia B1  0.133
```

TABLA 4.1
RESULTADO DE LAS PRUEBAS DE HIPÓTESIS PARA LA LÍNEA B1
Debido a que el P-Valor es mayor a 0.05 con una confianza del 95% se puede concluir que no existe suficiente evidencia estadística para rechazar la hipótesis nula. Es decir que la línea B1 en la simulación representa exitosamente la operación de la misma, en la planta.

FIGURA 4.1
DIAGRAMA DE CAJAS DE LA DIFERENCIA DE LOS TIEMPOS DE SALIDA REALES Y LOS OBTENIDOS POR EL MODELO PARA LA LINEA B1

En el grafico anterior se observa que las diferencias entre los tiempos oscilan entre cero y una hora. Al observar el intervalo de confianza podemos notar que el cero se encuentra incluido dentro del mismo lo que confirma lo obtenido en la prueba de hipótesis antes realizada
SimuladoB1Planificado B1800700600500400300200100Diagrama de Cajas para la comparación de la línea B1
FIGURA 4.2
DIAGRAMA DE CAJAS DE LOS TIEMPOS DE SALIDA Y OBSERVADOS EN EL MODELO DE SIMULACIÓN PARA LA LÍNEA B1

Este grafico presenta los dos diagramas de cajas de los datos reales y simulados de los tiempos de salida de la línea B1. Podemos observar que su diferencia es mínima.

Por último se demostrará que la simulación representa de manera confiable las operaciones de la planta se plantean las siguientes hipótesis para la línea B2:

H_0: No existe diferencia significativa entre los tiempos de salida promedio reales de la línea B2 y los obtenidos por el modelo.
H_1: La diferencia de las medias de los datos de salida reales de la línea B2 y los obtenidos por el modelo es diferente de cero.
$\mu_1 - \mu_2 = 0$ $\mu_1 - \mu_2 \neq 0$

One-Sample T: Diferencia B2

```
Test of mu = 0 vs. not = 0
```

Variable	N	Mean	StDev	SE Mean	95% CI	T
Diferencia B2	13	0.384615	0.869718	0.241217	(-0.140950, 0.910181)	1.59

Variable	P
Diferencia B2	0.137

TABLA 4.2
RESULTADO DE LA PRUEBA DE HIPÓTESIS PARA LA LÍNEA B2

Debido a que el P-Valor es mayor a 0.05 con una confianza del 95% se puede concluir que no existe suficiente evidencia estadística para rechazar la hipótesis nula. Es decir que la línea B2 en la simulación representa exitosamente la operación de la misma, en la planta.

FIGURA 4.3
DIAGRAMA DE CAJAS DE LA DIFERENCIA DE LOS TIEMPOS DE SALIDA REALES Y LOS OBTENIDOS POR EL MODELO PARA LA LÍNEA B2

En el grafico anterior se observa que las diferencias entre los tiempos oscilan entre cero y una hora. Al observar el intervalo de confianza podemos notar que el cero se encuentra incluido dentro del mismo lo que confirma lo obtenido en la prueba de hipótesis antes realizada

FIGURA 4.4
DIAGRAMA DE CAJAS DE LOS TIEMPOS DE SALIDA Y OBSERVADOS EN EL MODELO DE SIMULACIÓN PARA LA LÍNEA B2

Este grafico presenta los dos diagramas de cajas de los datos reales y simulados de los tiempos de salida de la línea B2. Se puede observar que su diferencia es mínima.

B1	74,27%	74,60%
Em	15,23%	14,43%
B2	32,39%	30,29%

TABLA 4.3
COMPARACIÓN DE LA HORA DE SALIDA ENTRE EL PLAN REAL Y EL PLAN SIMULADO

Al comparar los resultados de la utilización de los equipos se puede observar que los resultados en la realidad son muy similares a los obtenidos en el modelo de

simulación.

Al finalizar estos análisis se puede concluir que el modelo de simulación se comporta igual al proceso real para la línea B1 y para la línea B2; y podrá ser utilizado para resolver los objetivos planteados en este estudio.

CAPÍTULO 5

5. PLANTEAMIENTO Y VALIDACIÓN DE CAMBIOS

5.1. Modelación de Cambios

En el presente capítulo se procederá a plantear las alternativas de cambios o mejoras que nos permitan satisfacer los objetivos planteados inicialmente. Una vez evaluados los cambios, se presentará el impacto técnico como económico de los mismos.

Los cambios a analizar serán:

- Evaluación de diferentes métodos de secuenciamiento
- Evaluación de desempeño de cada operación a largo plazo, desplazando volumen a la línea B2

A continuación se detalla y se describe cada una de las alternativas planteadas

5.1.1. Evaluación de diferentes métodos de secuenciamiento

Los diferentes productos que maneja la empresa en estudio obligan a realizar un control y seguimiento a los planes de producciones anuales, mensuales y semanales (políticas de inventario negociadas entre los departamentos involucrados), con el objetivo de tener el producto en el momento y cantidad solicitada.

Un plan de secuenciamiento de producción tiene como objetivo final optimizar los cambios de formato y los tamaños de lote a producir. En esta sección se enfocará en la primera alternativa mencionada, en este caso se analizará cuál será la secuencia menor impacto considerando que un plan de producción semanal incluye todos los formatos en la línea de producción B1.

Los productos como se mencionó anteriormente están segmentados para diferentes mercados, por ende se enfocará a realizar el análisis por familia de producto, la primera familia es la B, como anteriormente se lo mencionó representan el 90% del volumen anual. Los productos D representan el 10% del volumen anual.

No se enfocará al secuenciamiento de productos entre las 2 familias, ya que está significativamente soportado y demostrado cual es la diferencia y que es lo que pesa al momento de secuenciar estos productos. Cada una de las familias de productos está compuesta por diferentes ingredientes de modo que para cambiar de formato entre un producto B a un producto D se generaría una actividad en el secuenciamiento de gran impacto. Esta actividad es una operación de limpieza y enjuague de 8 horas en todas las maquinarias utilizadas en el proceso, desde el inicio de la transformación de materia prima hasta el final, comenzando por la preparación hasta la línea de envasado. Por otro lado al realizar el cambio de producto desde la familia D hasta la B o entre productos de la misma familia solo se realizará un empuje, este empuje se produce al momento de pasar el siguiente producto por la tubería mezclando y arrastrando el contacto entre los productos, el producto fabricado lo arrastrará por las tuberías llevándolo al tanque de desperdicio, esta acción ha sido establecida para cada cambio, esto quiere decir que el operador de la línea tiene cronometrado el tiempo de arrastre hasta conseguir que la línea expulse el producto a fabricar, no el producto fabricado y la mezcla entre los dos.

Esta limpieza es una operación exigida por el comité de calidad, desarrollo y el de buenas prácticas de manufactura de la empresa, adicional a esto la limpieza de 8 horas en particular no afecta el índice de eficiencia por ende si en algún momento se requiere alternar el secuenciamiento desde productos B hacia productos D, esta limpieza estaría considerada como una actividad dentro del plan.

Como se mencionó en el capítulo anterior, la empresa cuenta con información completa de los diferentes cambios de formato entre los productos, el parámetro que mediremos en cada una de las diferentes secuencias será la cantidad de desperdicio junto con su respectivo costo, el cual se modelará para cada una de las diferentes opciones.

En la línea B1 la familia B posee moldes con formatos de 250 y 500 gramos, fabricando diferente productos, dando un total de 6 skus, la cual detallamos a continuación:

- BV250 y BV500
- BL250 y BL500

- BC250 y BC500

La familia D posee moldes con formatos 250 y 500 gramos, fabricando cantidad menor a la anterior familia de diferentes productos, dando un total de 3 skus, la cual detalla a continuación:

- DR250 y BR500
- DL250
-

TABLA 5.1
TABLA 5.1 COMBINACIONES DE LOS CAMBIOS DE FORMATOS ENTRE LOS SKUS

Como se puede ver en la tabla, hay ciertos parámetros para cada combinación de cambio de formatos. En los cambios de formatos existen dos perdidas el tiempo de espera y desperdicio, estas pérdidas pueden ser originadas por:
 La emulsión es la misma pero se cambia de un formato de 250 a 500 gramos o viceversa para este cambio se requiere una espera de 120 min.
 La emulsión cambia peroel formato se mantiene esto genera desperdicio y un tiempo de espera de 30 min.
 La emulsión cambia y el formato cambia de 250 a 500 gramos o viceversa generando un desperdicio y una espera de 120 min.

5.1.1.1. Cambios Incorporados en el Modelo de Simulación

Para este estudio fueron modeladas 3 diferentes opciones de secuenciamiento.

El objetivo será obtener de cada escenario de secuenciación la cantidad total de kilos y dólares de emulsión que se desperdiciarán en cada una de ellas y de este modo identificar la alternativa adecuada.

Los skus a fabricar en cada uno de los escenarios están colocados dentro de rectángulos que tendrán en la parte superior un círculo identificado con un número que indicará el orden en que se fabricarán.

A continuación se detalla las diferentes 6 alternativas de secuenciación que

analizará:

Para la familia D:
1. Modelo (DL250 – DR250 – DR500)

FIGURA 5.1 SECUENCIAMIENTO PARA EL 1ER MODELO PROPUESTO

2. Modelo (DR500 – DR250 – DL250)

FIGURA 5.2 SECUENCIAMIENTO PARA EL 2DO MODELO PROPUESTO

3. Modelo (DR500 – DL250 – DR250)

FIGURA 5.3 SECUENCIAMIENTO PARA EL 3ER MODELO PROPUESTO

Para la familia B:
4. Modelo (BL500 – BV500 – BC500 – BC250 – BV250 – BL250)

FIGURA 5.4 SECUENCIAMIENTO PARA EL 4TO MODELO PROPUESTO

5. Modelo (BC500 – BL500 – BV500 – BV250 – BL250 – BC250)

FIGURA 5.5 SECUENCIAMIENTO PARA EL 5TO MODELO PROPUESTO

6. Modelo (BL500 – BC500 – BV500 – BV250 – BC250 – BL250)

FIGURA 5.6 SECUENCIAMIENTO PARA EL 6TO MODELO PROPUESTO

5.1.1.2. Evaluación de Resultados

La información que se obtiene es la cantidad de emulsión que se desperdiciará por cada cambio de formato junto con su respectivo costo. A continuación los resultados arrojados por ProModel y validados con cada una de las secuencias propuestas.

Para el 1er modelo (DL250 – DR250 – DR500), los resultados fueron los siguientes:

B1	DL250	0	0	75,00	$ 0,86	$ 64,72
B1	DR250	1	30	75,00	$ 1,06	$ 79,20
B1	DR500	1	120	0,00	$ 1,06	$ 0,00
Total	2	150	150,00		$ 143,92	

TABLA 5.2 RESULTADOS DEL 1ER MODELO DE SECUENCIAMIENTO PROPUESTO

Para el 2do modelo (DR500 – DR250 – DL250), los resultados fueron los siguientes:

B1	DR500	0	0	0,00	$ 1,06	$ 0,00
B1	DR250	1	120	100,00	$ 1,06	$ 105,61
B1	DL250	1	30	100,00	$ 0,86	$ 86,29
Total	2	150	200,00		$ 191,89	

TABLA 5.3 RESULTADOS DEL 2DO MODELO DE SECUENCIAMIENTO PROPUESTO

Para el 3er modelo (DR500 – DL250 – DR250), los resultados fueron los siguientes:

B1	DR500	0	0	100,00	$ 1,06	$ 105,61
B1	DL250	1	120	175,00	$ 0,86	$ 151,00

B1	DR250	1	30	75,00	$ 1,06	$ 79,20
Total	2	150	350,00			$ 335,81

TABLA 5.4 RESULTADOS DEL 3ER MODELO DE SECUENCIAMIENTO PROPUESTO

Para todos los modelos planteados para la familia D tenemos el mismo tiempo perdido pero diferentes cantidades de producto desperdiciado, en relación a este parámetro la secuencia menos impactante es la 1ra opción debido a que es la más rentable en base a costos.

FIGURA 5.7 RESULTADOS DEL DESPERDICIO Y COSTOS GENERADOS POR LA FAMILA D

En los siguientes 3 modelos se incluyen los productos de la familia B, los resultados se muestran a continuación:

Para el 4to modelo (BL500 – BV500 – BC500 – BC250 – BV250 – BL250), los resultados fueron los siguientes:

B1	BL500	0	0	75,00	$ 0,79	$ 58,89
B1	BV500	1	30	175,00	$ 0,93	$ 162,44
B1	BC500	1	30	100,00	$ 0,91	$ 91,42
B1	BC250	1	120	100,00	$ 0,91	$ 91,42
B1	BV250	1	30	200,00	$ 0,93	$ 185,64
B1	BL250	1	30	100,00	$ 0,79	$ 78,53
Total	5	240	750,00			$ 668,34

TABLA 5.5 RESULTADOS DEL 4TO MODELO DE SECUENCIAMIENTO PROPUESTO

Para el 5to modelo (BC500 – BL500 – BV500 – BV250 – BL250 – BC250), los resultados fueron los siguientes:

B1	BC500	0	0	100,00	$ 0,91	$ 91,42
B1	BL500	1	30	175,00	$ 0,79	$ 137,42
B1	BV500	1	30	75,00	$ 0,93	$ 69,62
B1	BV250	1	120	100,00	$ 0,93	$ 92,82
B1	BL250	1	30	200,00	$ 0,79	$ 157,05
B1	BC250	1	30	100,00	$ 0,91	$ 91,42
Total	5	240	750,00			$ 639,75

TABLA 5.6 RESULTADOS DEL 5TO MODELO DE SECUENCIAMIENTO PROPUESTO

Para el 6to modelo (BL500 – BC500 – BV500 – BV250 – BC250 – BL250), los resultados fueron los siguientes:

B1	BL500	0	0	75,00	$ 0,79	$ 58,89
B1	BC500	1	30	175,00	$ 0,91	$ 159,98
B1	BV500	1	30	100,00	$ 0,93	$ 92,82
B1	BV250	1	120	100,00	$ 0,93	$ 92,82
B1	BC250	1	30	200,00	$ 0,91	$ 182,83
B1	BL250	1	30	100,00	$ 0,79	$ 78,53

Total	5	240	750,00	$ 665,88

TABLA 5.7 RESULTADOS DEL 6TO MODELO DE SECUENCIAMIENTO PROPUESTO

Para todos los modelos planteados para la familia B tenemos el mismo tiempo perdido y las mismas cantidades desperdiciadas al final del ejercicio, los costos por kilos de los productos son muy similares por ende no se ve una diferencia significante entre los modelos planteados. De todas maneras el modelo que genera menor costo es el 5to modelo.

Es importante comentar que los productos BC250 y BC500 a pesar de que están incluidos en la familia B (90% del volumen de la empresa) no representan un volumen considerable y en la práctica estos productos debido al lote mínimo de preparación se recomienda secuenciarlos uno a continuación de otro, el 5to y 6to modelo no satisfacen esta consideración únicamente lo hace el modelo 4to.

FIGURA 5.8 RESULTADOS DEL DESPERDICIO Y COSTOS GENERADOS POR LOS ESCENARIOS DE LA FAMILIA B

En conclusión considerando el costo más bajo y la posibilidad de secuenciar de manera continua los skus BC, se sugiere el empleo del 4to o 5to modelo evaluado.

5.1.2. Evaluación de desempeño de cada operación a largo plazo

Normalmente se utilizan archivos en Excel para evaluar el desempeño de una planta de producción en el corto, mediano y largo plazo, determinando si se tendrá la suficiente disponibilidad de capacidad en los equipos para abastecer los volúmenes que proyecta la empresa vender en el futuro. Esta es una limitada manera de realizar análisis dado que se centra en las operaciones restrictivas y no en toda la operación. Lo que puede ocasionar que las inversiones realizadas para aumentar capacidad no resulten como se esperan. Adicionalmente esta forma de análisis no permite evaluar diferentes escenarios que por experiencia de la planta se podrían presentar.

A continuación se muestra un resumen del escenario a largo plazo de la planta en el cual se observa la utilización de la línea B1 a medida que el volumen de producto se incrementa.

Para la línea B1:

2009	7762,5	74,3%
2010	8218,7	78,7%

2011	8681,8	83,1%
2012	9243,0	88,5%

TABLA 5.8 TABLA DE RESULTADOS DE LA SITUACIÓN ACTUAL PARA LA LINEA B1

FIGURA 5.9 RESULTADOS DE LA SITUACIÓN ACTUAL PARA LA LINEA B1

Analizando el cuadro vemos una tendencia elevada de crecimiento en la capacidad de utilizada de la línea de envasado, llegando incluso al nivel máximo permisible por una planta de manufactura (80%). Comprometerse con un nivel de utilización igual o mayor al 80% mostrado en la gráfica anterior a partir del 2011, es muy riesgoso ya que las plantas están sometidas a eventos extraordinarios externos no controlables y las recomendaciones es que las líneas envasadoras tengan copado no más del 80% de su capacidad.

Para la línea B2 y Em:

2009	3145,3	47,5%
2010	3191,6	48,4%
2011	3239,1	49,4%
2012	3287,8	50,4%

TABLA 5.9 TABLA DE RESULTADOS DE LA SITUACIÓN ACTUAL PARA LA LINEA B2 Y EM

FIGURA 5.10 RESULTADOS DE LA SITUACIÓN ACTUAL PARA LA LINEA B2 Y EM

Para este caso, el volumen y utilización de las líneas es baja, por ende no existen problemas de capacidad a largo plazo y manufactura podría comprometerse con el volumen solicitado sin que existan inconvenientes.

A continuación se muestra un gráfico resumen donde están incluidas todas las líneas de envasado:

FIGURA 5.11 RESULTADOS DE LA SITUACIÓN ACTUAL PARA TODAS LAS LÍNEAS DE PRODUCCIÓN

Al evidenciar el problema de capacidad a largo plazo, este subcapítulo entregará a la empresa una idea sustentada que permitirá satisfacer en incremento en la demanda en el futuro.

Iniciando el análisis, las líneas de envasado B2 junto a la Em tienen un porcentaje de utilización que no pasa del 60%, por ende el objetivo será trasladar volumen de la línea B1 a la B2.

Con esta idea lo que se logrará es descongestionar la máquina B1 y trasladar parte del volumen a la máquina B2 bajando el nivel de utilización del equipo con problemas de capacidad.

5.1.2.1. Cambios Incorporados en el Modelo de Simulación

La B2 no posee moldes para fabricar ninguna de las 2 presentaciones que tiene la B1 (250 y 500gr) solo posee moldes para fabricar productos de 1000gr. Por esta razón se usará el modelo de simulación para agregar el formato de 500 gr a la línea B2 y de esta manera evaluar como esta idea repercutirá en el desempeño de las líneas de envasado y de todas las operaciones de proceso productivo.

Antes de realizar cualquier cambio para modelar esta alternativa en el modelo de simulación se procedió a comprobar la tasa natural y efectiva del producto de 500gr procesado en la línea B2. Esto se realizó junto con el equipo de proyectos y tecnología de la empresa los cuales poseen información de plantas similares en otros países pertenecientes a Productos Varios S.A.

La repartición de los volúmenes se la realizó por prueba y error hasta dar con un porcentaje de utilización aceptable para cada una de las líneas de envasado.

En cuanto al modelo de simulación, se creó la entidad BV500 para la línea B2, se traspasó volumen de la línea B1 a la B2 hasta llegar a un nivel de utilización aceptable para la línea B1 sin perjudicar dicho nivel en la B2.

5.1.2.2. Evaluación de Resultados

Una vez validado el modelo de simulación en el capítulo 4, se continuará mostrando la capacidad utilizada y disponible de: tanques de preparación, tanques de almacenamiento, MPUs y envasadoras, arrojada por el modelo con volúmenes de producción proyectados al 2012 antes de realizar los cambios.

Premix1	51,9%	48,1%

Pulmon1	49,7%	50,3%
Kombinator	48,8%	51,2%
B1	87,3%	12,7%
Premix2	20,1%	79,9%
Pulmon2	16,6%	83,4%
Votator	21,4%	78,6%
Em	15,1%	84,9%
B2	35,0%	65,0%

TABLA 5.10 RESULTADOS DEL PORCENTAJE DE UTILIZACIÓN AL 2012 DE TODAS LAS OPERACIONES DEL PROCESO PRODUCTIVO CON EL ESCENARIO ACTUAL

Analizando los datos podemos concluir que la línea de envasado B1 presentará problemas de capacidad al año 2012 llegando al 87% de utilización de su capacidad, mientras que el tanque Pulmón 1 como el Premix 1 y MPU Kombinator no lo tendrán.

Las líneas B2 y Em como ya se dijo presentan un nivel bajo de utilización, por ende a estas operaciones les sobra capacidad disponible.

Una vez realizados los cambios propuestos para sustentar los volúmenes a largo plazo, se mostrará la capacidad utilizada y disponible en la línea B1 de: tanques de preparación, tanques de almacenamiento, MPUs y envasadoras, arrojadas por el modelo con volúmenes de producción proyectados al 2012:

Premix1	46,0%	54,0%
Pulmon1	43,8%	56,2%
Kombinator	42,9%	57,1%
B1	77,9%	22,1%
Premix2	28,2%	71,8%
Pulmon2	25,1%	74,9%

Votator	32,4%	67,6%
Em	15,7%	84,3%
B2	53,6%	46,4%

TABLA 5.11 RESULTADOS DEL PORCENTAJE DE UTILIZACIÓN AL 2012 DE TODAS LAS OPERACIONES DEL PROCESO PRODUCTIVO CON LOS CAMBIOS INCORPORADOS

Para una mejor ilustración de la comparación de los datos, los se mostrará en una misma tabla a continuación:

Premix1	51,9%	46,0%
Pulmon1	49,7%	43,8%
Kombinator	48,8%	42,9%
B1	87,3%	77,9%
Premix2	20,1%	28,2%
Pulmon2	16,6%	25,1%
Votator	21,4%	32,4%
Em	15,1%	15,7%
B2	35,0%	53,6%

TABLA 5.12 COMPARACIÓN DE RESULTADOS DEL PORCENTAJE DE UTILIZACIÓN DE CADA OPERACIÓN DEL PROCESO PRODUCTIVO ENTRE EL ESCENARIO ACTUAL Y EL PROPUESTO.

Analizando cada uno de los datos, vemos una reducción en los niveles de utilización de la línea de envasado B1 y de sus operaciones anteriores, dado un nivel aceptable de utilización de los equipos.

Se observa un incremento en las utilizaciones de las líneas B2, Em, tanque de

preparación, tanque de almacenamiento y MPU. Sin embargo este incremento en su producción no generará ningún tipo de riesgo. Incluso para las líneas B2 y Em las utilizaciones suman 69,3% lo cual no representa una restricción, a continuación se mostrará un gráfico con la capacidad utilizada con los cambios incorporados:

FIGURA 5.12 RESULTADOS DEL PROCESO PRODUCTIVO CON EL ESCENARIO PROPUESTO.

Los resultados de esta alternativa para resolver el problema a futuro están evidenciados por este simulador que incluso muestra aún más tiempo disponible para un flexible crecimiento de la categoría.

CAPÍTULO 6

6. ANÁLISIS DE LA INVERSIÓN FINANCIERA

En este capítulo procederemos a desarrollar un análisis costo-beneficio de las alternativas de mejora planteadas en el capítulo anterior.

Las alternativas de mejora a evaluar financieramente, pueden ser clasificadas de 2 maneras:

- Métodos de secuenciamiento
- Evaluación de desempeño de cada operación a largo plazo, desplazando volumen a la línea B2

Los métodos de secuenciamiento propuestos, no requieren ningún tipo de inversión, sin embargo la forma en que los materiales serán ingresados de acuerdo a un método de secuenciamiento u otro, pueden generar diferentes costos los cuales deben ser analizados. A continuación se presenta con detalle cada uno de estos casos.

6.1. Cuantificación del Impacto de las Alternativas de Mejora

Para la primera alternativa de mejora, en el quinto capítulo dejamos a consideración de la empresa en utilizar entre el 4to (BL500 – BV500 – BC500 –

BC250 – BV250 – BL250) y 5to (BC500 – BL500 – BV500 – BV250 – BL250 – BC250) escenario.

La propuesta a utilizar el 5to escenario es por tener el menor costo en desperdicio de emulsión, pero el 4to escenario es por tener los skus BC250 y BC500 que poseen ingredientes que por su naturaleza deben ser utilizados inmediatamente por lo tanto estos deben ser necesariamente producidos en secuencia, si esto no sucede el ingrediente no podrá ser utilizado para la producción.

Este ingrediente es muy sensible al ambiente y es necesario que al abrirlo sea utilizado de forma inmediata caso contrario sus propiedades organolépticas que lo constituyen se pierden, y el ingrediente no podrá ser utilizado para la preparación de la emulsión, este ingrediente es despachado en bidones de 950 kilos y utilizado en los skus BC250 y BC500.

El estimado mensual de cada uno de estos 2 skus es muy pequeño y en la actualidad se los fabrica uno seguido de otro para poder consumir todo el lote del ingrediente, llegando a la cobertura de producto terminado solicitada y no desperdiciando materia prima.

De acuerdo al 5to escenario, el menos costoso (BC500 – BL500 – BV500 – BV250 – BL250 – BC250), los 2 skus (BC250 y BC500) fabricándolos por separado podrán consumir un bidón del ingrediente cada uno. Esto generará 3 meses stock de producto terminado de cada uno de los skus, lo cual incrementará el capital de trabajo (dinero en inventario) y pondrá en riesgo el producto terminado que tiene como tiempo de vida útil 3 meses.

A continuación se detallará el desperdicio generado del ingrediente cuando este es consumido por los skus BC250 y BC500 cuando se encuentran en un mismo escenario pero no secuenciados inmediatamente. Considerando que se utilizará el 50% del bidón del ingrediente, se generarán 1,5 meses de inventario de producto terminado de cada uno de los skus.

BC 250	950	475	$ 0,10	$ 47,50
BC 500	950	475	$ 0,10	$ 47,50
			Total	$ 95,00

TABLA 6.1 CUANTIFICACIÓN DEL DESPERDICIO GENERADO AL NO ESTAR

SECUENCIADOS INMEDIATAMENTE LOS SKUS BC250 Y BC500

Por lo cual la empresa si decide utilizar el escenario donde los skus BC250 y BC500 no se encuentran en secuencia deberá considerar $ 95,00 adicionales en desperdicio de materiales.

La segunda alternativa de mejora, evaluación de desempeño de cada operación a largo plazo, desplazando volumen a la línea B2, será posible únicamente mediante la instalación de un kit que contenga moldes para poder fabricar los productos con los formatos de 500gr que son los moldes que hemos identificado serán trasladados a la B2.

A continuación se detallará el impacto del kit de moldes

Kit de 500gr	1	$ 9.750,00
Mano de Obra	1	$ 5.000,00
	Total	$ 14.750,00

TABLA 6.2 TABLA RESUMEN DEL COSTO TOTAL PARA LA INSTALACIÓN DEL KIT DE 500GR

6.2. Evaluación de la Inversión Financiera

Para la primera alternativa, métodos de secuenciamiento, a continuación se mostrará el análisis financiero que presenta el costo total por cambio de formato de acuerdo a cada método de secuenciamiento.

A continuación se mostrará los resultados de los 2 escenarios en comparación:

4to	(BC500 – BL500 – BV500 – BV250 – BL250 – BC250)	$ 668,34
5to	(BL500 – BC500 – BV500 – BV250 – BC250 – BL250)	$ 639,75

TABLA 6.3 RESULTADO DEL COSTO DE CAMBIO DE FORMATO DE LOS ESCENARIOS EN COMPARACIÓN

Incluyendo el desperdicio generado por los skus BC250 y BC500 cuando se encuentran en un mismo escenario pero que uno no está secuenciado inmediatamente después del otro por el ingrediente es de:

4to	(BL500 – BV500 – BC500 – BC250 – BV250 – BL250)	$ 668,34	$ 0,00	$ 668,34
5to	(BL500 – BC500 – BV500 – BV250 – BC250 – BL250)	$ 639,75	$ 95,00	$ 734,75

TABLA 6.4 TABLA RESUMEN DE LOS ESCENARIOS COMPARADOS INCLUYENDO EL DESPERDICIO DEL MATERIAL EXCLUSIVO

Finalmente podemos concluir que el 4to escenario (BL500 – BV500 – BC500 – BC250 – BV250 – BL250), es lo más recomendable para el negocio al momento de secuenciar.

Referente a la segunda alternativa, evaluación de desempeño de cada operación a largo plazo, se mostrará a continuación un cuadro donde se detallará la cantidad de producto terminado que se dejaría de vender por no tener capacidad en la línea B1 en caso de no incurrir en la inversión del kit de 500gr para la línea B2.

Volumen Restringido	Ton	355,0	881,1	1236,1	$ 2.800,0	$ 3.461.005,1

TABLA 6.5 TABLA DE RESULTADOS DEL COSTO DE OPORTUNIDAD SI NO SE REALIZA LA INVERSIÓN DE LOS MOLDES.

Mediante esta tabla se puede evidenciar la cantidad de producto restringido por no realizar la inversión y compra del kit de 500gr, y la gran cantidad de ingresos que perdería por no realizarla hasta el 2012.

El negocio iniciaría el 2011 con problemas por falta de capacidad llegando al 2012 con un acumulado de 1236 toneladas, esto es un volumen considerable que se dejaría de vender al mercado lo cual podría generar a demás de clientes insatisfechos, el ingreso de la competencia.

Para finalizar este capítulo se puede evidenciar en base a costo-beneficio la recomendación de la compra e instalación del kit de 500gr para la línea B2.

CAPÍTULO 7

7. CONCLUSIONES Y RECOMENDACIONES

▪ A través de la simulación se pudo conocer el estado actual de cada una de las operaciones del proceso productivo de la planta de margarinas y evaluar si existen operaciones restrictivas que afecta el desempeño del sistema.

▪ Usando los volúmenes de demanda al 2012 entregados por la organización, fue posible identificar las potenciales operaciones con capacidad restringida y al mismo tiempo conocer aquellas que no podrán ser consideradas limitantes en el futuro.

▪ Uno de los resultados finales del estudio fue el planteamientos de la opción de realizar una inversión de un kit de moldes para pasar volumen de una línea con problemas de capacidad (B1) a otra que no los tiene (B2). La inversión sería $14,000 (€ 9536), con el fin de evitar perder

ventas que superan los $3, 500,000 (€ 2, 384,196) en producto terminado.

▪ Se sugiere realizar un estudio del impacto por la pérdida de participación en el mercado, si es que la empresa no decide realizar la inversión con el fin de pasar volumen entre líneas de producción.

▪ El modelo permitió simular varios escenarios con el cual se identificó el

secuenciamiento mas adecuado al evaluar el costo de la emulsión y desperdicio.

- Se logró crear un modelo de simulación continua-discreta, representado de manera efectiva el proceso real. Para el cual se hacen uso de locaciones ficticias y diagramas de flujo.

- El tener un departamento de planificación capacitado en la aplicación de un software de simulación puede llevar a la organización a la obtención de beneficios basados en el desarrollo de proyectos que

pueden ir desde el mejoramiento del servicio al cliente, la identificación de cuellos de botella y la obtención de ahorros en toda la cadena de suministro.

APENDICES

Entity...	Prior Entity...	Logic...	Disable
B_V500	B_V250	wait 120 min□□	No
B_L250	B_V250	wait 30 min	No
B_L500	B_V250	wait 120 min	No
B_C250	B_V250	wait 30 min	No
B_C500	B_V250	wait 120 min	No
D_R250	B_V250	DesperdicioBl = DesperdicioBl + 200	No
D_R500	B_V250	DesperdicioBl = DesperdicioBl + 200	No
D_L250	B_V250	DesperdicioBl = DesperdicioBl + 200	No
D_L500	B_V250	DesperdicioBl = DesperdicioBl + 200	No
B_L500	B_L250	wait 120 min	No
B_C250	B_L250	wait 30 min	No
B_C500	B_L250	wait 120 min	No
D_R250	B_L250	DesperdicioBl = DesperdicioBl + 200	No
D_R500	B_L250	DesperdicioBl = DesperdicioBl + 200	No
D_L250	B_L250	DesperdicioBl = DesperdicioBl + 200	No
D_L500	B_L250	DesperdicioBl = DesperdicioBl + 200	No
B_V250	B_L250	DesperdicioBl = DesperdicioBl + 150	No
B_V500	B_L250	wait 120 min	No
B_C500	B_C250	wait 120 min	No
D_R250	B_C250	DesperdicioBl = DesperdicioBl + 200	No
D_R500	B_C250	DesperdicioBl = DesperdicioBl + 200	No
D_L250	B_C250	DesperdicioBl = DesperdicioBl + 200	No
D_L500	B_C250	DesperdicioBl = DesperdicioBl + 200	No
B_V250	B_C250	wait 30 min	No
B_V500	B_C250	wait 120 min	No
B_L250	B_C250	wait 30 min	No
B_L500	B_C250	wait 120 min	No
D_R500	D_R250	wait 120 min	No
D_L250	D_R250	wait 30 min	No
D_L500	D_R250	wait 120 min	No
B_V250	D_R250	wait 30 min	No
B_V500	D_R250	wait 120 min	No
B_V500	D_R250	wait 120 min□□DesperdicioBl = DesperdicioB	No
B_L250	D_R250	wait 30 min	No
B_L500	D_R250	wait 120 min	No
B_C250	D_R250	wait 30 min	No
B_C500	D_R250	wait 120 min	No
D_L500	D_L250	wait 120 min	No
B_V250	D_L250	wait 30 min	No
B_V500	D_L250	wait 120 min	No
B_L250	D_L250	wait 30 min	No
B_L500	D_L250	wait 120 min	No
B_C250	D_L250	wait 30 min	No
B_C500	D_L250	wait 120 min	No
D_R250	D_L250	DesperdicioBl = DesperdicioBl + 150	No
D_R500	D_L250	wait 120 min	No
B_L250	B_V500	wait 120 min	No
B_L500	B_V500	wait 30 min	No
B_C250	B_V500	wait 120 min	No
B_C500	B_V500	wait 30 min	No
D_R250	B_V500	DesperdicioBl = DesperdicioBl + 200	No
D_R500	B_V500	DesperdicioBl = DesperdicioBl + 200	No
D_L250	B_V500	DesperdicioBl = DesperdicioBl + 200	No
D_L500	B_V500	DesperdicioBl = DesperdicioBl + 200	No
B_V250	B_V500	wait 120 min	No
B_C250	B_L500	wait 120 min	No
B_C500	B_L500	DesperdicioBl = DesperdicioBl + 200	No
D_R250	B_L500	DesperdicioBl = DesperdicioBl + 200	No
D_R500	B_L500	DesperdicioBl = DesperdicioBl + 200	No
D_L250	B_L500	DesperdicioBl = DesperdicioBl + 200	No
D_L500	B_L500	DesperdicioBl = DesperdicioBl + 200	No
B_V250	B_L500	wait 120 min	No
B_V500	B_L500	DesperdicioBl = DesperdicioBl + 150	No
B_L250	B_L500	wait 120 min	No
B_V500	D_R250	wait 120 min	No

B_L250	B_L500	wait 120 min□□DesperdicioBl = DesperdicioB	No
D_R250	B_C500	DesperdicioBl = DesperdicioBl + 200	No
D_R500	B_C500	DesperdicioBl = DesperdicioBl + 200	No
D_L250	B_C500	DesperdicioBl = DesperdicioBl + 200	No
D_L500	B_C500	DesperdicioBl = DesperdicioBl + 200	No
B_V250	B_C500	wait 120 min	No
B_V500	B_C500	wait 30 min	No
B_L250	B_C500	wait 120 min	No
B_L500	B_C500	wait 30 min	No
B_C250	B_C500	wait 120 min	No
D_L250	D_R500	wait 120 min	No
D_L500	D_R500	wait 30 min	No
B_V250	D_R500	wait 120 min	No
B_V500	D_R500	wait 30 min	No
B_L250	D_R500	wait 120 min	No
B_L500	D_R500	wait 30 min	No
B_C250	D_R500	wait 120 min	No
B_C500	D_R500	wait 30 min	No
D_R250	D_R500	wait 120 min	No
B_V250	D_L500	wait 120 min	No
B_V500	D_L500	wait 30 min	No
B_L250	D_L500	wait 120 min	No
B_L500	D_L500	wait 30 min	No
B_C250	D_L500	wait 120 min	No
B_C500	D_L500	wait 30 min	No
D_R250	D_L500	wait 120 min	No
D_R500	D_L500	wait 30 min	No
D_L250	D_L500	wait 120 min	No

```
********************************************************************************
*                                                                              *
*                    Formatted Listing of Model:                               *
* C:\Documents and Settings\Santiago\My Documents\Oscar\Tesis\Tesis Spreads OPP 2-08-09\Simulación\Alternativa
1\SecuenciamientoD1.MOD *
*                                                                              *
********************************************************************************

  Time Units:                   Minutes
  Distance Units:                Meters
  Initialization Logic:          ACTIVATE LlenadoTanksI1
                              ACTIVATE LlenadoTanksI2
                              ACTIVATE LlenadoTanksI3
                              ACTIVATE LlenadoTanksI4
                              ACTIVATE LlenadoTanksI5
                              ACTIVATE LlenadoTanksI6
                              ACTIVATE LlenadoTanksI7
                              ACTIVATE LlenadoTanksI9
                              ACTIVATE LlenadoTanksI8
                              ACTIVATE LlenadoTanksI10


  Termination Logic:             // Updates tank statistics to account for final state and contents
                              int Tank_StatIndex = 1
                              while Tank_StatIndex <= 100 do
                               {
                               Tank_UpdateStats(Tank_StatIndex)
                               Tank_SetState(Tank_StatIndex, Tank_Idle)
                               Inc Tank_StatIndex
                               }



********************************************************************************
*                              Locations                                       *
********************************************************************************

  Name               Cap   Units Stats       Rules      Cost
  ------------------- ----- ----- ----------- ---------- ------------
  I1                 50000 1     Time Series Oldest, ,
  I5                 30000 1     Time Series Oldest, ,
  I4                 30000 1     Time Series Oldest, ,
  I3                 20000 1     Time Series Oldest, ,
  I2                 20000 1     Time Series Oldest, ,
  I7                 100   1     Time Series Oldest, ,
  I9                 2400  1     Time Series Oldest, ,
  I10                1000  1     Time Series Oldest, ,
  I8                 4400  1     Time Series Oldest, ,
  Premix1             3800  1     Time Series Oldest, ,
  Pulmon1             3800  1     Time Series Oldest, ,
  Cola_Kombinator      inf   1     Time Series Oldest, ,
  Kombinator          1     1     Time Series Oldest, ,
  RecirculacionB1      1000  1     Time Series Oldest, ,
  B1                 1     1     Time Series Oldest, ,
  Premix2             2800  1     Time Series Oldest, ,
  Pulmon2             2800  1     Time Series Oldest, ,
  cola_Votator         inf   1     Time Series Oldest, ,
  Votator            1     1     Time Series Oldest, ,
  recirculacionB2      1000  1     Time Series Oldest, ,
  ventanaB2           1     1     Time Series Oldest, ,
  B2                 1     1     Time Series Oldest, ,
  Em                 1     1     Time Series Oldest, ,
  BPTB1               Inf   1     Time Series Oldest, ,
  BPTB2               Inf   1     Time Series Oldest, ,
  BPTEm               Inf   1     Time Series Oldest, ,
  I6                 50000 1     Time Series Oldest, ,
  Cola_SepOrd          inf   1     Time Series Oldest, ,
  Separacion_Ordenes   1     1     Time Series Oldest, ,
```

```
Cola_PlanificacionB1 inf   1     Time Series Oldest, ,
Cola_PalnificacionB2 inf   1     Time Series Oldest, ,
PlanificacionB1      1     1     Time Series Oldest, ,
ControldePlanB1      inf   1     Time Series Oldest, ,
PlanificacionB2      1     1     Time Series Oldest, ,
CnotroldePlanB2      inf   1     Time Series Oldest, ,
ProduccionB1Tanques  1     1     Time Series Oldest, ,
ProduccionB2Tanques  1     1     Time Series Oldest, ,
VaciadoTanqueB1      1     1     Time Series Oldest, ,
VaciadoTanqueB2      1     1     Time Series Oldest, ,
ProduccionB1         1     1     Time Series Oldest, ,
ProduccionB2         1     1     Time Series Oldest, ,


*****************************************************************************
*                      Setup downtimes for Locations                        *
*****************************************************************************

Loc        Entity     Prior Entity Logic
---------- ---------- ------------ -------------------
B1         B_Vital500 B_Vital250   wait 120 min

           B_Light250 B_Vital250   wait 30 min
                                   DesperdicioB1 = DesperdicioB1 + 200
           B_Light500 B_Vital250   wait 120 min
                                   DesperdicioB1 = DesperdicioB1 + 200
           B_Canola250 B_Vital250   wait 30 min
                                    DesperdicioB1 = DesperdicioB1 + 200
           B_Canola500 B_Vital250   wait 120 min
                                    DesperdicioB1 = DesperdicioB1 + 200
           D_Regular250 B_Vital250   DesperdicioB1 = DesperdicioB1 + 200
           D_Regular500 B_Vital250   DesperdicioB1 = DesperdicioB1 + 200
           D_Light250 B_Vital250   DesperdicioB1 = DesperdicioB1 + 200
           D_Light500 B_Vital250   DesperdicioB1 = DesperdicioB1 + 200
           B_Light500 B_Light250   wait 120 min

           B_Canola250 B_Light250   wait 30 min
                                    DesperdicioB1 = DesperdicioB1 + 200
           B_Canola500 B_Light250   wait 120 min
                                    DesperdicioB1 = DesperdicioB1 + 200
           D_Regular250 B_Light250   DesperdicioB1 = DesperdicioB1 + 200
           D_Regular500 B_Light250   DesperdicioB1 = DesperdicioB1 + 200
           D_Light250 B_Light250   DesperdicioB1 = DesperdicioB1 + 200
           D_Light500 B_Light250   DesperdicioB1 = DesperdicioB1 + 200
           B_Vital250 B_Light250   DesperdicioB1 = DesperdicioB1 + 150
           B_Vital500 B_Light250   wait 120 min
                                   DesperdicioB1 = DesperdicioB1 + 150
           B_Canola500 B_Canola250  wait 120 min
           D_Regular250 B_Canola250  DesperdicioB1 = DesperdicioB1 + 200
           D_Regular500 B_Canola250  DesperdicioB1 = DesperdicioB1 + 200
           D_Light250 B_Canola250  DesperdicioB1 = DesperdicioB1 + 200
           D_Light500 B_Canola250  DesperdicioB1 = DesperdicioB1 + 200
           B_Vital250 B_Canola250  wait 30 min
                                   DesperdicioB1 = DesperdicioB1 + 200
           B_Vital500 B_Canola250  wait 120 min
                                   DesperdicioB1 = DesperdicioB1 + 200
           B_Light250 B_Canola250  wait 30 min
                                   DesperdicioB1 = DesperdicioB1 + 200
           B_Light500 B_Canola250  wait 120 min
                                   DesperdicioB1 = DesperdicioB1 + 200
           D_Regular500 D_Regular250 wait 120 min

           D_Light250 D_Regular250 wait 30 min
                                   DesperdicioB1 = DesperdicioB1 + 200
           D_Light500 D_Regular250 wait 120 min
                                   DesperdicioB1 = DesperdicioB1 + 200
           B_Vital250 D_Regular250 wait 30 min
                                   DesperdicioB1 = DesperdicioB1 + 200
           B_Vital500 D_Regular250 wait 120 min
                                   DesperdicioB1 = DesperdicioB1 + 200
```

B_Light250 D_Regular250 wait 30 min
DesperdicioB1 = DesperdicioB1 + 200
B_Light500 D_Regular250 wait 120 min
DesperdicioB1 = DesperdicioB1 + 200
B_Canola250 D_Regular250 wait 30 min
DesperdicioB1 = DesperdicioB1 + 200
B_Canola500 D_Regular250 wait 120 min
DesperdicioB1 = DesperdicioB1 + 200
D_Light500 D_Light250 wait 120 min
B_Vital250 D_Light250 wait 30 min
DesperdicioB1 = DesperdicioB1 + 200
B_Vital500 D_Light250 wait 120 min
DesperdicioB1 = DesperdicioB1 + 200
B_Light250 D_Light250 wait 30 min
DesperdicioB1 = DesperdicioB1 + 200
B_Light500 D_Light250 wait 120 min
DesperdicioB1 = DesperdicioB1 + 200
B_Canola250 D_Light250 wait 30 min
DesperdicioB1 = DesperdicioB1 + 200
B_Canola500 D_Light250 wait 120 min
DesperdicioB1 = DesperdicioB1 + 200
D_Regular250 D_Light250 DesperdicioB1 = DesperdicioB1 + 150
D_Regular500 D_Light250 wait 120 min
DesperdicioB1 = DesperdicioB1 + 150
B_Light250 B_Vital500 wait 120 min
DesperdicioB1 = DesperdicioB1 + 200
B_Light500 B_Vital500 wait 30 min
DesperdicioB1 = DesperdicioB1 + 200
B_Canola250 B_Vital500 wait 120 min
DesperdicioB1 = DesperdicioB1 + 200
B_Canola500 B_Vital500 wait 30 min
DesperdicioB1 = DesperdicioB1 + 200
D_Regular250 B_Vital500 DesperdicioB1 = DesperdicioB1 + 200
D_Regular500 B_Vital500 DesperdicioB1 = DesperdicioB1 + 200
D_Light250 B_Vital500 DesperdicioB1 = DesperdicioB1 + 200
D_Light500 B_Vital500 DesperdicioB1 = DesperdicioB1 + 200
B_Vital250 B_Vital500 wait 120 min

B_Canola250 B_Light500 wait 120 min
DesperdicioB1 = DesperdicioB1 + 200
B_Canola500 B_Light500 DesperdicioB1 = DesperdicioB1 + 200
D_Regular250 B_Light500 DesperdicioB1 = DesperdicioB1 + 200
D_Regular500 B_Light500 DesperdicioB1 = DesperdicioB1 + 200
D_Light250 B_Light500 DesperdicioB1 = DesperdicioB1 + 200
D_Light500 B_Light500 DesperdicioB1 = DesperdicioB1 + 200
B_Vital250 B_Light500 wait 120 min
DesperdicioB1 = DesperdicioB1 + 150
B_Vital500 B_Light500 DesperdicioB1 = DesperdicioB1 + 150
B_Light250 B_Light500 wait 120 min
DesperdicioB1 = DesperdicioB1 + 200
D_Regular250 B_Canola500 DesperdicioB1 = DesperdicioB1 + 200
D_Regular500 B_Canola500 DesperdicioB1 = DesperdicioB1 + 200
D_Light250 B_Canola500 DesperdicioB1 = DesperdicioB1 + 200
D_Light500 B_Canola500 DesperdicioB1 = DesperdicioB1 + 200
B_Vital250 B_Canola500 wait 120 min
DesperdicioB1 = DesperdicioB1 + 200
B_Vital500 B_Canola500 wait 30 min
DesperdicioB1 = DesperdicioB1 + 200
B_Light250 B_Canola500 wait 120 min
DesperdicioB1 = DesperdicioB1 + 200
B_Light500 B_Canola500 wait 30 min
DesperdicioB1 = DesperdicioB1 + 200
B_Canola250 B_Canola500 wait 120 min
D_Light250 D_Regular500 wait 120 min
DesperdicioB1 = DesperdicioB1 + 200
D_Light500 D_Regular500 wait 30 min
DesperdicioB1 = DesperdicioB1 + 200
B_Vital250 D_Regular500 wait 120 min
DesperdicioB1 = DesperdicioB1 + 200
B_Vital500 D_Regular500 wait 30 min

```
                    DesperdicioB1 = DesperdicioB1 + 200
          B_Light250 D_Regular500 wait 120 min
                    DesperdicioB1 = DesperdicioB1 + 200
          B_Light500 D_Regular500 wait 30 min
                    DesperdicioB1 = DesperdicioB1 + 200
          B_Canola250 D_Regular500 wait 120 min
                     DesperdicioB1 = DesperdicioB1 + 200
          B_Canola500 D_Regular500 wait 30 min
                     DesperdicioB1 = DesperdicioB1 + 200
          D_Regular250 D_Regular500 wait 120 min
          B_Vital250 D_Light500   wait 120 min
                    DesperdicioB1 = DesperdicioB1 + 200
          B_Vital500 D_Light500   wait 30 min
                    DesperdicioB1 = DesperdicioB1 + 200
          B_Light250 D_Light500   wait 120 min
                    DesperdicioB1 = DesperdicioB1 + 200
          B_Light500 D_Light500   wait 30 min
                    DesperdicioB1 = DesperdicioB1 + 200
          B_Canola250 D_Light500   wait 120 min
                     DesperdicioB1 = DesperdicioB1 + 200
          B_Canola500 D_Light500   wait 30 min
                     DesperdicioB1 = DesperdicioB1 + 200
          D_Regular250 D_Light500   wait 120 min
                      DesperdicioB1 = DesperdicioB1 + 150
          D_Regular500 D_Light500   wait 30 min
                      DesperdicioB1 = DesperdicioB1 + 150
          D_Light250 D_Light500   wait 120 min

*****************************************************************************
*                           Entities                                        *
*****************************************************************************

  Name              Speed (mpm)  Stats       Cost
  ----------------- ------------ ----------- ------------
#
#4 por cada entidad
#
  B_Vital250        50           Time Series
#
#2 por cada entidad
  B_Vital500        50           Time Series
#
#4 por cada entidad
  B_Light250        50           Time Series
#
#2 por cada entidad
  B_Light500        50           Time Series
#
#4 por cada entidad
  B_Canola250       50           Time Series
  B_Canola500       50           Time Series
  B_Vital1000       50           Time Series
  B_Vital50         50           Time Series
#
#4 por cada entidad
  D_Regular250      50           Time Series
  D_Regular500      50           Time Series
#
#4 por cada entidad
  D_Light250        50           Time Series
  D_Light500        50           Time Series
  D_Regular1000     50           Time Series
  B_Vital3000       50           Time Series
  PlanDiario        50           Time Series
  OrdenProduccionB1 50           Time Series
  OrdenProduccionB2 50           Time Series


*****************************************************************************
```

```
*                          Resources                          *
*******************************************************************************

                    Res       Ent
 Name      Units Stats    Search     Search Path       Motion       Cost
 --------- ----- -------- ---------- ------ ---------- ------------- ------------
 Machinist 1     By Unit  Least Used Oldest           Empty: 50 mpm
                                                      Full: 50 mpm


*******************************************************************************
*                          Processing                         *
*******************************************************************************

                                  Process                          Routing

 Entity           Location           Operation          Blk  Output          Destination         Rule     Move Logic
 ---------------- ------------------ ------------------   ---- ---------------- ------------------- -------  ------------
 PlanDiario       Cola_SepOrd                             1    PlanDiario       Separacion_Ordenes   FIRST 1
 PlanDiario       Separacion_Ordenes   int Cantidad  // Cantidad a producir variable usada


                                 if LimpiezaB1 = 1 then
                                 {
                                  Route 1
                                  Route 2
                                  goto L1
                                 }

                                 if LimpiezaB2 = 1 then
                                 {
                                  Route 3
                                  Route 2
                                  goto L1
                                 }

                                 //-------------------------------------------------------------------BV250-------------------------------------------------
--------------------
                                 if BV250 > 0 then      // Orden de producción para BV250
                                 {

                                  if BV250 >= 3800 then
                                  {
                                   Do
                                   {
                                    Q = 3800
                                    Producto = 1
                                    Route 1
                                    Cantidad = Cantidad + 3800

                                   }
                                   until (BV250 - Cantidad) < 3800

                                  }



                                  if (BV250 - Cantidad) < 3800 and ((BV250 - Cantidad)) > 0 then
                                  {
                                   Q = BV250 - Cantidad
                                   Producto = 1
                                   Route 2
                                   Route 1
                                  }

                                  if ((BV250 - Cantidad)) = 0 then
                                   Route 2
                                 }
```

```
//-----------------------------------------------------------------BV500-----------------------------------------------------
--------------------
if BV500 > 0 then      // Orden de producción para BV500
{

 if BV500 >= 3800 then
 {
  Do
  {
   Q = 3800
   Producto = 2
   Route 1
   Cantidad = Cantidad + 3800

  }
  until (BV500 - Cantidad) < 3800

 }



 if (BV500 - Cantidad) < 3800 and ((BV500 - Cantidad)) > 0 then
 {
  Q = BV500 - Cantidad
  Producto = 2
  Route 2
  Route 1
 }

 if ((BV500 - Cantidad)) = 0 then
  Route 2
}



//-----------------------------------------------------------------BL250-----------------------------------------------------
--------------------
if BL250 > 0 then      // Orden de producción para BL250
{

 if BL250 >= 3800 then
 {
  Do
  {
   Q = 3800
   Producto = 6
   Route 1
   Cantidad = Cantidad + 3800

  }
  until (BL250 - Cantidad) < 3800

 }



 if (BL250 - Cantidad) < 3800 and ((BL250 - Cantidad)) > 0 then
 {
  Q = BL250 - Cantidad
  Producto = 6
  Route 2
  Route 1
 }

 if ((BL250 - Cantidad)) = 0 then
```

```
  Route 2

}

//-----------------------------------------------------------------BL500-----------------------------------------------------
-------------------
if BL500 > 0 then      // Orden de producción para BL500
{

 if BL500 >= 3800 then
 {
  Do
  {
   Q = 3800
   Producto = 7
   Route 1
   Cantidad = Cantidad + 3800

  }
  until (BL500 - Cantidad) < 3800

 }



 if (BL500 - Cantidad) < 3800 and ((BL500 - Cantidad)) > 0 then
 {
  Q = BL500 - Cantidad
  Producto = 7
  Route 2
  Route 1
 }

 if ((BL500 - Cantidad)) = 0 then
  Route 2

}


//-----------------------------------------------------------------BC250-----------------------------------------------------
-------------------
if BC250 > 0 then      // Orden de producción para BC250
{

 if BC250 >= 3800 then
 {
  Do
  {
   Q = 3800
   Producto = 8
   Route 1
   Cantidad = Cantidad + 3800

  }
  until (BC250 - Cantidad) < 3800

 }



 if (BC250 - Cantidad) < 3800 and ((BC250 - Cantidad)) > 0 then
 {
  Q = BC250 - Cantidad
  Producto = 8

  Route 2
  Route 1
 }
```

```
 if ((BC250 - Cantidad)) = 0 then
  Route 2

}

//-------------------------------------------------------------------BC500-------------------------------------------------
--------------------
if BC500 > 0 then      // Orden de producción para BC500
{

 if BC500 >= 3800 then
 {
  Do
  {
   Q = 3800
   Producto = 9
   Route 1
   Cantidad = Cantidad + 3800

  }
  until (BC500 - Cantidad) < 3800

 }



 if (BC500 - Cantidad) < 3800 and ((BC500 - Cantidad)) > 0 then
 {
  Q = BC500 - Cantidad
  Producto = 9

  Route 2
  Route 1
 }

 if ((BC500 - Cantidad)) = 0 then
  Route 2
}


//-------------------------------------------------------------------DR250-------------------------------------------------
--------------------
if DR250 > 0 then      // Orden de producción para DR250
{

 if DR250 >= 3800 then
 {
  Do
  {
   Q = 3800
   Producto = 10
   Route 1
   Cantidad = Cantidad + 3800

  }
  until (DR250 - Cantidad) < 3800

 }



 if (DR250 - Cantidad) < 3800 and ((DR250 - Cantidad)) > 0 then
 {
  Q = DR250 - Cantidad
  Producto = 10

  Route 2
```

```
  Route 1
 }

 if ((DR250 - Cantidad)) = 0 then
  Route 2

}


//-----------------------------------------------------------------DR500-----------------------------------------------
--------------------
if DR500 > 0 then      // Orden de producción para DR500
{
 //debug

 if DR500 >= 3800 then
 {
  Do
  {
   Q = 3800
   Producto = 11
   Route 1
   Cantidad = Cantidad + 3800

  }
  until (DR500 - Cantidad) < 3800

 }



 if (DR500 - Cantidad) < 3800 and ((DR500 - Cantidad)) > 0  then
 {
  Q = DR500 - Cantidad
  Producto = 11

  Route 2
  Route 1
 }

 if ((DR500 - Cantidad)) = 0 then
  Route 2

}


//-----------------------------------------------------------------DL250-----------------------------------------------
--------------------
if DL250 > 0 then      // Orden de producción para DL250
{

 if DL250 >= 3800 then
 {
  Do
  {
   Q = 3800
   Producto = 13
   Route 1
   Cantidad = Cantidad + 3800

  }
  until (DL250 - Cantidad) < 3800

 }
```

```
 if (DL250 - Cantidad) < 3800 and ((DL250 - Cantidad)) > 0 then
 {
  Q = DL250 - Cantidad
  Producto = 13

  Route 2
  Route 1
 }

 if ((DL250 - Cantidad)) = 0 then
  Route 2
}


//-------------------------------------------------------------------DL500-------------------------------------------------
--------------------
if DL500 > 0 then      // Orden de producción para DL500
{

 if DL500 >= 3800 then
 {
  Do
  {
   Q = 3800
   Producto = 14
   Route 1
   Cantidad = Cantidad + 3800

  }
  until (DL500 - Cantidad) < 3800

 }



 if (DL500 - Cantidad) < 3800 and ((DL500 - Cantidad)) > 0then
 {
  Q = DL500 - Cantidad
  Producto = 14

  Route 2
  Route 1
 }

 if ((DL500 - Cantidad)) = 0 then
  Route 2
}


//-------------------------------------------------------------------BV1000-------------------------------------------------
--------------------
if BV1000 > 0 then      // Orden de producción para BV1000
{

 if BV1000 >= 2800 then
 {
  Do
  {
   Q = 2800
   Producto =3
   Route 3
   Cantidad = Cantidad + 2800

  }
  until (BV1000 - Cantidad) < 2800

 }
```

```
 if (BV1000 - Cantidad) < 2800 and ((BV1000 - Cantidad)) > 0 then
 {
  Q = BV1000 - Cantidad
  Producto = 3
  Route 2
  Route 3
 }

 if ((BV1000 - Cantidad)) = 0 then
  Route 2
}


//-------------------------------------------------------------------BV3000-------------------------------------------------
--------------------
if BV3000 > 0 then      // Orden de producción para BV3000
{

 if BV3000 >= 2800 then
 {
  Do
  {
   Q = 2800
   Producto = 4
   Route 3
   Cantidad = Cantidad + 2800

  }
  until (BV3000 - Cantidad) < 2800

 }



 if (BV3000 - Cantidad) < 2800 and ((BV3000 - Cantidad)) > 0 then
 {
  Q = BV3000 - Cantidad
  Producto = 4
  Route 2
  Route 3
 }

 if ((BV3000 - Cantidad)) = 0 then
  Route 2
}


//-------------------------------------------------------------------BV50-------------------------------------------------
------------------
if BV50 > 0 then      // Orden de producción para BV50
{

 if BV50 >= 2800 then
 {
  Do
  {
   Q = 2800
   Producto = 5
   Route 3
   Cantidad = Cantidad + 2800

  }
  until (BV50 - Cantidad) < 2800

 }
```

```
if (BV50 - Cantidad) < 2800 and ((BV50 - Cantidad)) > 0 then
{
 Q = BV50 - Cantidad
 Producto = 5
 Route 2
 Route 3
}

if ((BV50 - Cantidad)) = 0 then
 Route 2
}

//----------------------------------------------------------------------DR1000----------------------------------------------
----------------------
if DR1000 > 0 then      // Orden de producción para DR1000
{

 if DR1000 >= 2800 then
 {
  Do
  {
   Q = 2800
   Producto = 12
   Route 3
   Cantidad = Cantidad + 2800

  }
  until (DR1000 - Cantidad) < 2800

 }



 if (DR1000 - Cantidad) < 2800 and ((DR1000 - Cantidad)) > 0 then
 {
  Q = DR1000 - Cantidad
  Producto = 12
  Route 2
  Route 3
 }

 if ((DR1000 - Cantidad)) = 0 then
  Route 2
}




L1:                1*   OrdenProduccionB1 Cola_PlanificacionB1 FIRST 1
                   2    PlanDiario        EXIT                 FIRST 1
                   3*   OrdenProduccionB2 Cola_PalnificacionB2 FIRST 1
OrdenProduccionB1 Cola_PlanificacionB1                1    OrdenProduccionB1 PlanificacionB1      FIRST 1
OrdenProduccionB1 PlanificacionB1

if limpiezaB1 = 1 then
{
 Tank_Prep( Premix1, 8*60)      // Limpieza programada antes de del primer bacth de producción
  route 3
 goto L1
}
```

```
                    if Tank_FreeCap(Premix1) = 3800 and Contents(ProduccionB1Tanques)= 0  then    // Lo aguanta
hasta que el premix este vacio
                     Route 1
                    Else
                     Route 2


                    L1:                1    OrdenProduccionB1 ProduccionB1Tanques  FIRST 1
                                     2    OrdenProduccionB1 ProduccionB1Tanques  SEND 1
                                     3    OrdenProduccionB2 EXIT                 FIRST 1
 OrdenProduccionB1 ProduccionB1Tanques  int Cantidad  // Cantidad a producir variable usada



                    //*display Q
                    //Display Producto
                    //display BV250
                    //-------------------------------------------------------------------BV250-------------------------------------------------
--------------------
                    if Q > 0 and Producto = 1 then      // Orden de producción para BV250
                    {

                     if Q = 3800 and Producto = 1 then
                     {
                      if ValorActB1 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
                       ValorActB1 = 1

                       Emulsion3800B1_1          // Operacion de mezla macro

                     }

                     if Q < 3800 and Producto = 1 then
                     {
                       if ValorActB1 = 0  then        // Cambia de producto en el pulmon solo si no hay nada en operacion
                        ValorActB1 = 1

                        PreparacionTanqueB1_1(Q)


                     }


                     if ValorActB1 = 1 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
                      Route 1            // Manda la orden para que vacie el tanque premix
                     Else
                      Route 2            // Sino aguanta la orden



                    }




                    //-------------------------------------------------------------------BV500-------------------------------------------------
--------------------
                    if Q > 0 and Producto = 2 then      // Orden de producción para BV500
                    {

                     if Q = 3800 and Producto = 2 then
                     {
                      if ValorActB1 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
                       ValorActB1 = 1

                       Emulsion3800B1_1          // Operacion de mezla macro
```

```
}

if Q < 3800 and Producto = 2 then
{
  if ValorActB1 = 0  then        // Cambia de producto en el pulmon solo si no hay nada en operacion
    ValorActB1 = 1

   PreparacionTanqueB1_1(Q)


}


if ValorActB1 = 1 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
 Route 1            // Manda la orden para que vacie el tanque premix
Else
 Route 2            // Sino aguanta la orden

}


//-----------------------------------------------------------------------BV1000-----------------------------------------------------
---------------------
if Q > 0 and Producto = 3 then      // Orden de producción para BV1000
{

if Q = 3800 and Producto = 3 then
{
 if ValorActB1 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
  ValorActB1 = 1

  Emulsion3800B1_1          // Operacion de mezla macro

}

if Q < 3800 and Producto = 3 then
{
  if ValorActB1 = 0 then        // Cambia de producto en el pulmon solo si no hay nada en operacion
    ValorActB1 = 1

   PreparacionTanqueB1_1(Q)


}


if ValorActB1 = 1 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
 Route 1            // Manda la orden para que vacie el tanque premix
Else
 Route 2            // Sino aguanta la orden

}

//-----------------------------------------------------------------------BV3000-----------------------------------------------------
---------------------
if Q > 0 and Producto = 4 then      // Orden de producción para BV3000
{

if Q = 3800 and Producto = 4 then
{
 if ValorActB1 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
  ValorActB1 = 1

  Emulsion3800B1_1          // Operacion de mezla macro

}

if Q < 3800 and Producto = 4 then
```

```
{
  if ValorActB1 = 0 then        // Cambia de producto en el pulmon solo si no hay nada en operacion
    ValorActB1 = 1

  PreparacionTanqueB1_1(Q)


}


if ValorActB1 = 1 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
 Route 1            // Manda la orden para que vacie el tanque premix
Else
 Route 2            // Sino aguanta la orden

}


//-------------------------------------------------------------------BV50-------------------------------------------------------
------------------
if Q > 0 and Producto = 5 then      // Orden de producción para BV50
{

 if Q = 3800 and Producto = 5 then
 {
  if ValorActB1 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
   ValorActB1 = 1

   Emulsion3800B1_1          // Operacion de mezla macro

 }

 if Q < 3800 and Producto = 5 then
 {
  if ValorActB1 = 0 then        // Cambia de producto en el pulmon solo si no hay nada en operacion
    ValorActB1 = 1

   PreparacionTanqueB1_1(Q)


 }


 if ValorActB1 = 1 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
 Route 1            // Manda la orden para que vacie el tanque premix
Else
 Route 2            // Sino aguanta la orden

}


//-------------------------------------------------------------------BL250-------------------------------------------------------
-------------------
if Q > 0 and Producto = 6 then      // Orden de producción para BL250
{

 if Q = 3800 and Producto = 6 then
 {
  if ValorActB1 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
   ValorActB1 = 2

   Emulsion3800B1_2          // Operacion de mezla macro

 }

 if Q < 3800 and Producto = 6 then
 {
  if ValorActB1 = 0 then        // Cambia de producto en el pulmon solo si no hay nada en operacion
```

```
    ValorActB1 = 2

   PreparacionTanqueB1_2(Q)


  }


  if ValorActB1 = 2 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
   Route 1            // Manda la orden para que vacie el tanque premix
  Else
   Route 2            // Sino aguanta la orden

 }

 //-------------------------------------------------------------------BL500-------------------------------------------------
-------------------
 if Q > 0 and Producto = 7 then      // Orden de producción para BL500
 {

  if Q = 3800 and Producto = 7 then
  {
   if ValorActB1 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
    ValorActB1 = 2

    Emulsion3800B1_2         // Operacion de mezla macro

  }

  if Q < 3800 and Producto = 7 then
  {
    if ValorActB1 = 0 then        // Cambia de producto en el pulmon solo si no hay nada en operacion
     ValorActB1 = 2

    PreparacionTanqueB1_2(Q)


  }


  if ValorActB1 = 2 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
   Route 1            // Manda la orden para que vacie el tanque premix
  Else
   Route 2            // Sino aguanta la orden

 }

 //-------------------------------------------------------------------Bc250-------------------------------------------------
-------------------
 if Q > 0 and Producto = 8 then      // Orden de producción para Bc250
 {

  if Q = 3800 and Producto = 8 then
  {
   if ValorActB1 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
    ValorActB1 = 3

    Emulsion3800B1_3         // Operacion de mezla macro

  }

  if Q < 3800 and Producto = 8 then
  {
    if ValorActB1 = 0 then        // Cambia de producto en el pulmon solo si no hay nada en operacion
     ValorActB1 = 3

    PreparacionTanqueB1_3(Q)
```

```
}

if ValorActB1 = 3 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
 Route 1            // Manda la orden para que vacie el tanque premix
Else
 Route 2            // Sino aguanta la orden

}

//-------------------------------------------------------------------Bc500-------------------------------------------------
-------------------
if Q > 0 and Producto = 9 then      // Orden de producción para Bc500
{

 if Q = 3800 and Producto = 9 then
 {
  if ValorActB1 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
   ValorActB1 = 3

   Emulsion3800B1_3          // Operacion de mezla macro

 }

 if Q < 3800 and Producto = 9 then
 {
   if ValorActB1 = 0 then        // Cambia de producto en el pulmon solo si no hay nada en operacion
    ValorActB1 = 3

   PreparacionTanqueB1_3(Q)


 }


 if ValorActB1 = 3 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
 Route 1            // Manda la orden para que vacie el tanque premix
Else
 Route 2            // Sino aguanta la orden

}

//-------------------------------------------------------------------DR250-------------------------------------------------
-------------------
if Q > 0 and Producto = 10 then      // Orden de producción para DR250
{

 if Q = 3800 and Producto = 10 then
 {
  if ValorActB1 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
   ValorActB1 = 4

   Emulsion3800B1_4          // Operacion de mezla macro

 }

 if Q < 3800 and Producto = 10 then
 {
   if ValorActB1 = 0 then        // Cambia de producto en el pulmon solo si no hay nada en operacion
    ValorActB1 = 4

   PreparacionTanqueB1_4(Q)


 }
```

```
 if ValorActB1 = 4 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
  Route 1            // Manda la orden para que vacie el tanque premix
 Else
  Route 2            // Sino aguanta la orden

}

//-----------------------------------------------------------------DR500-----------------------------------------------
-------------------
if Q > 0 and Producto = 11 then      // Orden de producción para DR250
{


 if Q = 3800 and Producto = 11 then
 {
  if ValorActB1 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
   ValorActB1 = 4

   Emulsion3800B1_4          // Operacion de mezla macro

 }

 if Q < 3800 and Producto = 11 then
 {
   if ValorActB1 = 0 then        // Cambia de producto en el pulmon solo si no hay nada en operacion
    ValorActB1 = 4

   PreparacionTanqueB1_4(Q)


 }


 if ValorActB1 = 4 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
  Route 1            // Manda la orden para que vacie el tanque premix
 Else
  Route 2            // Sino aguanta la orden

}

//-----------------------------------------------------------------DR1000-----------------------------------------------
----------------------
if Q > 0 and Producto = 12 then      // Orden de producción para DR1000
{

 if Q = 3800 and Producto = 12 then
 {
  if ValorActB1 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
   ValorActB1 = 4

   Emulsion3800B1_4          // Operacion de mezla macro

 }

 if Q < 3800 and Producto = 12 then
 {
   if ValorActB1 = 0 then        // Cambia de producto en el pulmon solo si no hay nada en operacion
    ValorActB1 = 4

   PreparacionTanqueB1_4(Q)


 }


 if ValorActB1 = 4 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
  Route 1            // Manda la orden para que vacie el tanque premix
```

```
 Else
  Route 2          // Sino aguanta la orden

}

//-----------------------------------------------------------------DL250-------------------------------------------------
--------------------
if Q > 0 and Producto = 13 then      // Orden de producción para DL250
{

 if Q = 3800 and Producto = 13 then
 {
  if ValorActB1 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
   ValorActB1 = 5

   Emulsion3800B1_5          // Operacion de mezla macro

 }

 if Q < 3800 and Producto = 13 then
 {
  if ValorActB1 = 0 then        // Cambia de producto en el pulmon solo si no hay nada en operacion
    ValorActB1 = 5

   PreparacionTanqueB1_5(Q)


 }


 if ValorActB1 = 5 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
  Route 1          // Manda la orden para que vacie el tanque premix
 Else
  Route 2          // Sino aguanta la orden

}


//-----------------------------------------------------------------DL500-------------------------------------------------
--------------------
if Q > 0 and Producto = 14 then      // Orden de producción para DL500
{

 if Q = 3800 and Producto = 14 then
 {
  if ValorActB1 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
   ValorActB1 = 5

   Emulsion3800B1_5          // Operacion de mezla macro

 }

 if Q < 3800 and Producto = 14 then
 {
  if ValorActB1 = 0 then        // Cambia de producto en el pulmon solo si no hay nada en operacion
    ValorActB1 = 5

   PreparacionTanqueB1_5(Q)


 }


 if ValorActB1 = 5 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
  Route 1          // Manda la orden para que vacie el tanque premix
 Else
  Route 2          // Sino aguanta la orden
```

```
                        }

                        L1:              1   OrdenProduccionB1 VaciadoTanqueB1     FIRST 1
                                         2   OrdenProduccionB1 VaciadoTanqueB1     SEND 1
                                         3   OrdenProduccionB1 VaciadoTanqueB1     SEND 1
OrdenProduccionB1 VaciadoTanqueB1
                        If producto = 1 then
                         ValorActB1 = 1
                        If producto = 2 then
                         ValorActB1 = 1
                        If producto = 6 then
                         ValorActB1 = 2
                        If producto = 7 then
                         ValorActB1 = 2
                        If producto = 8 then
                         ValorActB1 = 3
                        If producto = 9 then
                         ValorActB1 = 3
                        If producto = 10 then
                         ValorActB1 = 4
                        If producto = 11 then
                         ValorActB1 = 4
                        If producto = 13 then
                         ValorActB1 = 5
                        If producto = 14 then
                         ValorActB1 = 5


                        do
                         Tank_Transfer(Premix1, Pulmon1,1, 227, 0, 3799)  // Se encarga de vaciar el premix
                        until Tank_FreeCap(Premix1) = 3800

                        if contents(PlanificacionB1,OrdenProduccionB1) = 1 then  //Revisa si hay ordenes esperando en
PlanificacionB1
                         {
                          SEND 1 OrdenProduccionB1 TO ProduccionB1Tanques
                         }









                                         1   OrdenProduccionB1 ProduccionB1        FIRST 1
OrdenProduccionB1 ProduccionB1        int contador  //  Contador que registra los kilos q se manda a producir




                        if Q > 0 then
                        {
                         do
                         {

                         contador = contador + 1

                         wait 1.5 sec          // Tiempo que demora el producto desde el tanque pulmon al MPU  se definio
por las distancias y hp de la bomba(?)

                         if Producto = 1 then
                          ORDER 1 B_Vital250 TO Cola_Kombinator

                         if Producto = 2 then
```

```
                    ORDER 1 B_Vital500 TO Cola_Kombinator

                   if Producto = 6 then
                    ORDER 1 B_Light250 TO Cola_Kombinator

                   if Producto = 7 then
                    ORDER 1 B_Light500 TO Cola_Kombinator

                   if Producto = 8 then
                    ORDER 1 B_Canola250 TO Cola_Kombinator

                   if Producto = 9 then
                    ORDER 1 B_Canola500 TO Cola_Kombinator

                   if Producto = 10 then
                    ORDER 1 D_Regular250 TO Cola_Kombinator

                   if Producto = 11 then
                    ORDER 1 D_Regular500 TO Cola_Kombinator

                   if Producto = 13 then
                    ORDER 1 D_Light250 TO Cola_Kombinator

                   if Producto = 14 then
                    ORDER 1 D_Light500 TO Cola_Kombinator

                   Tank_Dec(Pulmon1, 1)

                   }
                   until Q - contador < 1



                  }
                                1    OrdenProduccionB1 EXIT              FIRST 1
ALL          Cola_Kombinator                1    ALL            Kombinator         FIRST 1
ALL          Kombinator         wait 1.5 sec    // Espera el tiempo de proceso definido por la tasas
                                1    ALL            RecirculacionB1     FIRST 1
ALL          RecirculacionB1                1    ALL            B1                FIRST 1
ALL          B1
                  wait tasa sec


                  if FreeCap(RecirculacionB1) = Cap(RecirculacionB1) and contents(Kombinator)= 0 then
                  {

                   ValorAntB1 = ValorActB1
                   ValorActB1=0  // Se resetea la var de control


                   if contents(ProduccionB1Tanques,OrdenProduccionB1) = 1  then
                   {
                    SEND 1 OrdenProduccionB1 TO VaciadoTanqueB1
                   }
                  }


                  /*

                  SalidaPlanCantidad[indice] = SalidaPlanCantidad[indice] + 1

                  If SalidaPlanCantidad[Indice] = EntradaDemanda[Indice] then
                  {
                   SalidaPlanDia[Indice] = caldom()   //Solo sirve en pm7
                   SalidaPlanMes[Indice] = calmonth()  //Solo sirve en pm7
                   SalidaPlanHora[Indice] = CalHour()
                  }


                  */                1    ALL            BPTB1             FIRST 1
```

```
ALL            BPTB1

                                        1    ALL              EXIT              FIRST 1
OrdenProduccionB2 Cola_PalnificacionB2                  1    OrdenProduccionB2 PlanificacionB2      FIRST 1
OrdenProduccionB2 PlanificacionB2
                              // Revisa si existe una orden de limpieza

                              if limpiezaB2 = 1 then
                              {
                               Tank_Prep( Premix2, 8*60)     // Limpieza programada antes de del primer bacth de producción
                               //Tank_Prep( Pulmon2, 8*60)     // Limpieza programada antes de del primer bacth de producción
                               route 3
                               goto L1
                              }



                              if Tank_FreeCap(Premix2) = 2800 and Contents(ProduccionB2Tanques)= 0  then    // Lo aguanta
hasta que el premix este vacio
                               Route 1
                              Else
                               Route 2



                              L1:                1    OrdenProduccionB2 ProduccionB2Tanques  FIRST 1
                                                 2    OrdenProduccionB2 ProduccionB2Tanques  SEND 1
                                                 3    OrdenProduccionB2 EXIT                 FIRST 1
OrdenProduccionB2 ProduccionB2Tanques  int Cantidad  // Cantidad a producir variable usada


                              //-------------------------------------------------------------------BV1000-------------------------------------------------
---------------------
                              if Q > 0 and Producto = 3 then      // Orden de producción para BV1000
                              {



                               if Q = 2800 and Producto = 3 then
                               {
                                if ValorActB2 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
                                 ValorActB2 = 1

                                 Emulsion2800B2_1          // Operacion de mezla macro

                               }

                               if Q < 2800 and Producto = 3 then
                               {
                                 if ValorActB2 = 0 then        // Cambia de producto en el pulmon solo si no hay nada en operacion
                                  ValorActB2 = 1

                                 PreparacionTanqueB2_1(Q)


                               }


                               if ValorActB2 = 1 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
                                Route 1           // Manda la orden para que vacie el tanque premix
                               Else
                                Route 2           // Sino aguanta la orden

                              }

                              //-------------------------------------------------------------------BV3000-------------------------------------------------
---------------------
                              if Q > 0 and Producto = 4 then      // Orden de producción para BV3000
                              {
```

```
if Q = 2800 and Producto = 4 then
{
 if ValorActB2 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
  {
  ValorActB2 = 1
  Display "se da"
  }

  Emulsion2800B2_1          // Operacion de mezla macro

}

if Q < 2800 and Producto = 4 then
{
  if ValorActB2 = 0 then        // Cambia de producto en el pulmon solo si no hay nada en operacion
    ValorActB2 = 1

  PreparacionTanqueB2_1(Q)


}


if ValorActB2 = 1 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
 {

 Route 1            // Manda la orden para que vacie el tanque premix
 }
Else
 Route 2            // Sino aguanta la orden

}


//-----------------------------------------------------------------BV50-----------------------------------------------------
------------------
if Q > 0 and Producto = 5 then      // Orden de producción para BV50
{

if Q = 2800 and Producto = 5 then
{
 if ValorActB2 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
  ValorActB2 = 1

  Emulsion2800B2_1          // Operacion de mezla macro

}

if Q < 2800 and Producto = 5 then
{
  if ValorActB2 = 0 then        // Cambia de producto en el pulmon solo si no hay nada en operacion
    ValorActB2 = 1

  PreparacionTanqueB2_1(Q)


}


if ValorActB2 = 1 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
 Route 1            // Manda la orden para que vacie el tanque premix
Else
 Route 2            // Sino aguanta la orden

}
```

```
//-----------------------------------------------------------------DR1000-----------------------------------------------
---------------------
if Q > 0 and Producto = 12 then      // Orden de producción para DR1000
{

 if Q = 2800 and Producto = 12 then
 {
  if ValorActB2 = 0 then   // Cambia de producto en el pulmon solo si no hay nada en operacion
   ValorActB2 = 4

   Emulsion2800B2_4          // Operacion de mezla macro

 }

 if Q < 2800 and Producto = 12 then
 {
   if ValorActB2 = 0 then        // Cambia de producto en el pulmon solo si no hay nada en operacion
    ValorActB2 = 4

   PreparacionTanqueB2_4(Q)


 }


 if ValorActB2 = 4 then        // Si el producto que esta en el premix es igual al que esta en el pulom
o el pulmon esta vacio
  Route 1            // Manda la orden para que vacie el tanque premix
 Else
  Route 2            // Sino aguanta la orden

}

L1:                 1    OrdenProduccionB2 VaciadoTanqueB2      FIRST 1
                    2    OrdenProduccionB2 VaciadoTanqueB2      SEND 1
 OrdenProduccionB2 VaciadoTanqueB2
 if Producto = 3 then
  ValorActB2 = 1

 if Producto = 4 then
  ValorActB2 = 1


 if Producto = 5 then
  ValorActB2 = 1

 if Producto = 12 then
  ValorActB2 = 4
do
 Tank_Transfer(Premix2, Pulmon2,1, 227, 0, 2799)  // Se encarga de vaciar el premix
until Tank_FreeCap(Premix2) = 2800

if contents(PlanificacionB2,OrdenProduccionB2) = 1 then  //Revisa si hay ordenes esperando en
PlanificacionB2
 {
  SEND 1 OrdenProduccionB2 TO ProduccionB2Tanques
 }


/*if contents(RecirculacionB2) >= 1000 then
 {
  Route 2
 }
Else
 Route 1
*/



                    1    OrdenProduccionB2 ProduccionB2         FIRST 1
```

```
OrdenProduccionB2 ProduccionB2        int contador  //  Contador que registra los kilos q se manda a producir


                                //display Q
                                //Display Producto


                                if Q > 0 then
                                {
                                 do
                                 {

                                 contador = contador + 1

                                 wait 1.5 sec         // Tiempo que demora el producto desde el tanque pulmon al MPU  se definio
por las distancias y hp de la bomba(?)

                                 if Producto = 3 then
                                  ORDER 1 B_Vital1000 TO cola_Votator

                                 if Producto = 4 then
                                  ORDER 1 B_Vital3000 TO cola_Votator

                                 if Producto = 5 then
                                  ORDER 1 B_Vital50 TO cola_Votator

                                 if Producto = 12 then
                                  ORDER 1 D_Regular1000 TO cola_Votator

                                 Tank_Dec(Pulmon2, 1)

                                 }
                                 until Q - contador < 1



                                }                1    OrdenProduccionB2 EXIT              FIRST 1
ALL            cola_Votator                          1    ALL             Votator            FIRST 1
ALL            Votator            wait 2 sec     // Espera el tiempo de proceso definido por la tasas
                                                1    ALL             recirculacionB2      FIRST 1
ALL            recirculacionB2      /*
                                if contents(RecirculacionB2) <= 500 and Contents(VaciadoTanqueB2) = 1 then
                                {
                                 SEND 1 OrdenProduccionB2 TO ProduccionB2
                                }
                                */



                                                1    ALL             ventanaB2           FIRST 1
ALL            ventanaB2           if contents(ventanaB2, D_Regular1000) = 1 then
                                 Route 1
                                if contents(ventanaB2, B_Vital3000) = 1 then
                                 Route 2
                                if contents(ventanaB2, B_Vital1000) = 1 then
                                 Route 3
                                if contents(ventanaB2, B_Vital50) = 1 then
                                 Route 4            1    D_Regular1000     B2                FIRST 1
                                                2    B_Vital3000       Em                FIRST 1
                                                3    B_Vital1000       B2                FIRST 1
                                                4    B_Vital50         Em                FIRST 1
ALL            B2
                                wait tasa sec


                                if FreeCap(RecirculacionB2) = Cap(RecirculacionB2) and contents(Votator)= 0 and
Tank_FreeCap(Pulmon2) = Tank_Cap(Pulmon2) and contents(Em) = 0 then

                                {
```

```
                    ValorAntB2 = ValorActB2
                    ValorActB2=0  // Se resetea la var de control


                    if contents(ProduccionB2Tanques,OrdenProduccionB2) = 1  then
                    {
                     SEND 1 OrdenProduccionB2 TO VaciadoTanqueB2
                    }
                   }

                   /*
                   SalidaPlanCantidad[indice] = SalidaPlanCantidad[indice] + 1

                   If SalidaPlanCantidad[Indice] = EntradaDemanda[Indice] then
                   {
                    SalidaPlanDia[Indice] = caldom()   //Solo sirve en pm7
                    SalidaPlanMes[Indice] = calmonth()  //Solo sirve en pm7
                    SalidaPlanHora[Indice] = CalHour()
                   }



                   */              1    ALL            BPTB2             FIRST 1
 ALL          BPTB2                        1    ALL            EXIT             FIRST 1
 ALL          Em
                   wait tasa sec


                   if FreeCap(RecirculacionB2) = Cap(RecirculacionB2) and contents(Votator)= 0 and contents(B2)=0
and  Tank_FreeCap(Pulmon2) = Tank_Cap(Pulmon2) then
                   {
                    ValorAntB2 = ValorActB2
                    ValorActB2=0  // Se resetea la var de control


                    if contents(ProduccionB2Tanques,OrdenProduccionB2) = 1  then
                    {
                     SEND 1 OrdenProduccionB2 TO VaciadoTanqueB2
                    }
                   }

                   /*
                   SalidaPlanCantidad[indice] = SalidaPlanCantidad[indice] + 1

                   If SalidaPlanCantidad[Indice] = EntradaDemanda[Indice] then
                   {
                    SalidaPlanDia[Indice] = caldom()   //Solo sirve en pm7
                    SalidaPlanMes[Indice] = calmonth()  //Solo sirve en pm7
                    SalidaPlanHora[Indice] = CalHour()
                   }
                   */



                                   1    ALL            BPTEm             FIRST 1
 ALL          BPTEm                        1    ALL            EXIT             FIRST 1


*******************************************************************************
*                          Arrivals                                           *
*******************************************************************************

  Entity     Location    Qty each   First Time Occurrences Frequency  Logic
  ---------- ----------- ---------- ---------- ----------- ---------- ------------
  PlanDiario Cola_SepOrd 1                  1          0         BV250 = 3800
                                                  Tasa = 3.38
  PlanDiario Cola_SepOrd 1                  1          0         BV500= 3800
                                                  Tasa = 2.82
```

```
  PlanDiario Cola_SepOrd 1                1          0          DR500 = 5000
                                                  Tasa =  2.82

  PlanDiario Cola_SepOrd 1                1          0          DL500 = 3800
                                                  Tasa = 2.82
  PlanDiario Cola_SepOrd 1                1          0          BL250 = 2000
                                                  Tasa = 3.38
  PlanDiario Cola_SepOrd 1                1          0          BL500 = 4000
                                                  Tasa = 2.82
  PlanDiario Cola_SepOrd 1                1          0          DL250 = 7000
                                                  Tasa = 2.82
  PlanDiario Cola_SepOrd 1                1          0          BV1000= 2800
                                                  Tasa = 3.84
  PlanDiario Cola_SepOrd 1                1          0          dr1000 = 2800
                                                  Tasa = 3.84
  PlanDiario Cola_SepOrd 1                1          0          BV3000= 2800
                                                  tasa= 4.02
  PlanDiario Cola_SepOrd 1                1          0          bv50 = 2800
                                                  Tasa = 7.55


*******************************************************************************
*                          Attributes                                         *
*******************************************************************************

  ID         Type         Classification
  ---------- ------------ --------------
  BV250      Integer      Entity
  BV500      Integer      Entity
  BV1000     Integer      Entity
  BV3000     Integer      Entity
  BV50       Integer      Entity
  BL250      Integer      Entity
  BL500      Integer      Entity
  BC250      Integer      Entity
  BC500      Integer      Entity
  DR250      Integer      Entity
  DR500      Integer      Entity
  DR1000     Integer      Entity
  DL250      Integer      Entity
  DL500      Integer      Entity
  LimpiezaB1 Integer      Entity
  LimpiezaB2 Integer      Entity
#
#Registra el producto en la locación
#
  PrdActB1   Integer      Location
#
#Registra el producto en la locación
  PrdActB2   Integer      Location
#
#Valor menor a 3.8 ton q falta para completar el batch
  FaltanteB1 Integer      Entity
#
#Valor menor a 2.8 ton q falta para completar el batch
  FaltanteB2 Integer      Entity
#
#Tiempo de ciclo definido en el archivo de arribos
  Tasa       Real         Entity
#
#Cantidad de cada orden de producción
  Q          Integer      Entity
#
#1.  BV250
#2.  BV500
#3.  BV1000
#4.  BV3000
#5.  BV50
#6.  BL250
#7.  BL500
```

```
#8.  BC250
#9.  BC500
#10. DR250
#11. DR500
#12. DR1000
#13. DL250
#14. DL500
#
  Producto   Integer      Entity
#
#Controla los planes definido en el archivo de entrada
  Indice     Integer      Entity
  Att1       Integer      Entity


*******************************************************************************
*                          Variables (global)                                 *
*******************************************************************************

  ID            Type         Initial value Stats
  ------------- ------------ ------------- -----------
#
#Records the quantity remaining after a fill if there was insufficient capacity.
  Tank_QtyLeft  Real         0             None
#
#Define la cantidad que se debe producir en esa linea hasta pasar a otro producto
#
  ValorActB1    Integer      0             None
  ValorAntB1    Integer      0             None
#
#Define la cantidad que se debe producir en esa linea hasta pasar a otro producto
#
  ValorActB2    Integer      0             Basic
  ValorAntB2    Integer      0             None
#
#Se produce al haber cambio de emulsión (ValorActB1)
  DesperdicioB1 Integer      0             Time Series


*******************************************************************************
*                                Arrays                                       *
*******************************************************************************

  ID                 Dimensions   Type
  ------------------ ------------ ------------
  Tank_Level         100          Real
  Tank_Statistics    100,14       Real
  Tank_State         100          Integer
  Tank_Product       100          Integer
  Tank_Fills         100          Integer
  SalidaPlanDia      46           Integer
  SalidaPlanMes      46           Integer
  SalidaPlanHora     46           Integer
  SalidaPlanCantidad 46           Integer
  EntradaDemanda     46           Integer


*******************************************************************************
*                                Macros                                       *
*******************************************************************************

  ID                      Text
  ------------------------ ------------
  Tank_Idle                0 // state for an idle tank
  Tank_Operation           1
  Tank_Setup               2
  Tank_Filling             3
  Tank_Emptying            4
  Tank_Blocked             5
  Tank_Down                6
```

```
Tank_Loop               While 1 do
Tank_TimeStep             .2
Tank_Stop               -1.0
Tank_LongestIdle          1
Tank_LongestBlocked       2
Tank_InOrder              3
Emulsion3800B1_1          //BV
                    // Macro para  preparacion = 3800

                      Tank_Transfer(I1, Premix1, 1201.58, 97.7+ L(29.4,25.5), 0, 0)
                      Tank_Transfer(I2, Premix1, 345.80, 1+L(251,112), 0, 0)
                      Tank_Transfer(I3, Premix1, 292.6, 1+L(251,112), 0, 0)
                      Tank_Transfer(I4, Premix1, 425.6,  1+L(248,144), 0, 0)
                      Tank_Transfer(I5, Premix1, 399, 1+L(248,144), 0, 0)
                      Tank_Transfer(I6, Premix1, 598.61, 309 + L(614,951), 0, 0)
                      wait 1 min
                      Tank_Inc(Premix1,0.23)
                      Tank_Dec(I7,0.23)
                      //Tank_Transfer(I7, Premix1, 0.23, N(0.305,0.152), 0, 0)
                      Tank_Transfer(I8, Premix1, 475, 245+L(131,536), 0, 0)
                      Tank_Transfer(I9, Premix1, 42.22,20.6+L(78.8,354) , 0, 0)
                      Tank_Transfer(I10, Premix1, 19.36,1+L(1.63,0.644), 0, 0)
                      Tank_DoOperation(Premix1, 10)          // Operacion de mezcla
Emulsion3800B1_2          // BL
                    // Macro para  preparacion = 3800

                      Tank_Transfer(I1, Premix1, 952.869, 97.7+ L(29.4,25.5), 0, 0)
                      Tank_Transfer(I2, Premix1, 271.70, 1+L(251,112), 0, 0)
                      Tank_Transfer(I3, Premix1, 229.9,1+L(251,112), 0, 0)
                      Tank_Transfer(I4, Premix1, 334.4, 1+L(248,144), 0, 0)
                      Tank_Transfer(I5, Premix1, 313.5, 1+L(248,144), 0, 0)
                      Tank_Transfer(I6, Premix1, 1211.869, 309 + L(614,951), 0, 0)
                      wait 1 min
                      Tank_Inc(Premix1,0.228)
                      Tank_Dec(I7,0.228)
                      //Tank_Transfer(I7, Premix1, 0.228, N(0.305,0.152), 0, 0)
                      Tank_Transfer(I8, Premix1, 323, 245+L(131,536), 0, 0)
                      Tank_Transfer(I9, Premix1, 137.222, 20.6+L(78.8,354) , 0, 0)
                      Tank_Transfer(I10, Premix1, 25.312, 1+L(1.63,0.644), 0, 0)
                      Tank_DoOperation(Premix1, 10)          // Operacion de mezcla
Emulsion3800B1_3          // BC
                    // Macro para  preparacion = 3800

                      Tank_Transfer(I1, Premix1, 1117 , 97.7+ L(29.4,25.5), 0, 0)
                      Tank_Transfer(I2, Premix1, 321.1, 1+L(251,112), 0, 0)
                      Tank_Transfer(I3, Premix1, 271.7, 1+L(251,112), 0, 0)
                      Tank_Transfer(I4, Premix1, 370.5, 1+L(248,144), 0, 0)
                      Tank_Transfer(I5, Premix1, 395.2, 1+L(248,144), 0, 0)
                      Tank_Transfer(I6, Premix1, 788.797, 309 + L(614,951), 0, 0)
                      wait 1 min
                      Tank_Inc(Premix1,0.228)
                      Tank_Dec(I7,0.228)
                      //Tank_Transfer(I7, Premix1, 0.228, N(0.305,0.152), 0, 0)
                      Tank_Transfer(I8, Premix1, 475, 245+L(131,536), 0, 0)
                      Tank_Transfer(I9, Premix1, 42.222, 20.6+L(78.8,354) , 0, 0)
                      Tank_Transfer(I10, Premix1, 18.253,1+L(1.63,0.644), 0, 0)
                      Tank_DoOperation(Premix1, 10)          // Operacion de mezcla
Emulsion3800B1_4          // DR
                    // Macro para  preparacion = 3800

                      Tank_Transfer(I1, Premix1, 2337, 97.7+ L(29.4,25.5), 0, 0)
                      Tank_Transfer(I2, Premix1, 0, 1+L(251,112), 0, 0)
                      Tank_Transfer(I3, Premix1, 0, 1+L(251,112), 0, 0)
                      Tank_Transfer(I4, Premix1, 0, 1+L(248,144), 0, 0)
                      Tank_Transfer(I5, Premix1, 239, 1+L(248,144), 0, 0)
                      Tank_Transfer(I6, Premix1, 720, 309 + L(614,951), 0, 0)
                      wait 1 min
                      Tank_Inc(Premix1,89)
                      Tank_Dec(I7,89)
                      //Tank_Transfer(I7, Premix1, 89, N(0.305,0.152), 0, 0)
```

```
                    Tank_Transfer(I8, Premix1, 399, 245+L(131,536), 0, 0)
                    Tank_Transfer(I9, Premix1, 0, 20.6+L(78.8,354) , 0, 0)
                    Tank_Transfer(I10, Premix1, 16, 1+L(1.63,0.644), 0, 0)
                    Tank_DoOperation(Premix1, 10)          // Operacion de mezcla
Emulsion3800B1_5        // DL
                // Macro para  preparacion = 3800


                    Tank_Transfer(I1, Premix1, 1840.3387, 97.7+ L(29.4,25.5), 0, 0)
                    Tank_Transfer(I2, Premix1, 0, 1+L(251,112), 0, 0)
                    Tank_Transfer(I3, Premix1, 0, 1+L(251,112), 0, 0)
                    Tank_Transfer(I4, Premix1, 0, 1+L(248,144), 0, 0)
                    Tank_Transfer(I5, Premix1, 188.1, 1+L(248,144), 0, 0)
                    Tank_Transfer(I6, Premix1, 1286.9477, 309 + L(614,951), 0, 0)
                    wait 1 min
                    Tank_Inc(Premix1,69.9516)
                    Tank_Dec(I7,69.9516)
                    //Tank_Transfer(I7, Premix1, 69.9516, N(0.305,0.152), 0, 0)  Siempre demora 1 min
                    Tank_Transfer(I8, Premix1, 399, 245+L(131,536), 0, 0)
                    Tank_Transfer(I9, Premix1, 0, 20.6+L(78.8,354) , 0, 0)
                    Tank_Transfer(I10, Premix1, 15.662, 1+L(1.63,0.644), 0, 0)
                    Tank_DoOperation(Premix1, 10)          // Operacion de mezcla
Emulsion2800B2_1        //BV
                // Macro para  preparacion = 2800

                    Tank_Transfer(I1, Premix2, 885.374, 97.7+ L(29.4,25.5), 0, 0)
                    Tank_Transfer(I2, Premix2, 254.8, 1+L(251,112), 0, 0)
                    Tank_Transfer(I3, Premix2, 215.6, 1+L(251,112), 0, 0)
                    Tank_Transfer(I4, Premix2, 313.6, 1+L(248,144), 0, 0)
                    Tank_Transfer(I5, Premix2, 294, 1+L(248,144), 0, 0)
                    Tank_Transfer(I6, Premix2, 441.083, 309 + L(614,951), 0, 0)
                    wait 1 min
                    Tank_Inc(Premix2,0.168)
                    Tank_Dec(I7,0.168)
                    //Tank_Transfer(I7, Premix2, 0.168, N(0.305,0.152), 0, 0)
                    Tank_Transfer(I8, Premix2, 350, 245+L(131,536), 0, 0)
                    Tank_Transfer(I9, Premix2, 31.111,20.6+L(78.8,354) , 0, 0)
                    Tank_Transfer(I10, Premix2, 14.264,1+L(1.63,0.644), 0, 0)
                    Tank_DoOperation(Premix2, 10)          // Operacion de mezcla
Emulsion2800B2_4        // DR
                // Macro para  preparacion = 2800

                    Tank_Transfer(I1, Premix2, 1721.767, 97.7+ L(29.4,25.5), 0, 0)
                    Tank_Transfer(I2, Premix2, 0, 1+L(251,112), 0, 0)
                    Tank_Transfer(I3, Premix2, 0, 1+L(251,112), 0, 0)
                    Tank_Transfer(I4, Premix2, 0, 1+L(248,144), 0, 0)
                    Tank_Transfer(I5, Premix2, 176.4, 1+L(248,144), 0, 0)
                    Tank_Transfer(I6, Premix2, 530.506, 309 + L(614,951), 0, 0)
                    wait 1 min
                    Tank_Inc(Premix2,65.543)
                    Tank_Dec(I7,65.543)
                    //Tank_Transfer(I7, Premix2, 65.543, N(0.305,0.152), 0, 0)
                    Tank_Transfer(I8, Premix2, 294, 245+L(131,536), 0, 0)
                    Tank_Transfer(I9, Premix2, 0, 20.6+L(78.8,354) , 0, 0)
                    Tank_Transfer(I10, Premix2, 11.784, 1+L(1.63,0.644), 0, 0)
                    Tank_DoOperation(Premix2, 10)          // Operacion de mezcla


*****************************************************************************
*                          Subroutines                                      *
*****************************************************************************

 ID                  Type        Parameter          Type        Logic
 ------------------- ----------- ------------------ ----------- ------------------
 Tank_Fill           None        Tank_ID            Integer     // Fills a tank_ID by a quantity and rate.
                                 Tank_FillQty       Real
                                 Tank_FillRate      Real        Real Tank_TimeInc
                                 Tank_ResumeLevel   Real        Real Tank_FillInc
                                                                Real Tank_FillCapacity = Tank_Cap(Tank_ID)
                                                                Int Tank_VarRate=0
```

```
Real Tank_FillRequest = Tank_FillQty

If Tank_FillRate <= 0.0
Then Tank_VarRate = 1
Else Tank_FillRate = Tank_FillRate * Tank_TimeStep // convert units per minute
to units per timestep

Wait Until Tank_State[Tank_ID] <> Tank_Down    // Don't fill if Tank_ID is down
While Tank_FillQty > 0.0000001 do
     {
 Tank_TimeInc = Tank_TimeStep
 If Tank_Level [Tank_ID] = Tank_FillCapacity  then // Is Tank_ID already full?
  {
  If Tank_ResumeLevel = Tank_Stop then
   {
   Tank_QtyLeft = Tank_FillQty
   Return
   }
  If Tank_ResumeLevel = 0.0
  Then Stop "Attempted to fill "$Loc(Tank_ID) $ " beyond its capacity of "
$Tank_Cap(Tank_ID)

  Else If Tank_ResumeLevel  >= Tank_FillCapacity
  Then Stop "Resume Level in Tank_Fill subroutine for "$Loc(Tank_ID)$ " must
be less than its capacity"

  Else Wait Until Tank_Level[Tank_ID] <= Tank_ResumeLevel // wait for drop to
resume filling

  }
 if Tank_VarRate then
  {
  Tank_FillRate = Tank_Rate(0, Tank_ID) * Tank_TimeStep
  if Tank_FillRate = 0.0 then
   {
   Tank_QtyLeft =Tank_FillQty
   Return
   }
  }
 // Adjust rate for last remaining quantity
 If Tank_FillQty < Tank_FillRate then
  {
  Tank_TimeInc = Tank_TimeInc * Tank_FillQty/ Tank_FillRate // proportionalize
  Tank_FillInc = Tank_FillQty
  }
 Else  Tank_FillInc = Tank_FillRate
 If Tank_FillCapacity - Tank_Level[Tank_ID]  < Tank_FillInc then // Test for
insufficient cap BEFORE Wait

  {
  Tank_TimeInc = Tank_TimeInc * (Tank_FillCapacity - Tank_Level[Tank_ID]) /
Tank_FillInc // proportionalize

  Tank_FillInc = Tank_FillCapacity - Tank_Level[Tank_ID]
  }
 if Tank_State[Tank_ID] <> Tank_Filling
 then Tank_SetState(Tank_ID,Tank_Filling)
 Wait Tank_TimeInc
 Wait Until Tank_State[Tank_ID] <> Tank_Down    // Don't fill if Tank_ID is down
 If Tank_FillCapacity - Tank_Level[Tank_ID]  < Tank_FillInc // Test for insufficient
cap AFTER Wait

 then Tank_FillInc = Tank_FillCapacity - Tank_Level[Tank_ID]
 Dec Tank_FillQty, Tank_FillInc
 Inc Tank_Level [Tank_ID], Tank_FillInc
 Tank_UpdateStats (Tank_ID)
 }
// Adjust for floating point error
if Tank_Level[Tank_ID]<Tank_FillRequest
then if Tank_Level[Tank_ID] + .0000001 > Tank_FillRequest then
 {
 Tank_Level[Tank_ID] = Tank_FillRequest
 Tank_UpdateStats (Tank_ID)
 }
Tank_QtyLeft = 0.0
Inc Tank_Fills[Tank_ID]
```

```
Tank_Empty          None        Tank_ID          Integer      // Empties a Tank_ID by a quantity and rate.
                      Tank_EmptyQty     Real
                      Tank_EmptyRate    Real        Real Tank_TimeInc
                      Tank_ResumeLevel  Real        Real Tank_EmptyInc
                                        Int Tank_VarRate

                                        If Tank_EmptyRate <= 0.0
                                        Then Tank_VarRate = 1
                                        Else Tank_EmptyRate = Tank_EmptyRate * Tank_TimeStep // convert units per
minute to units per timestep
                                        Wait Until Tank_State[Tank_ID] <> Tank_Down // Don't empty if Tank_ID is down
                                        While Tank_EmptyQty > 0.0 do
                                         {
                                         Tank_TimeInc = Tank_TimeStep
                                         If Tank_Level [Tank_ID] = 0.0 then // Is Tank_ID empty?
                                          {
                                          If Tank_ResumeLevel = Tank_Stop then
                                           {
                                           Tank_QtyLeft = Tank_EmptyQty
                                           Return
                                           }
                                          If Tank_ResumeLevel = 0.0
                                          Then Stop "Attempted to empty "$Loc(Tank_ID) $ " by more than its contents"
                                          Else If Tank_ResumeLevel  > Tank_Cap(Tank_ID)
                                          Then Stop "Resume Level in Tank_Empty subroutine for "$Loc(Tank_ID)$ "
must be less than or equal to its capacity"
                                          Else Wait Until Tank_Level[Tank_ID] >= Tank_ResumeLevel // wait for rise to
resume filling
                                          }
                                         If Tank_VarRate then
                                         {
                                         Tank_EmptyRate = Tank_Rate(Tank_ID, 0) * Tank_TimeStep
                                         if Tank_EmptyRate = 0.0 then
                                          {
                                          Tank_QtyLeft = Tank_EmptyQty
                                          Return
                                          }
                                         }
                                         If Tank_EmptyQty < Tank_EmptyRate then // test for final partial amount
                                         {
                                         Tank_TimeInc = Tank_TimeInc * Tank_EmptyQty / Tank_EmptyRate
//proportionalize
                                         Tank_EmptyInc = Tank_EmptyQty
                                         }
                                         Else Tank_EmptyInc = Tank_EmptyRate
                                         If Tank_Level [Tank_ID] < Tank_EmptyInc then // Test for insufficient qty
BEFORE Wait
                                         {
                                         Tank_TimeInc = Tank_TimeInc * Tank_Level [Tank_ID] / Tank_EmptyInc //
proportionalize
                                         Tank_EmptyInc = Tank_Level [Tank_ID]
                                         }
                                         if Tank_State[Tank_ID] <> Tank_Emptying
                                         then Tank_SetState(Tank_ID,Tank_Emptying)
                                         Wait Tank_TimeInc
                                         Wait Until Tank_State[Tank_ID] <> Tank_Down // Don't empty if Tank_ID is
down
                                         If Tank_Level [Tank_ID] < Tank_EmptyInc then // Test for insufficient qty AFTER
Wait
                                          Tank_EmptyInc = Tank_Level [Tank_ID]
                                         Dec Tank_EmptyQty, Tank_EmptyInc
                                         Dec Tank_Level [Tank_ID], Tank_EmptyInc
                                         Tank_UpdateStats (Tank_ID)
                                         }
                                         Tank_QtyLeft = 0.0

Tank_Transfer       None        Tank_FromID      Integer      // Transfers a quantity from one Tank_ID to another.
                      Tank_ToID         Integer
```

```
                    Tank_TransferQty   Real        Real Tank_ToQty
                    Tank_FromRate      Real        Real Tank_FromQty
                    Tank_ToRate        Real        Real Tank_TimeInc
                    Tank_ResumeLevel   Real        Real Tank_ToCapacity = Tank_Cap(Tank_ToID)

                                       Int Tank_VarRate = 0
                                       Wait Until Tank_State[Tank_FromID] <> Tank_Down    // Don't transfer if FROM
Tank_ID is down
                                       If Tank_FromRate <= 0.0 // test for variable rate transfer
                                       Then Tank_VarRate = 1 // used as a flag
                                       Else
                                        {
                                        Tank_FromRate = Tank_FromRate * Tank_TimeStep // convert units per clock
unit to per timestep
                                        If Tank_ToRate = 0.0
                                        Then Tank_ToRate = Tank_FromRate
                                        Else Tank_ToRate = Tank_ToRate * Tank_TimeStep // convert units per clock
unit to per timestep
                                        }
                                       While Tank_TransferQty > 0.0000001 do
                                       {
                                       Tank_TimeInc = Tank_TimeStep
                                       // Make sure there is available TO capacity
                                       If Tank_Level [Tank_ToID] >= Tank_ToCapacity  then
                                        {
                                        If Tank_ResumeLevel = Tank_Stop then
                                         {
                                         Tank_QtyLeft = Tank_TransferQty
                                         Return
                                         }
                                        If Tank_ResumeLevel = 0.0
                                        Then Stop "Attempted to fill "$Loc(Tank_ToID) $ " beyond its capacity of "
$Tank_ToCapacity
                                        Else If Tank_ResumeLevel  >= Tank_ToCapacity
                                        Then Stop "Resume Level in Tank_Transfer subroutine for
"$Loc(Tank_FromID)$ " must be less than its capacity"
                                        else  Wait Until Tank_Level[Tank_ToID] <= Tank_ResumeLevel
                                        }
                                       If Tank_Level [Tank_FromID] <= 0.0 then Wait Until Tank_Level [Tank_FromID]
> 0.0  // test for empty FROM tank_ID

                                       // For variable flow rate, call Tank_Rate subroutine to find rate
                                       If Tank_VarRate then
                                        {
                                        Tank_FromRate =Tank_Rate (Tank_FromID, Tank_ToID) * Tank_TimeStep //
convert var units per clock unit to per timestep
                                        if Tank_FromRate = 0.0 then
                                         {
                                         Tank_QtyLeft = Tank_TransferQty
                                         Return
                                         }
                                        Tank_ToRate = Tank_FromRate
                                        }

                                       // Test for final remaining quantity
                                       If Tank_TransferQty < Tank_FromRate then
                                        {
                                        Tank_FromQty = Tank_TransferQty  // reduce to remaining quantity
                                        Tank_TimeInc = Tank_TimeInc * Tank_FromQty / Tank_FromRate  //
proportionalize
                                        Tank_ToQty = Tank_ToRate * Tank_FromQty/ Tank_FromRate  //
proportionalize
                                        }
                                       Else
                                        {
                                        Tank_FromQty = Tank_FromRate
                                        Tank_ToQty = Tank_ToRate
                                        }

                                       // Adjust flow rates based on FROM level BEFORE time step
```

```
                                        If Tank_Level[Tank_FromID] < Tank_FromQty then
                                        {
                                        Tank_TimeInc = Tank_TimeInc * Tank_Level[Tank_FromID] / Tank_FromQty  //
proportionalize
                                        Tank_ToQty = Tank_ToQty * Tank_Level[Tank_FromID]/ Tank_FromQty  //
proportionalize
                                        Tank_FromQty = Tank_Level[Tank_FromID]  // reduce to remaining quantity
                                        }

                                       // Further adjust flow rates based on TO level BEFORE time step
                                       If Tank_ToCapacity - Tank_Level[Tank_ToID] < Tank_ToQty then
                                        {
                                        Tank_TimeInc = Tank_TimeInc *  (Tank_ToCapacity - Tank_Level[Tank_ToID])
/ Tank_ToQty
                                        Tank_FromQty = Tank_FromQty *  (Tank_ToCapacity -
Tank_Level[Tank_ToID]) / Tank_ToQty
                                        Tank_ToQty =  Tank_ToCapacity - Tank_Level[Tank_ToID]
                                        }
                                       if Tank_State[Tank_FromID] <> Tank_Emptying
                                       then Tank_SetState(Tank_FromID, Tank_Emptying)
                                       if Tank_State[Tank_ToID] <> Tank_Filling
                                       then Tank_SetState(Tank_ToID, Tank_Filling)
                                       Wait Tank_TimeInc
                                       Wait Until Tank_State[Tank_FromID] <> Tank_Down    // Don't transfer if FROM
Tank_ID is down
                                       // Adjust rate based on FROM level after time step
                                       If Tank_Level[Tank_FromID]  < Tank_FromQty  then
                                        {
                                        Tank_ToQty  = Tank_ToQty * Tank_Level[Tank_FromID]  / Tank_FromQty  //
proportionalize
                                        Tank_FromQty = Tank_Level[Tank_FromID]  // reduce to remaining qty
                                        }
                                       // Further adjust flow rates based on available TO capacity
                                       If Tank_ToCapacity - Tank_Level[Tank_ToID] < Tank_ToQty  then
                                        {
                                        Tank_FromQty = Tank_FromQty *  (Tank_ToCapacity -
Tank_Level[Tank_ToID]) / Tank_ToQty // proportionalize
                                        Tank_ToQty = Tank_ToCapacity - Tank_Level[Tank_ToID]
                                        }
                                       Dec Tank_Level [Tank_FromID], Tank_FromQty
                                       Tank_UpdateStats (Tank_FromID)
                                       Inc Tank_Level [Tank_ToID],  Tank_ToQty
                                       Tank_UpdateStats (Tank_ToID)
                                       Dec Tank_TransferQty, Tank_FromQty
                                       }
                                      Tank_QtyLeft = 0.0
                                      Inc Tank_Fills[Tank_ToID]

 Tank_TransferDownTo   None        Tank_FromID       Integer     // Transfers until FROM tank_ID drops to a specified
level
                          Tank_ToID         Integer
                          Tank_FromLevel    Real        if (Tank_FromLevel < 0.0) or (Tank_FromLevel >
Tank_Cap(Tank_FromID))
                          Tank_FromRate     Real        then Stop ""$Tank_FromLevel $ " is not a valid level for
"$Loc(Tank_FromID)$"  in Tank_TransferDownTo subroutine call"
                          Tank_ToRate       Real        if Tank_Level[Tank_FromID] <= Tank_FromLevel
                                       then Stop "Level specified in Tank_TransferDownTo is greater than or equal to
current level for "$loc(Tank_FromID)


                                       Real Tank_ToQty
                                       Real Tank_FromQty
                                       Real Tank_TimeInc
                                       Real Tank_ToCapacity = Tank_Cap(Tank_ToID)

                                       Int Tank_VarRate = 0
                                       Wait Until Tank_Level [Tank_FromID] > 0.0 // Don't start until there is at least
something in From Tank_ID
                                       Wait Until Tank_State[Tank_FromID] <> Tank_Down    // Don't transfer if FROM
Tank_ID is down
```

```
If Tank_FromRate <= 0.0 // test for variable rate transfer
Then Tank_VarRate = 1 // used as a flag
Else
 {
 Tank_FromRate = Tank_FromRate * Tank_TimeStep // convert units per clock unit to per timestep
 If Tank_ToRate = 0.0
 Then Tank_ToRate = Tank_FromRate
 Else Tank_ToRate = Tank_ToRate * Tank_TimeStep //convert units per clock unit to per timestep
 }
While Tank_Level[Tank_FromID] > Tank_FromLevel Do
 {
 Tank_TimeInc = Tank_TimeStep
 // Make sure there is capacity in TO tank_ID before time step or an infinite loop will occur
 If Tank_Level[Tank_ToID] >= Tank_ToCapacity
 then Wait Until Tank_Level [Tank_ToID] < Tank_ToCapacity

 // For variable flow rate, call Tank_Rate subroutine to find rate
 If Tank_VarRate then
  {
  Tank_FromRate =Tank_Rate(Tank_FromID, Tank_ToID) * Tank_TimeStep // convert var units per clock unit to per time step
  if Tank_FromRate <=  0.0 then Stop "Tank_Rate must return a value greater than 0 in Tank_TranferToLevel subroutine"
  Tank_ToRate = Tank_FromRate
  }

 // Adjust flow rates based on FROM contents
 If  Tank_Level[Tank_FromID] - Tank_FromLevel < Tank_FromRate then
  {
  Tank_FromQty =  Tank_Level[Tank_FromID] - Tank_FromLevel  // reduce to remaining qty
  Tank_TimeInc = Tank_TimeInc * Tank_FromQty / Tank_FromRate  // proportionalize
  Tank_ToQty = Tank_ToRate * Tank_FromQty / Tank_FromRate  // proportionalize
  }
 Else
  {
  Tank_FromQty = Tank_FromRate
  Tank_ToQty = Tank_ToRate
  }
 // Further adjust flow rates based on available TO capacity
 If Tank_ToCapacity - Tank_Level[Tank_ToID] < Tank_ToQty  then
  {
  Tank_FromQty = Tank_FromQty *  (Tank_ToCapacity - Tank_Level[Tank_ToID]) / Tank_ToQty
  Tank_TimeInc = Tank_TimeInc * (Tank_ToCapacity - Tank_Level[Tank_ToID]) / Tank_ToQty
  Tank_ToQty = Tank_ToCapacity - Tank_Level[Tank_ToID]
  }
 if Tank_State[Tank_FromID] <> Tank_Emptying
 then Tank_SetState(Tank_FromID, Tank_Emptying)
 if Tank_State[Tank_ToID] <> Tank_Filling
 then Tank_SetState(Tank_ToID, Tank_Filling)
 Wait Tank_TimeInc
 Wait Until Tank_State[Tank_FromID] <> Tank_Down    // Don't transfer if FROM Tank_ID is down
 // Adjust transfer qty based on FROM level AFTER time step
 If Tank_Level[Tank_FromID] - Tank_FromLevel < Tank_FromQty  then
  {
  Tank_ToQty  = Tank_ToQty * (Tank_Level[Tank_FromID] - Tank_FromLevel) / Tank_FromQty  // proportionalize
  Tank_FromQty = Tank_Level[Tank_FromID] - Tank_FromLevel  // reduce to remaining qty
  }
 // Further adjust flow rates based on available TO capacity
 If Tank_ToCapacity - Tank_Level[Tank_ToID] < Tank_ToQty  then
```

```
                                        {
                                        Tank_FromQty = Tank_FromQty *  (Tank_ToCapacity -
Tank_Level[Tank_ToID]) / Tank_ToQty // proportionalize
                                        Tank_ToQty = Tank_ToCapacity - Tank_Level[Tank_ToID]
                                        }
                                        Dec Tank_Level [Tank_FromID], Tank_FromQty
                                        Tank_UpdateStats (Tank_FromID)
                                        Inc Tank_Level [Tank_ToID],  Tank_ToQty
                                        Tank_UpdateStats (Tank_ToID)
                                        }

 Tank_TransferUpTo    None       Tank_FromID       Integer     // Transfers until TO tank_ID  rises to a specified level
                      Tank_ToID         Integer
                      Tank_ToLevel      Real       if (Tank_ToLevel < 0.0) or (Tank_ToLevel > Tank_Cap(Tank_ToID))
                      Tank_FromRate     Real        then Stop ""$Tank_ToLevel $ " is not a valid level for "
$Loc(Tank_ToID)$" in Tank_TransferUpTo subroutine call"
                      Tank_ToRate       Real       if Tank_Level[Tank_ToID] >= Tank_ToLevel
                                        then Stop "Level specified in Tank_TransferUpTo is less than or equal to current
level for "$loc(Tank_ToID)

                                        Real Tank_ToQty
                                        Real Tank_FromQty
                                        Real Tank_TimeInc

                                        Int Tank_VarRate = 0
                                        Wait Until Tank_State[Tank_FromID] <> Tank_Down    // Don't transfer if FROM
Tank_ID is down
                                        If Tank_FromRate <= 0.0 // test for variable rate transfer
                                        Then Tank_VarRate = 1 // used as a flag
                                        Else
                                        {
                                        Tank_FromRate = Tank_FromRate * Tank_TimeStep // convert units per clock
unit to per timestep
                                        If Tank_ToRate = 0.0
                                        Then Tank_ToRate = Tank_FromRate
                                        Else Tank_ToRate = Tank_ToRate * Tank_TimeStep //convert units per clock
unit to per timestep
                                        }
                                        While Tank_Level[Tank_ToID] < Tank_ToLevel Do
                                        {
                                        Tank_TimeInc = Tank_TimeStep
                                        // Make sure there is enough From qty BEFORE time step or an infinite loop will
occur
                                        If Tank_Level [Tank_FromID] <= 0.0
                                        then Wait Until Tank_Level [Tank_FromID] > 0.0

                                        // For variable flow rate, call Tank_Rate subroutine to find rate
                                        If Tank_VarRate then
                                        {
                                        Tank_FromRate =Tank_Rate(Tank_FromID, Tank_ToID) * Tank_TimeStep //
convert var units per clock unit to per time step
                                        if Tank_FromRate <=  0.0 then Stop "Tank_Rate must return a value greater
than 0 in Tank_TranferToLevel subroutine"
                                        Tank_ToRate = Tank_FromRate
                                        }

                                        // Adjust flow rates based on TO level BEFORE time step.
                                        If Tank_ToLevel - Tank_Level[Tank_ToID] < Tank_ToRate then
                                        {
                                        Tank_ToQty = Tank_ToLevel - Tank_Level[Tank_ToID]  // reduce to remaining
qty
                                        Tank_TimeInc = Tank_TimeInc * Tank_ToQty / Tank_ToRate  //  proportionalize
                                        Tank_FromRate = Tank_FromRate * Tank_ToQty / Tank_ToRate  //
proportionalize
                                        }
                                        Else
                                        {
                                        Tank_FromQty = Tank_FromRate
                                        Tank_ToQty = Tank_ToRate
                                        }
```

```
                                        // Further adjust flow rates based on available From qty and To level BEFORE
time step
                                        If Tank_Level [Tank_FromID] < Tank_FromRate  then   // test for less than From
qty
                                         {
                                         Tank_FromQty = Tank_Level [Tank_FromID]
                                         Tank_ToQty = Tank_ToRate * Tank_FromQty / Tank_FromRate
                                         Tank_TimeInc = Tank_TimeInc * Tank_FromQty / Tank_FromRate
                                         }
                                        if Tank_State[Tank_FromID] <> Tank_Emptying
                                        then Tank_SetState(Tank_FromID, Tank_Emptying)
                                        if Tank_State[Tank_ToID] <> Tank_Filling
                                        then Tank_SetState(Tank_ToID, Tank_Filling)
                                        Wait Tank_TimeInc
                                        Wait Until Tank_State[Tank_FromID] <> Tank_Down    // Don't transfer if FROM
Tank_ID is down

                                        // Adjust transfer qty based on levels AFTER time step
                                        If Tank_ToLevel - Tank_Level[Tank_ToID] < Tank_ToQty  then
                                         {
                                         if Tank_Level[Tank_ToID] >= Tank_ToLevel then Return  // terminate if already
filled to level
                                         Tank_FromQty = Tank_FromQty * (Tank_ToLevel - Tank_Level[Tank_ToID]) /
Tank_ToQty  // proportionalize
                                         Tank_ToQty = Tank_ToLevel - Tank_Level[Tank_ToID]  // reduce to remaining
qty
                                         }
                                        If Tank_Level [Tank_FromID] < Tank_FromQty  then
                                         {
                                         Tank_ToQty = Tank_ToQty *  Tank_Level[Tank_FromID] /  Tank_FromQty
                                         Tank_FromQty = Tank_Level [Tank_FromID]
                                         }
                                        Dec Tank_Level [Tank_FromID], Tank_FromQty
                                        Tank_UpdateStats (Tank_FromID)
                                        Inc Tank_Level [Tank_ToID],  Tank_ToQty
                                        Tank_UpdateStats (Tank_ToID)
                                        }

 Tank_Inc          None      Tank_ID         Integer     // Instantly increases the level of a Tank_ID by a specified quantity.
                       Tank_FillQty     Real       // If the Tank_ID has insufficient capacity, it is filled as capacity becomes
available.

                                        Real Tank_FillAmt

                                        While Tank_FillQty > 0 do
                                         {
                                         Tank_FillAmt = Tank_FillQty
                                         Wait Until Tank_Level[Tank_ID] <  Cap (Loc(Tank_ID)) // wait until Tank_ID is
not full
                                         If Tank_FreeCap(Tank_ID) < Tank_FillAmt then Tank_FillAmt =
Tank_FreeCap(Tank_ID)
                                         Dec Tank_FillQty, Tank_FillAmt
                                         Inc Tank_Level[Tank_ID], Tank_FillAmt
                                         Tank_UpdateStats(Tank_ID)
                                         }

 Tank_Dec          None      Tank_ID         Integer     // Instantly decreases the level of a Tank_ID by a specified
quantity.
                       Tank_EmptyQty     Real       // If the Tank_ID has insufficient quantity, it is emptied as material
becomes available.

                                        Real Tank_EmptyAmt

                                        While Tank_EmptyQty > 0 do
                                         {
                                         Tank_EmptyAmt = Tank_EmptyQty
                                         Wait Until Tank_Level[Tank_ID] > 0.0  // wait until Tank_ID is not empty
                                         If Tank_Level [Tank_ID] < Tank_EmptyAmt then Tank_EmptyAmt = Tank_Level
[Tank_ID]
```

```
                                        Dec Tank_EmptyQty, Tank_EmptyAmt
                                        Dec Tank_Level[Tank_ID], Tank_EmptyAmt
                                        Tank_UpdateStats(Tank_ID)
                                        }

 Tank_FallTrigger     None       Tank_ID          Integer     // Waits until a Tank_ID falls to a specified level to trigger some
action
                      Tank_FallLevel   Real
                                        Wait Until Tank_Level [tank_ID]  > Tank_FallLevel  // wait until level first rises
above trigger level
                                        Wait Until Tank_Level [tank_ID]  <= Tank_FallLevel // wait until level drops to
trigger level

 Tank_RiseTrigger     None       Tank_ID          Integer     // Waits until a Tank_ID rises to a specified level to trigger some
action
                      Tank_RiseLevel   Real
                                        Wait Until Tank_Level [tank_ID]  <  Tank_RiseLevel  // wait until level first falls
below trigger level
                                        Wait Until Tank_Level [tank_ID]  >= Tank_RiseLevel // wait until level rises to
trigger level

 Tank_Prep          None       Tank_ID          Integer     // Prepares or cleans a Tank_ID before filling.
                      Tank_PrepTime    Real       // Time is counted as Setup time.

                                        Tank_Product[Tank_ID] = 0   // Clears out current product type
                                        Tank_SetState (Tank_ID, Tank_Setup)
                                        Wait Tank_PrepTime
                                        Tank_SetState (Tank_ID, Tank_Idle)


 Tank_SetLevel      None       Tank_ID          Integer     // Instantly sets the Tank_ID level to the specified amount
                      Tank_SetQty      Real       // If less than 0 or greater than the capacity, an error occurs.

                                        If (Tank_SetQty < 0.0) or (Tank_SetQty > Tank_Cap(Tank_ID)) then
                                         Stop "Set level " $ Tank_SetQty  $ " is out of range for " $ Loc(Tank_ID)
                                        Tank_Level[Tank_ID] = Tank_SetQty
                                        Tank_UpdateStats(Tank_ID)

 Tank_DoOperation     None       Tank_ID          Integer     // Delays some time for an operation such as mixing.
                      Tank_OperationTime Real       // Time is counted as Operation  time.

                                        Tank_SetState (Tank_ID, Tank_Operation)
                                        Wait Tank_OperationTime
                                        Tank_SetState (Tank_ID, Tank_Blocked)

 Tank_GoDown        None       Tank_ID          Integer     // Sets Tank_ID state to Tank_Down for a specified time and
then sets it to state prior to downtime.
                      Tank_DownTime    Real       // Downtime statistics are updated.

                                        int Tank_LastState = Tank_State [Tank_ID]
                                        Inc Tank_Statistics[Tank_ID, 14] // Current number of DTs
                                        if Tank_State [Tank_ID] <> Tank_Down
                                        then
                                         {
                                         Inc Tank_Statistics[Tank_ID,3], Tank_Statistics[Tank_ID,1] * (Clock() -
Tank_Statistics[Tank_ID,2]) // inc cum level
                                         Tank_Statistics[Tank_ID,2] = Clock() // Update time since last content change
                                         Tank_SetState (Tank_ID, Tank_Down)
                                         }
                                        Wait Tank_Downtime
                                        Dec Tank_Statistics [Tank_ID,14]
                                        If (Tank_Statistics[Tank_ID, 14] = 0) and (Tank_State[Tank_ID] = Tank_Down)
                                        then
                                         {
                                         Tank_Statistics[Tank_ID,2] = Clock() // Update time since last content change
without changing contents
                                         Tank_SetState (Tank_ID, Tank_LastState)
                                         }
```

```
Tank_GoDownSched    None    Tank_ID        Integer    // Sets Tank_ID state to Tank_Down for a specified time and
then sets it to state prior to downtime.
                            Tank_DownTime    Real      // Downtime statistics are NOT updated.

                                                    int Tank_LastState = Tank_State [Tank_ID]
                                                    Inc Tank_Statistics[Tank_ID, 14] // Current number of DTs
                                                    if Tank_State [Tank_ID] <> Tank_Down then
                                                    {
                                                     Tank_State[Tank_ID] =  Tank_Down
                                                     Tank_Statistics [Tank_ID, 6] = Clock()  // update last state change time
                                                    }
                                                    Wait Tank_Downtime
                                                    Dec Tank_Statistics [Tank_ID,14]
                                                    If (Tank_Statistics[Tank_ID, 14] = 0) and (Tank_State[Tank_ID] = Tank_Down)
then
                                                    {
                                                     // Tank_SetState (Tank_ID, Tank_LastState) --> Removed this line and replaced
it with the following two lines
                                                     Tank_State[Tank_ID] = Tank_LastState
                                                     Tank_Statistics[Tank_ID, 6] = Clock() // update last state change time
                                                    }
 Tank_SetState      None      Tank_ID        Integer    // Updates Tank_Statistics and sets the Tank_State.
                            SetState        Integer
                                                    Int Stat_Index = Tank_State[Tank_ID] + 7
                                                    Inc Tank_Statistics [Tank_ID, Stat_Index], Clock() - Tank_Statistics [Tank_ID, 6]
// inc cum state time
                                                    Tank_Statistics [Tank_ID,6] = Clock()  // update last state change time
                                                    Tank_State[Tank_ID] = SetState


 Tank_FreeCap       Real      Tank_ID        Integer    // Returns the available capacity of a Tank_ID

                                                    Return Cap(Loc(Tank_ID)) - Tank_Level [Tank_ID]
 Tank_Cap           Real      Tank_ID        Integer    // Returns the defined capacity of a Tank_ID

                                                    Return Cap(Loc(Tank_ID))
 Tank_UpdateStats   None      Tank_ID        Integer    // Updates the statistics for a Tank_ID  whenever the Level
changes.
                                                    // It is called automatically in all of the pre-defined subroutines that fill, empty and
transfer.

                                                    // Tank_Statistics array column numbers referenced:
                                                    // 1 Last level
                                                    // 2 Last change time
                                                    // 3 Cum time-weighted level
                                                    // 4 Entries
                                                    // 5 Max contents

                                                    Inc Tank_Statistics[Tank_ID,3], Tank_Statistics[Tank_ID,1] * (Clock() -
Tank_Statistics[Tank_ID,2]) // inc cum level
                                                    Tank_Statistics[Tank_ID,2] = Clock() // Update time since last change
                                                    If Tank_Level[Tank_ID] > Tank_Statistics[Tank_ID,1]  then  // test for increase in
level
                                                    {
                                                     Inc Tank_Statistics[Tank_ID,4], Tank_Level[Tank_ID] -
Tank_Statistics[Tank_ID,1]   // increment unit entries
                                                     If Tank_Level[Tank_ID] > Tank_Statistics[Tank_ID,5] Then  // test for new max
conts
                                                      Tank_Statistics[Tank_ID,5] = Tank_Level[Tank_ID]  // assign new max conts
                                                     if Tank_Level[Tank_ID] = Cap(Loc(Tank_ID)) Then // test to see if now full
                                                     {
                                                      if Tank_State[Tank_ID] <> Tank_Down then
                                                      Tank_SetState (Tank_ID, Tank_Blocked)
                                                     }
                                                     Else if Tank_State [Tank_ID] <> Tank_Filling then
                                                      Tank_SetState (Tank_ID, Tank_Filling)
                                                    }
                                                    Else if Tank_Level[Tank_ID] < Tank_Statistics[Tank_ID,1] then  // must be a drop
in level
                                                    {
```

```
                                        If Tank_Level[Tank_ID] = 0 Then  // test to see if now empty
                                         Tank_SetState (Tank_ID, Tank_Idle)
                                        Else if Tank_State [Tank_ID] <> Tank_Emptying Then
                                         Tank_State [Tank_ID] = Tank_Emptying
                                       }
                                       Tank_Statistics[Tank_ID,1] = Tank_Level[Tank_ID]  // Update last level to current
level

  Tank_Rate            Real        Tank_FromID       Integer     // Used to dynamically change a flow rate for an emptying
FROM tank_ID and/or a filling TO tank_ID.
                         Tank_ToID         Integer     // The user must modify this routine to return the desired rate.

                                       //  For example, if the flow rate from TankA to TankB decreases from 150 gpm  to
100 gpm when TankB  rises above 4000,
                                       //  the following code might be entered:

                                       //  If (Tank_FromID = TankA) and (Tank_ToID = TankB)
                                       //  then if Tank_Level[Tank_FromID] > 4000
                                       //  then return 100
                                       //  else return 150

                                       // Replace the following code with your code.
                                       if Tank_FromID  = 0
                                       then Stop "Passing 0 as the fill rate for " $ loc(Tank_ToID) $ " requires that you
return a rate value using the subroutine called Tank_Rate."
                                       else Stop "Passing 0 as the empty or transfer rate for " $ loc(Tank_FromID) $ "
requires that you return a rate value using the subroutine called Tank_Rate."


  Tank_SelectOutput     Integer     Tank_Start        Integer     // Selects an output tank_ID according to a selection rule.
Works for up to 10 outputs.
                         Tank_Qty          Integer
                         Tank_Rule         Integer     int Tank_Index
                         Tank_Limit        Real        Int Tank_WithLongestIdleTime
                         Tank_Prod         Integer     Real Tank_LongestIdleTime

                                      Tank_Loop
                                      {
                                      If Tank_Limit > 0 then    // test for OK to fill a partial tank_ID
                                       {
                                       Tank_Index = Tank_Start
                                       while Tank_Index < Tank_Start + Tank_Qty  do
                                        {

                                        //Test for partially full tank_ID with level equal to or less  than the maximum
level and same product type.
                                        // To select a partially full tank_ID only when it is filling and not emptying, add
test for Tank_State[Tank_Index] <> Tank_Emptying. (see added change further below)
                                        if (Tank_Level [Tank_Index] > 0) and (Tank_Level [Tank_Index] <= Tank_Limit)
and (Tank_Product[Tank_Index] = Tank_Prod)
                                         Then Return Tank_Index
                                         Else Inc Tank_Index
                                         }
                                        }

                                      // Test for idle tank_ID  that meets selection rule
                                      Tank_Index = Tank_Start
                                      If Tank_Rule = Tank_InOrder
                                      Then while Tank_Index < Tank_Start + Tank_Qty  do
                                       If Tank_State[Tank_Index] = Tank_Idle
                                       Then Return Tank_Index
                                       Else Inc Tank_Index
                                      Else if Tank_Rule = Tank_LongestIdle then
                                       {
                                       Tank_WithLongestIdleTime  = 0
                                       Tank_LongestIdleTime = 999999999.0
                                       while Tank_Index < Tank_Start + Tank_Qty  do
                                        {
                                        If Tank_State [Tank_Index] = Tank_Idle
                                        Then if Tank_Statistics[Tank_Index, 6] < Tank_LongestIdleTime then
```

```
                                        {
                                        Tank_LongestIdleTime = Tank_Statistics[Tank_Index, 6]
                                        Tank_WithLongestIdleTime = Tank_Index
                                        }
                                       Inc Tank_Index
                                       }
                                      If Tank_WithLongestIdleTime  > 0 then Return Tank_WithLongestIdleTime
                                      }
                                     Else Stop "Invalid output selection rule defined for "$ Loc(Tank_Start)

                                     //  Wait for a tank_ID to become available
                                     If Tank_Limit > 0 // Eliminate this test if a partial tank_ID can only fill when
already filling and not emptying.
                                     then Wait 1 min  // if waiting on a limit, check every minute since it is dependent
on product type as well
                                     else // Wait for the first tank_ID to become idle
                                      If Tank_Qty < 2
                                      then Wait Until Tank_State[Tank_Start] = Tank_Idle
                                      else If Tank_Qty < 3
                                      then Wait Until (Tank_State[Tank_Start] = Tank_Idle) or
(Tank_State[Tank_Start+1] = Tank_Idle)
                                      else If Tank_Qty < 4
                                      then Wait Until (Tank_State[Tank_Start] = Tank_Idle) or
(Tank_State[Tank_Start+1] = Tank_Idle) or (Tank_State[Tank_Start+2] = Tank_Idle)
                                      else If Tank_Qty < 5
                                      then Wait Until (Tank_State[Tank_Start] = Tank_Idle) or
(Tank_State[Tank_Start+1] = Tank_Idle) or (Tank_State[Tank_Start+2] = Tank_Idle)
                                       or (Tank_State[Tank_Start+3] = Tank_Idle)
                                      else If Tank_Qty < 6
                                      then Wait Until (Tank_State[Tank_Start] = Tank_Idle) or
(Tank_State[Tank_Start+1] = Tank_Idle) or (Tank_State[Tank_Start+2] = Tank_Idle)
                                       or (Tank_State[Tank_Start+3] = Tank_Idle)or (Tank_State[Tank_Start+4] =
Tank_Idle)
                                      else If Tank_Qty < 7
                                      then Wait Until (Tank_State[Tank_Start] = Tank_Idle) or
(Tank_State[Tank_Start+1] = Tank_Idle) or (Tank_State[Tank_Start+2] = Tank_Idle)
                                       or (Tank_State[Tank_Start+3] = Tank_Idle) or (Tank_State[Tank_Start+4] =
Tank_Idle) or (Tank_State[Tank_Start+5] = Tank_Idle)
                                      else If Tank_Qty < 8
                                      then Wait Until (Tank_State[Tank_Start] = Tank_Idle) or
(Tank_State[Tank_Start+1] = Tank_Idle) or (Tank_State[Tank_Start+2] = Tank_Idle)
                                       or (Tank_State[Tank_Start+3] = Tank_Idle) or (Tank_State[Tank_Start+4] =
Tank_Idle) or (Tank_State[Tank_Start+5] = Tank_Idle)
                                       or (Tank_State[Tank_Start+6] = Tank_Idle)
                                      else If Tank_Qty < 9
                                      then Wait Until (Tank_State[Tank_Start] = Tank_Idle) or
(Tank_State[Tank_Start+1] = Tank_Idle) or (Tank_State[Tank_Start+2] = Tank_Idle)
                                       or (Tank_State[Tank_Start+3] = Tank_Idle) or (Tank_State[Tank_Start+4] =
Tank_Idle) or (Tank_State[Tank_Start+5] = Tank_Idle)
                                       or (Tank_State[Tank_Start+6] = Tank_Idle) or (Tank_State[Tank_Start+7] =
Tank_Idle)
                                      else If Tank_Qty < 10
                                      then Wait Until (Tank_State[Tank_Start] = Tank_Idle) or
(Tank_State[Tank_Start+1] = Tank_Idle) or (Tank_State[Tank_Start+2] = Tank_Idle)
                                       or (Tank_State[Tank_Start+3] = Tank_Idle) or (Tank_State[Tank_Start+4] =
Tank_Idle) or (Tank_State[Tank_Start+5] = Tank_Idle)
                                       or (Tank_State[Tank_Start+6] = Tank_Idle) or (Tank_State[Tank_Start+7] =
Tank_Idle) or (Tank_State[Tank_Start+8] = Tank_Idle)
                                      else If Tank_Qty < 11
                                      then Wait Until (Tank_State[Tank_Start] = Tank_Idle) or
(Tank_State[Tank_Start+1] = Tank_Idle) or (Tank_State[Tank_Start+2] = Tank_Idle)
                                       or (Tank_State[Tank_Start+3] = Tank_Idle) or (Tank_State[Tank_Start+4] =
Tank_Idle) or (Tank_State[Tank_Start+5] = Tank_Idle)
                                       or (Tank_State[Tank_Start+6] = Tank_Idle) or (Tank_State[Tank_Start+7] =
Tank_Idle) or (Tank_State[Tank_Start+8] = Tank_Idle)
                                       or (Tank_State[Tank_Start+9] = Tank_Idle)
                                      }
 Tank_SelectInput     Integer     Tank_Start        Integer     // Selects an input tank_ID according to a rule. Works for up to
10 inputs.
                          Tank_Qty          Integer
```

```
Tank_Rule          Integer      int Tank_Index
Tank_Limit         Real         Int Tank_WithLongestBlockedTime
Tank_Prod          Integer      Real Tank_LongestBlockedTime

                                Tank_Loop
                                {
                                If Tank_Limit > 0 then    // test for OK to draw froml a partial tank_ID
                                 {
                                 Tank_Index = Tank_Start
                                 while Tank_Index < Tank_Start + Tank_Qty  do
                                  {

                                  // test for partially full tank_ID with level equal to or greater than the minimum
level and same product type.
                                  // To select only a tank_ID that is not being filled, add test for
Tank_State[Tank_Index] <> Tank_Filling. (See additional note further below)
                                  if (Tank_FreeCap(Tank_Index) >  0) and (Tank_Level [Tank_Index] >=
Tank_Limit) and (Tank_Product[Tank_Index] = Tank_Prod)
                                   Then Return Tank_Index
                                   Else Inc Tank_Index
                                   }
                                }

                                // Test for blocked tank_ID that meets selection rule
                                Tank_Index = Tank_Start
                                If Tank_Rule = Tank_InOrder
                                Then while Tank_Index < Tank_Start + Tank_Qty  do
                                 If Tank_State[Tank_Index] = Tank_Blocked
                                 Then Return Tank_Index
                                 Else Inc Tank_Index
                                Else if Tank_Rule = Tank_LongestBlocked  then
                                 {
                                 Tank_WithLongestBlockedTime  = 0
                                 Tank_LongestBlockedTime =  999999999.0
                                 while Tank_Index < Tank_Start + Tank_Qty  do
                                  {
                                  If Tank_State [Tank_Index] = Tank_Blocked
                                  Then if Tank_Statistics[Tank_Index, 6] < Tank_LongestBlockedTime then
                                   {
                                   Tank_LongestBlockedTime = Tank_Statistics[Tank_Index, 6]
                                   Tank_WithLongestBlockedTime = Tank_Index
                                   }
                                  Inc Tank_Index
                                  }
                                 If Tank_WithLongestBlockedTime  > 0 then Return
Tank_WithLongestBlockedTime
                                }
                                Else Stop "Invalid input selection rule defined for "$ Loc(Tank_Start)

                                //  Wait for a tank_ID to have material for input
                                If Tank_Limit > 0   // If partial tanks can only selected if currently emptying (not
filling), eliminate this test.
                                then Wait 1 min    // if waiting on a limit, check every minute since it is dependent
on product type as well
                                else // Wait for the first tank_ID to become blocked (full)
                                 If Tank_Qty < 2
                                 then Wait Until Tank_State[Tank_Start] = Tank_Blocked
                                 else If Tank_Qty < 3
                                 then Wait Until (Tank_State[Tank_Start] = Tank_Blocked) or
(Tank_State[Tank_Start+1] = Tank_Blocked)
                                  else If Tank_Qty < 4
                                  then Wait Until (Tank_State[Tank_Start] = Tank_Blocked) or
(Tank_State[Tank_Start+1] = Tank_Blocked) or (Tank_State[Tank_Start+2] = Tank_Blocked)
                                  else If Tank_Qty < 5
                                  then Wait Until (Tank_State[Tank_Start] = Tank_Blocked) or
(Tank_State[Tank_Start+1] = Tank_Blocked) or (Tank_State[Tank_Start+2] = Tank_Blocked)
                                   or (Tank_State[Tank_Start+3] = Tank_Blocked)
                                  else If Tank_Qty < 6
                                  then Wait Until (Tank_State[Tank_Start] = Tank_Blocked) or
(Tank_State[Tank_Start+1] = Tank_Blocked) or (Tank_State[Tank_Start+2] = Tank_Blocked)
```

```
                                                or (Tank_State[Tank_Start+3] = Tank_Blocked)or (Tank_State[Tank_Start+4] =
Tank_Blocked)
                                               else If Tank_Qty < 7
                                               then Wait Until (Tank_State[Tank_Start] = Tank_Blocked) or
(Tank_State[Tank_Start+1] = Tank_Blocked) or (Tank_State[Tank_Start+2] = Tank_Blocked)
                                                or (Tank_State[Tank_Start+3] = Tank_Blocked) or (Tank_State[Tank_Start+4] =
Tank_Blocked) or (Tank_State[Tank_Start+5] = Tank_Blocked)
                                               else If Tank_Qty < 8
                                               then Wait Until (Tank_State[Tank_Start] = Tank_Blocked) or
(Tank_State[Tank_Start+1] = Tank_Blocked) or (Tank_State[Tank_Start+2] = Tank_Blocked)
                                                or (Tank_State[Tank_Start+3] = Tank_Blocked) or (Tank_State[Tank_Start+4] =
Tank_Blocked) or (Tank_State[Tank_Start+5] = Tank_Blocked)
                                                or (Tank_State[Tank_Start+6] = Tank_Blocked)
                                               else If Tank_Qty < 9
                                               then Wait Until (Tank_State[Tank_Start] = Tank_Blocked) or
(Tank_State[Tank_Start+1] = Tank_Blocked) or (Tank_State[Tank_Start+2] = Tank_Blocked)
                                                or (Tank_State[Tank_Start+3] = Tank_Blocked) or (Tank_State[Tank_Start+4] =
Tank_Blocked) or (Tank_State[Tank_Start+5] = Tank_Blocked)
                                                or (Tank_State[Tank_Start+6] = Tank_Blocked) or (Tank_State[Tank_Start+7] =
Tank_Blocked)
                                               else If Tank_Qty < 10
                                               then Wait Until (Tank_State[Tank_Start] = Tank_Blocked) or
(Tank_State[Tank_Start+1] = Tank_Blocked) or (Tank_State[Tank_Start+2] = Tank_Blocked)
                                                or (Tank_State[Tank_Start+3] = Tank_Blocked) or (Tank_State[Tank_Start+4] =
Tank_Blocked) or (Tank_State[Tank_Start+5] = Tank_Blocked)
                                                or (Tank_State[Tank_Start+6] = Tank_Blocked) or (Tank_State[Tank_Start+7] =
Tank_Blocked) or (Tank_State[Tank_Start+8] = Tank_Blocked)
                                               else If Tank_Qty < 11
                                               then Wait Until (Tank_State[Tank_Start] = Tank_Blocked) or
(Tank_State[Tank_Start+1] = Tank_Blocked) or (Tank_State[Tank_Start+2] = Tank_Blocked)
                                                or (Tank_State[Tank_Start+3] = Tank_Blocked) or (Tank_State[Tank_Start+4] =
Tank_Blocked) or (Tank_State[Tank_Start+5] = Tank_Blocked)
                                                or (Tank_State[Tank_Start+6] = Tank_Blocked) or (Tank_State[Tank_Start+7] =
Tank_Blocked) or (Tank_State[Tank_Start+8] = Tank_Blocked)
                                                or (Tank_State[Tank_Start+9] = Tank_Blocked)
                                               }
 LlenadoTanksI1        None                            L1:

                                               Tank_SetLevel(I1, 50000)

Tank_FallTrigger(I1,500)


                                               GOTO L1
 LlenadoTanksI2        None
                                               L1:

                                               Tank_SetLevel(I2, 20000)

Tank_FallTrigger(I2,500)



                                               GOTO L1
 LlenadoTanksI3        None
                                               L1:

                                               Tank_SetLevel(I3, 20000)

Tank_FallTrigger(I3,500)


                                               GOTO L1
 LlenadoTanksI4        None
                                               L1:

                                               Tank_SetLevel(I4, 30000)

Tank_FallTrigger(I4,500)
```

LlenadoTanksI5 None GOTO L1
L1:
Tank_SetLevel(I5, 30000)
Tank_FallTrigger(I5,500)

LlenadoTanksI6 None GOTO L1
L1:
Tank_SetLevel(I6, 50000)
Tank_FallTrigger(I6,500)

LlenadoTanksI7 None GOTO L1
L1:
Tank_SetLevel(I7, 100)
Tank_FallTrigger(I7,50)

LlenadoTanksI8 None GOTO L1
L1:
Tank_SetLevel(I8, 4400)
Tank_FallTrigger(I8,500)

LlenadoTanksI9 None GOTO L1
L1:
Tank_SetLevel(I9, 2400)
Tank_FallTrigger(I9,500)

LlenadoTanksI10 None GOTO L1
L1:
Tank_SetLevel(I10, 1000)
Tank_FallTrigger(I10,250)

PreparacionTanqueB1_1 None Cantidad Integer GOTO L1
Tank_Transfer(I1, Premix1, Cantidad * 0.31621 , 97.7+L(29.4,25.5), 0, 0) // Cantidad de aceite faltante para finalizar el lote
Tank_Transfer(I2, Premix1, Cantidad* 0.091, 1+L(251,112), 0, 0)
Tank_Transfer(I3, Premix1, Cantidad* 0.077, 1+L(251,112), 0, 0)
Tank_Transfer(I4, Premix1, Cantidad* 0.1120, 1+L(248,144), 0, 0)
Tank_Transfer(I5, Premix1, Cantidad* 0.1050, 1+L(248,144), 0, 0)
Tank_Transfer(I6, Premix1, Cantidad* 0.15753, 309 + L(614,951), 0, 0)
wait 1 min
Tank_Inc(Premix1,Cantidad* 0.00006)
Tank_Dec(I7,Cantidad* 0.00006)
//Tank_Transfer(I7, Premix1, Cantidad* 0.00006, N(0.305,0.152), 0, 0)
Tank_Transfer(I8, Premix1, Cantidad* 0.125 ,245+L(131,536), 0, 0)
Tank_Transfer(I9, Premix1, Cantidad* 0.01111,20.6+L(78.8,354) , 0, 0)
Tank_Transfer(I10, Premix1,Cantidad* 0.00509,1+L(1.63,0.644), 0, 0)
Tank_DoOperation(Premix1, 10)

```
PreparacionTanqueB1_2 None Cantidad Integer Tank_Transfer(I1, Premix1, Cantidad * 0.31891, 97.7+
L(29.4,25.5), 0, 0)  // Cantidad de aceite faltante para finalizar el lote
                                        Tank_Transfer(I2, Premix1, Cantidad* 0.0715, 1+L(251,112), 0, 0)
                                        Tank_Transfer(I3, Premix1, Cantidad* 0.0605, 1+L(251,112), 0, 0)
                                        Tank_Transfer(I4, Premix1, Cantidad* 0.088, 1+L(248,144), 0, 0)
                                        Tank_Transfer(I5, Premix1, Cantidad* 0.0825, 1+L(248,144), 0, 0)
                                        Tank_Transfer(I6, Premix1, Cantidad* 0.25076, 309 + L(614,951), 0, 0)
                                        wait 1 min
                                        Tank_Inc(Premix1,Cantidad* 0.00006)
                                        Tank_Dec(I7,Cantidad* 0.00006)
                                        //Tank_Transfer(I7, Premix1, Cantidad* 0.00006, N(0.305,0.152), 0, 0)
                                        Tank_Transfer(I8, Premix1, Cantidad* 0.085 , 245+L(131,536) , 0, 0)
                                        Tank_Transfer(I9, Premix1, Cantidad* 0.03611,20.6+L(78.8,354) , 0, 0)
                                        Tank_Transfer(I10, Premix1,Cantidad* 0.00666,1+L(1.63,0.644), 0, 0)
                                        Tank_DoOperation(Premix1, 10)

PreparacionTanqueB1_3 None Cantidad Integer Tank_Transfer(I1, Premix1, Cantidad * 0.294, 97.7+
L(29.4,25.5), 0, 0)  // Cantidad de aceite faltante para finalizar el lote
                                        Tank_Transfer(I2, Premix1, Cantidad* 0.0845, 1+L(251,112), 0, 0)
                                        Tank_Transfer(I3, Premix1, Cantidad* 0.0715, 1+L(251,112), 0, 0)
                                        Tank_Transfer(I4, Premix1, Cantidad* 0.0975, 1+L(248,144), 0, 0)
                                        Tank_Transfer(I5, Premix1, Cantidad* 0.104, 1+L(248,144), 0, 0)
                                        Tank_Transfer(I6, Premix1, Cantidad* 0.2076, 309 + L(614,951), 0, 0)
                                        wait 1 min
                                        Tank_Inc(Premix1,Cantidad* 0.0001)
                                        Tank_Dec(I7,Cantidad* 0.0001)
                                        //Tank_Transfer(I7, Premix1, Cantidad* 0.0001, N(0.305,0.152), 0, 0)
                                        Tank_Transfer(I8, Premix1, Cantidad* 0.125, 245+L(131,536), 0, 0)
                                        Tank_Transfer(I9, Premix1, Cantidad* 0.0110,20.6+L(78.8,354) , 0, 0)
                                        Tank_Transfer(I10, Premix1,Cantidad* 0.0048,1+L(1.63,0.644), 0, 0)
                                        Tank_DoOperation(Premix1, 10)
PreparacionTanqueB1_4 None Cantidad Integer Tank_Transfer(I1, Premix1, Cantidad * 0.6149, 97.7+
L(29.4,25.5), 0, 0)  // Cantidad de aceite faltante para finalizar el lote
                                        Tank_Transfer(I2, Premix1, Cantidad* 0, 1+L(251,112), 0, 0)
                                        Tank_Transfer(I3, Premix1, Cantidad* 0, 1+L(251,112), 0, 0)
                                        Tank_Transfer(I4, Premix1, Cantidad* 0, 1+L(248,144), 0, 0)
                                        Tank_Transfer(I5, Premix1, Cantidad* 0.063, 1+L(248,144), 0, 0)
                                        Tank_Transfer(I6, Premix1, Cantidad* 0.1895, 309 + L(614,951), 0, 0)
                                        wait 1 min
                                        Tank_Inc(Premix1,Cantidad* 0.0234)
                                        Tank_Dec(I7,Cantidad* 0.0234)
                                        //Tank_Transfer(I7, Premix1, Cantidad* 0.0234, N(0.305,0.152), 0, 0)      // BHT
demora 1 min
                                        Tank_Transfer(I8, Premix1, Cantidad* 0.105, 245+L(131,536), 0, 0)
                                        Tank_Transfer(I9, Premix1, Cantidad* 0, 20.6+L(78.8,354) , 0, 0)
                                        Tank_Transfer(I10, Premix1,Cantidad* 0.0042,1+L(1.63,0.644), 0, 0)
                                        Tank_DoOperation(Premix1, 10)
PreparacionTanqueB1_5 None Cantidad Integer
                                        Tank_Transfer(I1, Premix1, Cantidad * 0.6149, 97.7+ L(29.4,25.5), 0, 0)  //
Cantidad de aceite faltante para finalizar el lote
                                        Tank_Transfer(I2, Premix1, Cantidad* 0, 1+L(251,112), 0, 0)
                                        Tank_Transfer(I3, Premix1, Cantidad* 0, 1+L(251,112), 0, 0)
                                        Tank_Transfer(I4, Premix1, Cantidad* 0, 1+L(248,144), 0, 0)
                                        Tank_Transfer(I5, Premix1, Cantidad* 0.063, 1+L(248,144), 0, 0)
                                        Tank_Transfer(I6, Premix1, Cantidad* 0.1895, 309 + L(614,951), 0, 0)
                                        wait 1 min
                                        Tank_Inc(Premix1,Cantidad* 0.0234)
                                        Tank_Dec(I7,Cantidad* 0.0234)
                                        //Tank_Transfer(I7, Premix1, Cantidad* 0.0234, , 0, 0)      // Siempre demora 1
min.
                                        Tank_Transfer(I8, Premix1, Cantidad* 0.105, 245+L(131,536), 0, 0)
                                        Tank_Transfer(I9, Premix1, Cantidad* 0,20.6+L(78.8,354) , 0, 0)
                                        Tank_Transfer(I10, Premix1,Cantidad* 0.0042,1+L(1.63,0.644), 0, 0)
                                        Tank_DoOperation(Premix1, 10)
PreparacionTanqueB2_1 None Cantidad Integer Tank_Transfer(I1, Premix2, Cantidad * 0.31621 , 97.7+
```

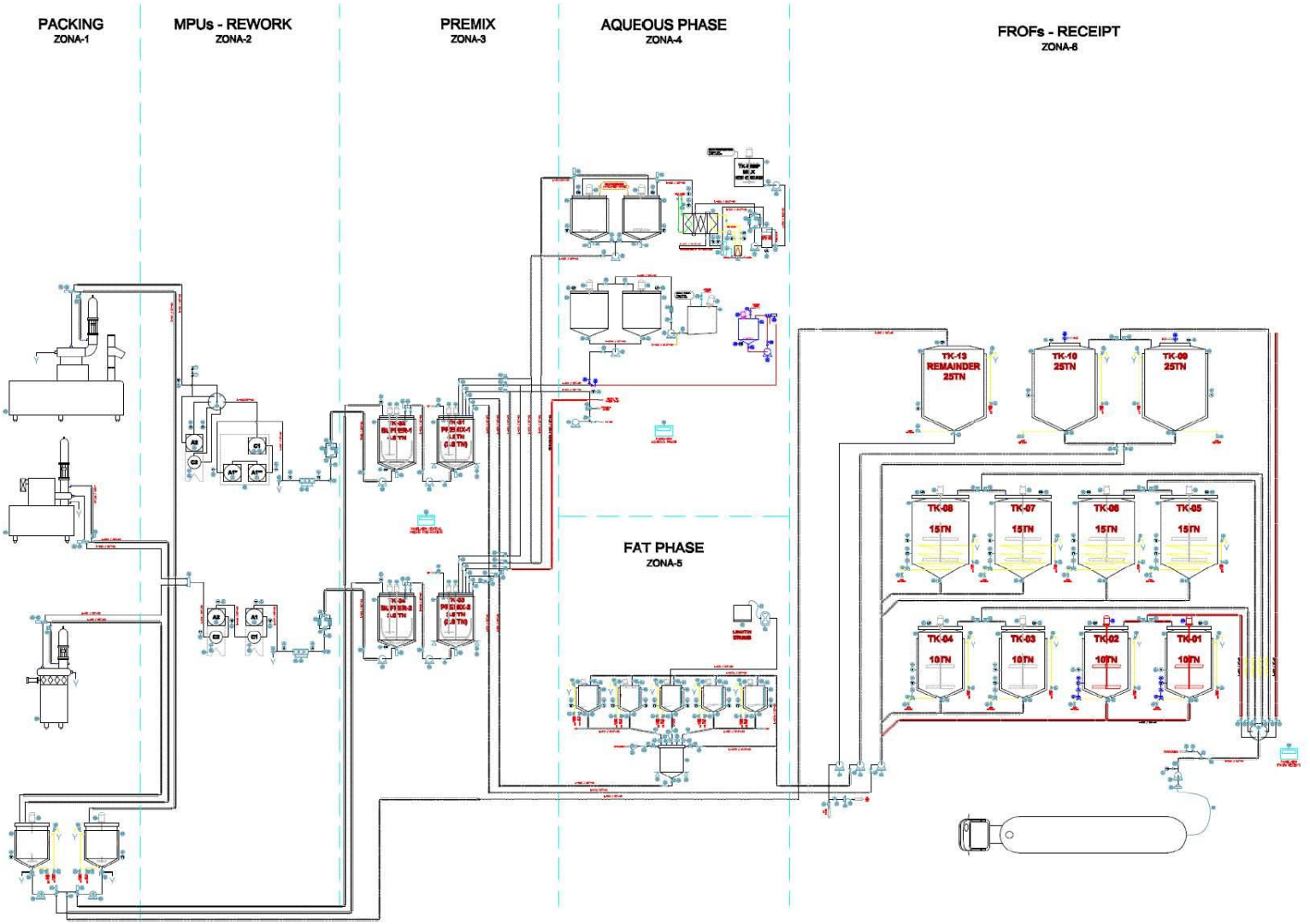
L(29.4,25.5), 0, 0) // Cantidad de aceite faltante para finalizar el lote
Tank_Transfer(I2, Premix2, Cantidad* 0.091, 1+L(251,112), 0, 0)
Tank_Transfer(I3, Premix2, Cantidad* 0.077, 1+L(251,112), 0, 0)
Tank_Transfer(I4, Premix2, Cantidad* 0.1120, 1+L(248,144), 0, 0)
Tank_Transfer(I5, Premix2, Cantidad* 0.1050, 1+L(248,144), 0, 0)
Tank_Transfer(I6, Premix2, Cantidad* 0.15753, 309 + L(614,951), 0, 0)
wait 1 min
Tank_Inc(Premix2,Cantidad* 0.00006)
Tank_Dec(I7,Cantidad* 0.00006)
//Tank_Transfer(I7, Premix2, Cantidad* 0.00006, N(0.305,0.152), 0, 0)
Tank_Transfer(I8, Premix2, Cantidad* 0.125 , 245+L(131,536), 0, 0)
Tank_Transfer(I9, Premix2, Cantidad* 0.01111,20.6+L(78.8,354) , 0, 0)
Tank_Transfer(I10, Premix2,Cantidad* 0.00509,1+L(1.63,0.644), 0, 0)
Tank_DoOperation(Premix2, 10)

// Operacion de mezcla
PreparacionTanqueB2_4 None Cantidad Integer Tank_Transfer(I1, Premix2, Cantidad * 0.6149, 97.7+
L(29.4,25.5), 0, 0) // Cantidad de aceite faltante para finalizar el lote
Tank_Transfer(I2, Premix2, Cantidad* 0, 1+L(251,112), 0, 0)
Tank_Transfer(I3, Premix2, Cantidad* 0, 1+L(251,112), 0, 0)
Tank_Transfer(I4, Premix2, Cantidad* 0, 1+L(248,144), 0, 0)
Tank_Transfer(I5, Premix2, Cantidad* 0.063, 1+L(248,144), 0, 0)
Tank_Transfer(I6, Premix2, Cantidad* 0.1895, 309 + L(614,951), 0, 0)
wait 1 min
Tank_Inc(Premix2,Cantidad* 0.0234)
Tank_Dec(I7,Cantidad* 0.0234)
//Tank_Transfer(I7, Premix2, Cantidad* 0.0234, N(0.305,0.152), 0, 0) // BHT
demora 1 min
Tank_Transfer(I8, Premix2, Cantidad* 0.105, 245+L(131,536), 0, 0)
Tank_Transfer(I9, Premix2, Cantidad* 0, 20.6+L(78.8,354) , 0, 0)
Tank_Transfer(I10, Premix2,Cantidad* 0.0042,1+L(1.63,0.644), 0, 0)
Tank_DoOperation(Premix2, 10)

****
* External Files *
****

ID	Type	File Name	Prompt
File1	General Read	(null)	

1.1 PLANOS



1.2

Plano Instalaciones de la Fábrica de Margarinas

BIBLIOGRAFÍA

[1] BANKS, J., CARSON, J., NELSON, B., NICOL, D., *Discrete Event System Simulation*, Prentice Hall, New Jersey, Estados Unidos, 2001.
[2] PROMODEL CORPORATION, ProModel User Guide, Versión 7, 2006.