

19



OFICINA ESPAÑOLA DE
PATENTES Y MARCAS

ESPAÑA



11 Número de publicación: **2 835 575**

51 Int. Cl.:

G06F 9/48 (2006.01)

G06F 11/07 (2006.01)

G06F 11/30 (2006.01)

12

TRADUCCIÓN DE PATENTE EUROPEA

T3

86 Fecha de presentación y número de la solicitud internacional: **16.03.2015** **PCT/GB2015/000091**

87 Fecha y número de publicación internacional: **19.11.2015** **WO15173533**

96 Fecha de presentación y número de la solicitud europea: **16.03.2015** **E 15716106 (8)**

97 Fecha y número de publicación de la concesión europea: **04.11.2020** **EP 3039543**

54 Título: **Una unidad de monitorización, así como un método para predecir el funcionamiento anormal de sistemas de ordenador activados por tiempo**

30 Prioridad:

11.05.2014 GB 201408285

11.05.2014 GB 201408286

45 Fecha de publicación y mención en BOPI de la traducción de la patente:

22.06.2021

73 Titular/es:

SAFETTY SYSTEMS LTD (100.0%)

15 Nether End

Great Dalby LE14 2EY, GB

72 Inventor/es:

PONT, MICHAEL JOSEPH

74 Agente/Representante:

SÁEZ MAESO, Ana

ES 2 835 575 T3

Aviso: En el plazo de nueve meses a contar desde la fecha de publicación en el Boletín Europeo de Patentes, de la mención de concesión de la patente europea, cualquier persona podrá oponerse ante la Oficina Europea de Patentes a la patente concedida. La oposición deberá formularse por escrito y estar motivada; sólo se considerará como formulada una vez que se haya realizado el pago de la tasa de oposición (art. 99.1 del Convenio sobre Concesión de Patentes Europeas).

DESCRIPCIÓN

Una unidad de monitorización, así como un método para predecir el funcionamiento anormal de sistemas de ordenador activados por tiempo

Campo de la invención

La presente invención se refiere a un aparato y un método que proporciona fiabilidad y seguridad mejoradas para sistemas informáticos. En particular, la presente invención se refiere a una unidad de monitorización para predecir el funcionamiento anormal de un sistema informático activado por tiempo, y un método para proporcionar dicha unidad de monitorización.

Antecedentes de la invención

Un sistema informático generalmente se compone de uno o más procesadores "listos para comercialización" (COTS), por ejemplo, microcontroladores o microprocesadores, y algún software que se ejecutará en dicho procesador(es): este software puede crearse, por ejemplo, usando un lenguaje de programación como "C" o Ada.

En muchos casos, los procesadores están "integrados" dentro de sistemas más grandes, incluidos automóviles, aeronaves, maquinaria industrial y agricultura, equipamiento médico, línea blanca y marrón e incluso en juguetes. Se estima que la gente en el mundo desarrollado se encuentra con alrededor de 300 de estos "sistemas integrados" todos los días mientras realiza sus actividades normales.

Otros usos relacionados de los sistemas informáticos incluyen aplicaciones de "escritorio" en tiempo real, tales como control del tráfico aéreo y gestión del tráfico.

Al crear dichos sistemas informáticos, los desarrolladores deben elegir una arquitectura de sistema adecuada. Una de esas arquitecturas es una arquitectura "activada por tiempo" (TT). La implementación de una arquitectura TT generalmente implicará el uso de una única interrupción que está vinculada al desbordamiento periódico de un temporizador. Esta interrupción puede impulsar un programador de tareas (una forma simple de "sistema operativo"). El programador, a su vez, comenzará la ejecución de las tareas del sistema (un proceso a veces llamado "liberar" las tareas, "activar" las tareas o "ejecutar" las tareas) en puntos predeterminados en el tiempo. Las tareas en sí mismas suelen denominarse bloques de código de programa que realizan una actividad en particular (por ejemplo, una tarea puede verificar si se ha presionado un interruptor): las tareas a menudo se implementan como funciones en lenguajes de programación como "C".

Este tipo de diseño TT puede ofrecer un comportamiento del sistema muy predecible, lo que hace que sea comparativamente fácil probar y verificar el correcto funcionamiento de los sistemas informáticos en tiempo real que se basan en dicha arquitectura: esta es la razón principal por la que los diseños TT son a menudo utilizados en sistemas críticos para la seguridad, sistemas de alta integridad y en otros productos donde la confiabilidad y/o seguridad del sistema son consideraciones de diseño importantes.

Independientemente de la arquitectura del sistema que se emplee, a menudo es importante intentar asegurarse de que el sistema informático funcione correctamente en caso de que se produzcan errores accidentales (como fallos de hardware y errores de software que puedan ser causados por interferencias electromagnéticas (EMI) o radiación relacionados con errores) o errores maliciosos (por ejemplo, los que pueden ser causados por intentos deliberados de efectuar cambios de comportamiento en el procesador o en el software, incluido, por ejemplo, el uso de "virus" informáticos). También se acepta generalmente que, a medida que los sistemas informáticos se hacen más grandes, es inevitable que habrá algunos errores residuales en el software en muchos sistemas que se han lanzado al campo: es claramente deseable que el sistema no funcione peligrosamente en el caso de tales errores residuales.

Se puede sentir que el uso de un procesador "redundante" (ver, por ejemplo, la Referencia 1) ofrece una solución a los problemas identificados anteriormente. Los diseños redundantes generalmente involucran dos (a veces más) procesadores que ejecutan el mismo código de programa en dos unidades diferentes de procesador, con varias formas de verificación cruzada (para detectar errores). En la mayoría de los casos (pero no en todos), la solución redundante involucrará dos o más unidades procesadoras muy próximas: esto genera preocupaciones sobre fallas de modo común: por ejemplo, el daño físico, vibración, humedad o fuentes de radiación pueden tener un impacto similar en todas las unidades de procesador, lo que hace que los errores no se detecten. Si podemos evitar tales fallas de modo común, los procesadores redundantes pueden permitirnos detectar el impacto de algunos errores (incluido, potencialmente, el impacto de EMI), pero brindan poca o ninguna protección contra errores de programación (incluidos errores residuales), o cambios en el programa (causados, por ejemplo, por virus o manipulación deliberada).

Además del uso de procesadores redundantes, también se han descrito previamente otras técnicas para detectar errores del tipo descrito anteriormente. Estos incluyen lo que a veces se denomina técnicas de "monitorización de flujo de programa" (PFM) (véase, por ejemplo, la Referencia 2). PFM generalmente implica técnicas basadas en hardware o software para detectar desviaciones del comportamiento normal del sistema (por ejemplo, monitorizando la actividad

del bus y/o los accesos a la memoria y buscando cambios en la firma "normal"). Otra forma bien conocida en la que se puede detectar el impacto de los errores anteriores en los sistemas informáticos es mediante el uso de mecanismos tales como temporizadores de "vigilancia" (véase, por ejemplo, la Referencia 3). Tales vigilancias generalmente imponen la detención o reinicio del sistema si no se "alimentan" con la frecuencia requerida (y se asume, a menudo implícitamente, que los errores del sistema interferirán con este proceso de alimentación).

En general, mientras que el uso de arquitecturas de procesador redundante y/o PFM y/o un temporizador de vigilancia pueden ayudar a mejorar la confiabilidad de un sistema dado, tales enfoques, en general, solo conferirán la capacidad de responder a cambios en el comportamiento de los sistemas que resultan de errores. Para el momento en que tales cambios sean lo suficientemente significativos como para ser detectados, es posible que ya se haya hecho una gran cantidad de daño. El documento WO 2009/113034A1 divulga técnicas para la gestión de tareas en procesadores de múltiples núcleos. El documento US 2008/215927 se refiere a ordenadores que ejecutan en modo de tiempo compartido, una serie de programas de aplicación separados e independientes.

Por ejemplo, consideremos que hemos detectado un error en el sistema de frenado de un automóvil de pasajeros: si el conductor ya está aplicando los frenos en una situación de emergencia cuando detectamos el error, el mecanismo de detección de errores tiene poco valor. La detección tardía de errores en los sistemas aeroespaciales, los sistemas industriales, los sistemas de defensa, los sistemas médicos, los sistemas financieros o incluso los enseres domésticos también puede resultar en lesiones, pérdida de vidas humanas y/o pérdidas económicas muy importantes.

Los documentos EP2528283 y WO2007054877 están relacionados con la programación de paquetes de red de acuerdo con una programación predeterminada donde un guardián monitoriza el orden de programación y si un paquete está programado para transmitirse fuera de este orden, el guardián evita esta transmisión de antemano.

Por lo tanto, un objeto de las realizaciones de la presente invención es mejorar la fiabilidad y la seguridad de los sistemas informáticos activados por tiempo detectando errores antes de que tengan consecuencias graves.

Resumen de la invención

De acuerdo con un primer aspecto de la presente invención, se proporciona un sistema informático activado por tiempo que comprende:

un procesador principal que comprende un programador activado por tiempo adaptado para ejecutar una o más tareas de acuerdo con uno o más programas de tareas predeterminados, que se determinan, validan y verifican cuando se diseña el sistema, y solo uno de ellos, un programa de tareas activo, estar activo en cualquier momento; y

un procesador monitor adaptado para determinar de antemano si el procesador principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activo utilizando una representación de secuencia de tareas que comprende un registro de una secuencia de ejecuciones de tareas que se espera que siga el procesador principal en el programa de tareas activo; y un enlace de comunicación para pasar información sobre futuras ejecuciones de tareas entre el procesador principal y el procesador monitor; y

un mecanismo de control mediante el cual el procesador monitor puede detener o restablecer el procesador principal y apagar, deshabilitar, habilitar o configurar los dispositivos conectados que están bajo el control del procesador principal, si el procesador monitor determina que el procesador principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activo.

De acuerdo con este aspecto de la invención, se proporciona un sistema informático que ejecuta tareas programadas con una probabilidad reducida de un fallo crítico.

El sistema informático está dispuesto para ejecutar tareas de acuerdo con uno o más programas de tareas predeterminados. Los programas de tareas determinan el orden en que se ejecutan las tareas y especifican si se permite o no que determinadas tareas interrumpan otras tareas. Las tareas se ejecutan en el procesador principal.

Normalmente, durante el funcionamiento, solo hay una programación de tareas predeterminada activa en un momento determinado (es decir, la "programación de tareas activa").

Ocasionalmente, debido a errores de programación, defectos de hardware u otros eventos imprevistos, los sistemas informáticos pueden ejecutar la tarea incorrecta durante la ejecución de un programa. En ciertas aplicaciones, esto puede resultar en una falla crítica del sistema ya que el procesador principal deja de funcionar de manera predecible o deja de funcionar por completo. Dependiendo de la aplicación del sistema informático, esto podría dar lugar a consecuencias altamente indeseables, como peligro para la vida (por ejemplo, en los sistemas de control de la aviación) o exposición a pérdidas financieras (por ejemplo, en aplicaciones comerciales financieras).

A diferencia de los sistemas convencionales, de acuerdo con este aspecto de la invención, el sistema informático está provisto de un Procesador monitor.

El procesador monitor está dispuesto para obtener información del procesador principal sobre al menos algunas de las tareas que el procesador principal planea ejecutar antes de que se ejecuten las tareas en cuestión. Esto permite que el procesador monitor determine de antemano si el procesador principal está a punto de ejecutar una tarea crítica que no está de acuerdo con el programa de tareas activo. Esto le brinda al Procesador monitor la oportunidad de tomar la acción correctiva apropiada, en virtud de un Mecanismo de Control, antes de que el Procesador Principal ejecute una tarea que no está de acuerdo con el programa de tareas activo. Dicha acción correctiva puede incluir, por ejemplo, detener el Procesador Principal: cuando el procesador se detiene, esto significa que dejará de funcionar y permanecerá en un estado fijo, normalmente hasta que se produzca una intervención externa (por ejemplo, la intervención de un operador capacitado). Alternativamente, dicha acción correctiva puede incluir, por ejemplo, restablecer el procesador principal: cuando el procesador principal se restablece, esto significa que comenzará sus operaciones nuevamente, desde un estado inicial conocido. Además, como alternativa, dicha acción correctiva puede incluir la desactivación de dispositivos que están bajo el control del Procesador principal: por ejemplo, esto podría significar quitar la energía de un dispositivo médico que está siendo controlado por el Procesador principal, evitando así daños a un paciente.

El Procesador Principal y el Procesador monitor pueden comprender cada uno, uno o más núcleos de procesador "blandos" o "duros" (que ejecutan algún software) y/o uno o más núcleos de hardware (que no ejecutan ningún software).

Opcionalmente, el Procesador Principal y el Procesador monitor se implementarán por medio de un solo chip de procesador, como un microcontrolador o listos para comercialización (COTS), Procesador de Señal Digital (DSP), Matriz de puerta programable de campo (FPGA), circuito integrado de aplicación específica (ASIC) o dispositivo similar.

Preferiblemente, el procesador principal y el procesador monitor se implementarán por medio de chips de procesador separados.

Preferiblemente, el procesador principal y el procesador monitor estarán separados físicamente.

Preferiblemente, el procesador principal comprenderá un programador activado por tiempo.

Preferiblemente, el procesador monitor tendrá acceso a al menos una representación de Secuencia de Tareas predeterminadas, habiendo sido determinada y almacenada esta Representación de Secuencia de Tareas en un momento antes de que el sistema informático comience a operar.

Opcionalmente, el procesador monitor creará una o más representaciones de secuencia de tareas durante la operación del sistema.

Opcionalmente, el procesador monitor creará una o más representaciones de secuencia de tareas durante la operación del sistema registrando la secuencia de tareas desde el procesador principal en puntos durante la operación del sistema.

Opcionalmente, cada representación de secuencia de tareas se almacenará en uno o más dispositivos de almacenamiento de datos.

Opcionalmente, el procesador monitor no procesará información sobre la siguiente tarea que se ejecutará para cada tarea en el programa de tareas activo.

Opcionalmente, el mecanismo de control comprenderá una salida de control del sistema y/o un enlace de reinicio, y/o un enlace de comunicación B.

Opcionalmente, el Procesador monitor, por medio del Mecanismo de Control, hará que el Procesador Principal se detenga o restablezca, en situaciones en las que el procesador monitor determina que el Procesador principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activo.

Opcionalmente, el proceso de detener el Procesador Principal en situaciones en las que el Procesador Monitor determina que el Procesador Principal está a punto de ejecutar una tarea que no está de acuerdo con la programación de la tarea activa, comprenderá además un intento de colocar el Procesador Principal en un estado seguro antes de detener el procesador principal.

Opcionalmente, el Procesador monitor señalará una alarma si determina que el Procesador principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activo.

También se divulga en este documento, pero no se reivindica, un sistema informático activado por tiempo que comprende un procesador para ejecutar:

un proceso principal adaptado para ejecutar una o más tareas de acuerdo con uno o más programas de tareas predeterminados, de los cuales solo uno, un programa de tareas activo, estará activo en cualquier momento; y un proceso monitor adaptado para determinar si el procesador principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activo; y un enlace de comunicación para pasar información sobre futuras ejecuciones de tareas entre el proceso principal y el proceso monitor; y un Mecanismo de Control por medio del cual el Proceso monitor puede detener o reiniciar el Proceso Principal y tomar otras acciones correctivas que involucren dispositivos a los cuales el sistema informático está conectado, si el Proceso monitor determina que el Proceso Principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activo.

También se divulga en el presente documento, pero no se reivindica, un método para controlar un sistema informático activado por tiempo que se ha adaptado para ejecutar una o más tareas de acuerdo con uno o más programas de tareas predeterminados, solo uno de los cuales, un programa de tareas activo, estará activo en cualquier momento, el método comprende los pasos de:

determinar si una tarea está a punto de ejecutarse que no está de acuerdo con el programa de tareas activo utilizando una Representación de secuencia de tareas que comprende un registro de una secuencia de ejecuciones de tareas que se espera seguir en el programa de tareas activo; y controlar la ejecución de una o más tareas que comprenden detener, reiniciar o reemplazar la una o más tareas, si se determina que está a punto de ejecutarse una tarea que no está de acuerdo con el programa de tareas activo; y controlar el funcionamiento de uno o más dispositivos a los que está conectado el sistema informático, si se determina que está a punto de ejecutarse una tarea que no está de acuerdo con el programa de tareas activo en el que se controla el funcionamiento de uno o más dispositivos comprende apagar, deshabilitar, habilitar o configurar uno o más dispositivos.

También se divulga en este documento, pero no se reivindica, un producto de programa informático que contiene una o más secuencias de instrucciones legibles por máquina, estando adaptadas las instrucciones para hacer que uno o más procesadores proporcionen un método que comprende los pasos de:

determinar si una tarea está a punto de ejecutarse que no está de acuerdo con el programa de tareas activo utilizando una Representación de secuencia de tareas que comprende un registro de una secuencia de ejecuciones de tareas que se espera seguir en el programa de tareas activo; y

controlar la ejecución de una o más tareas que comprenden detener, reiniciar o reemplazar la una o más tareas, si se determina que está a punto de ejecutarse una tarea que no está de acuerdo con el programa de tareas activo; y controlar el funcionamiento de uno o más dispositivos a los que está conectado el sistema informático, si se determina que está a punto de ejecutarse una tarea que no está de acuerdo con el programa de tareas activo en el que se controla el funcionamiento de uno o más dispositivos comprende apagar, deshabilitar, habilitar o configurar el uno o más dispositivos.

Breve descripción de las Figuras

La presente invención se describirá ahora solo a modo de ejemplo y con referencia a las Figuras adjuntas en las que:

La Figura 1 ilustra en forma esquemática una secuencia de ejecuciones de tareas en una forma que podría ser generada (por ejemplo) por un programador cooperativo activado por tiempo (TTC) que se ejecuta en un procesador principal;

La Figura 2 ilustra en forma esquemática una secuencia alternativa de ejecuciones de tareas en una forma que podría ser generada (por ejemplo) por un programador TTC que se ejecuta en un Procesador Principal en el que los pulsos del sistema no son periódicos;

La Figura 3 ilustra en forma esquemática una secuencia de ejecuciones de tareas en una forma que podría ser generada (por ejemplo) por un programador híbrido activado por tiempo (TTH) que se ejecuta en un procesador principal;

La Figura 4 ilustra en forma esquemática el funcionamiento de un sistema en el que el procesador principal ejecuta diferentes conjuntos de tareas en diferentes modos de sistema;

La Figura 5 ilustra en forma esquemática una posible estructura de código para un programador TTC que podría emplearse en un procesador principal;

La Figura 6 ilustra en forma esquemática el funcionamiento de una función de despachador de tareas típica de un programador TTC o TTH;

La Figura 7 ilustra en forma esquemática una posible estructura de código para un programador TTH que podría emplearse en un procesador principal;

La Figura 8 ilustra en forma esquemática una realización de un sistema informático activado por tiempo que comprende un Procesador principal y un Procesador monitor, con un Enlace de comunicación asociado (a través del cual el Procesador principal puede enviar información sobre ejecuciones de tareas futuras al Procesador Monitor), una representación interna y dos externas de la representación de secuencia de tareas activa (registrando la secuencia de ejecuciones de tareas que se espera que siga el procesador principal en el modo de sistema activo), una salida de control del sistema (para controlar dispositivos externos y/o el sistema informático mismo), un enlace de reinicio (mediante el cual el procesador monitor puede detener o reiniciar el procesador principal), un enlace de comunicación B (mediante el cual el procesador monitor puede enviar mensajes al procesador principal), un dispositivo de señal de error externo (para señalar errores), un dispositivo de registro de errores externo (para registrar errores), de acuerdo con un aspecto de la presente invención;

La Figura 9 ilustra en forma esquemática una realización de un sistema informático activado por tiempo (que comprende un procesador principal y un procesador monitor) que se está utilizando para controlar un robot industrial, de acuerdo con un aspecto de la presente invención;

La Figura 10 ilustra en forma esquemática el funcionamiento de una función de despachador de tareas típica de un programador TTC o TTH, que se ha modificado de acuerdo con un aspecto de la presente invención;

La Figura 11 ilustra en forma esquemática el funcionamiento del software que podría ejecutarse en el procesador monitor ilustrado en la Figura 8, de acuerdo con un aspecto de la presente invención;

La Figura 12 ilustra en forma esquemática el funcionamiento de una función de despachador de tareas desde un programador TTC o TTH en el que el Procesador Monitor está enviando información de regreso al Procesador Principal avisando si la ejecución de la tarea que está a punto de realizar está de acuerdo con el programa de tareas predeterminado, de acuerdo con un aspecto de la presente invención;

La Figura 13 ilustra en forma esquemática una realización alternativa adicional de un sistema informático en el que el procesador monitor no almacena una representación del programa de tareas predeterminado, sino que, en cambio, calcula este programa en tiempo de ejecución, de acuerdo con un aspecto de la presente invención;

La Figura 14 ilustra en forma esquemática el funcionamiento del software que podría ejecutarse en el procesador monitor ilustrado en la Figura 13, de acuerdo con un aspecto de la presente invención;

La Figura 15 ilustra en forma esquemática otra realización alternativa de un sistema informático en el que el proceso principal y el proceso monitor se ejecutan en un solo procesador y el proceso monitor emplea un área de memoria interna para almacenar información sobre la representación de secuencia de tareas predeterminada, de acuerdo con un aspecto de la presente invención; y

La Figura 16 ilustra de forma esquemática el funcionamiento del software que podría ejecutarse en el proceso monitor ilustrado en la Figura 15, de acuerdo con un aspecto de la presente invención.

Descripción detallada de la invención

Comenzamos describiendo algunas técnicas existentes que se emplean para crear sistemas informáticos que ejecutan tareas de acuerdo con un programa de tareas predeterminado. A continuación, describimos cómo se puede utilizar la presente invención para mejorar la fiabilidad y seguridad de tales sistemas informáticos.

Los sistemas informáticos que ejecutan tareas de acuerdo con un programa de tareas predeterminado se denominan a veces sistemas activados por tiempo (TT). Los sistemas TT se implementan típicamente (pero no necesariamente) utilizando un diseño basado en una única interrupción vinculada al desbordamiento periódico de un temporizador.

Por ejemplo, la Figura 1 muestra un sistema 100 informático TT compuesto por un procesador 101 principal que está ejecutando un conjunto de tareas de software (en este caso, Tarea A 102, Tarea B 103, Tarea C 104 y Tarea D 105) de acuerdo con un programa de tareas predeterminado. En la Figura 1, la liberación de cada subgrupo de tareas (por ejemplo, Tarea A 102 y Tarea B 103) se activa por lo que generalmente se llama un "pulso de temporizador" 106. El pulso de temporizador 106 generalmente se implementa mediante una interrupción del temporizador.

En la Figura 1, los pulsos de temporizador son periódicos. En una aplicación aeroespacial, se puede utilizar el "intervalo de pulso" (es decir, el intervalo de tiempo entre pulsos del temporizador) de 25 ms, pero en muchos sistemas también se utilizan intervalos de pulsos más cortos (por ejemplo, 1 ms o menos).

En la Figura 1, la secuencia de tareas ejecutada por el sistema informático es la siguiente: Tarea A, Tarea C, Tarea B, Tarea D.

5 En muchos diseños, dicha secuencia de tareas se determinará en el momento del diseño (para cumplir con los requisitos del sistema) y se repetirá “para siempre” cuando el sistema se ejecute (hasta que ocurra un error, o el sistema se detenga o apague).

10 A veces es útil (por ejemplo, durante el proceso de diseño) pensar en esta secuencia de tareas como una “lista de pulsos”: dicha lista establece qué tarea se ejecutarán en cada “pulso” del sistema y el orden en que estas ejecuciones ocurrirán.

Por ejemplo, la lista de pulsos correspondiente al conjunto de tareas que se muestra en la Figura 1 podría representarse de la siguiente manera:

15 [Pulso 0]
Tarea A
Tarea C
[Pulso 1]
Tarea B
20 Tarea D

Una vez que el sistema llega al final de la lista de pulsos, comienza de nuevo desde el principio.

25 Debe quedar claro que los pulsos de temporizador no necesitan ser periódicos en tales sistemas. Por ejemplo, la Figura 2 muestra una representación esquemática 200 de un conjunto de tareas (incluida la Tarea P 201) en el que los pulsos de temporizador 202 se han adaptado para coincidir con los tiempos de ejecución de las diversas tareas. Estos sistemas también se pueden representar mediante una lista de pulsos: por ejemplo, una posible lista de pulsos correspondiente al conjunto de tareas que se muestra en la Figura 2 podría ser la siguiente:

30 [Pulso 0, t = 0 ms]
Tarea P
[Pulso 1, t = 3 ms]
Tarea Q
[Pulso 2, t = 11 ms]
35 Tarea R
[Pulso 3, t = 17 ms]
Tarea S
[Pulso 4, t = 19 ms]
Tarea T
40

En la Figura 1 y la Figura 2, las tareas son de naturaleza cooperativa (o “no preventiva”): cada tarea debe completarse antes de que se pueda ejecutar otra tarea. El diseño que se muestra en estas Figuras puede describirse como “cooperativa que se activa por tiempo” (TTC) por naturaleza.

45 Nótese que la implementación de diseños TTC no necesita implicar el uso de interrupciones en absoluto. Por ejemplo, podríamos ejecutar el conjunto de tareas que se muestran en la Figura 1 (102, 103, 104, 105) de la siguiente manera:

50 Tarea A
Tarea C
[Retraso]
Tarea B
Tarea D
[Retraso]

55 De nuevo, la secuencia anterior se repetiría “para siempre”. Este tipo de solución a veces se denomina “Super Bucle”. En general, el control de la sincronización de la tarea es menos preciso en los sistemas Super Bucle que en los que involucran interrupciones, pero para diseños TTC con lo que a veces se llama restricciones de tiempo “suaves”, el uso de un Super Bucle puede ser apropiado.

60 Otros diseños de TT populares implican el uso de la prevención de tarea, donde una tarea de alta prioridad puede prevenir (anticipar) una tarea de menor prioridad mientras se está ejecutando (es decir, la tarea de menor prioridad). Por ejemplo, el diagrama 301 superior en la Figura 3 muestra un conjunto de tres tareas 301, Tarea A 303 (una tarea cooperativa de baja prioridad), Tarea B 304 (otra tarea cooperativa de baja prioridad) y Tarea P (una tarea preventiva de prioridad más alta). En este ejemplo 301 302, las tareas de menor prioridad pueden ser prevenidas periódicamente por la tarea 302 de mayor prioridad. De manera más general, este tipo de diseño de “híbrido activado por tiempo”

65

(TTH) puede implicar múltiples tareas cooperativas (todas con igual prioridad) y una o más tareas preventivas (de mayor prioridad).

La implementación de diseños TTH generalmente implicará el uso de interrupciones.

La secuencia de tareas para el conjunto de tareas que se muestra en el diagrama 301 superior en la Figura 3 podría enumerarse de la siguiente manera: Tarea P, Tarea A, Tarea P, Tarea B, Tarea P, Tarea A, Tarea P, Tarea B.

Para cualquier conjunto de tareas periódicas, el ciclo de ejecuciones de tareas siempre se repetirá, después de un intervalo conocido como hiperperíodo (o "ciclo mayor"). La duración del hiperperíodo se puede determinar calculando el "mínimo común múltiplo" (LCM) de los periodos de todas las tareas del conjunto.

Por ejemplo, supongamos que tenemos un conjunto de dos tareas, con periodos de 4 ms y 5 ms.

Los múltiplos de 4 son: 4, 8, 12, 15, 20, 24 ...

Los múltiplos de 5 son: 5, 10, 15, 20, 25 ...

El mínimo común múltiplo (y la longitud del hiperperíodo) es 20 ms.

Cuando un conjunto de tareas periódicas es asincrónico, existe un retraso (o compensación) antes de que una o más tareas se ejecuten por primera vez. Este es el caso, por ejemplo, en el conjunto de tareas mostrado 100 en la Figura 1. En este caso, la Tarea A 102 y la Tarea B 103 se ejecutan con una compensación de 0, pero la Tarea C 104 y la Tarea D 105 se ejecutan después de un retraso de 1 Pulso. Tenga en cuenta que las cuatro tareas tienen el mismo período (2 pulsos) en este ejemplo.

Cuando se ejecuta un conjunto de tareas periódicas asincrónicas, puede haber un período inicial durante el cual la secuencia de tareas no es la misma que la secuencia de tareas cuando el sistema está en un "estado estable". Nos referimos a este período inicial como el período de inicialización de la secuencia de tareas.

La referencia 4 demuestra que, si es necesario, la duración de este período de inicialización de la secuencia de tareas puede calcularse mediante: (compensación de tarea más largo) + 2 * (duración del hiperperíodo).

Por ejemplo, para el siguiente conjunto de tareas:

Tarea_A: Periodo = 5 ms, Compensación = 0 ms
Tarea_B: Periodo = 10 ms, Compensación = 1 ms
Tarea_C: Periodo = 25 ms, Compensación = 2 ms
Tarea_D: Periodo = 25 ms, Compensación = 3 ms
Tarea_E: Periodo = 50 ms, Compensación = 4 ms

La duración del hiperperíodo es de 50 ms.

La duración del período de inicialización de la secuencia de tareas es de 104 ms.

Cuando un conjunto de tareas periódicas es sincrónico, todas las tareas se ejecutan por primera vez en el primer pulso.

Los diseños de TTC y TTH (así como otros diseños de TT) pueden emplear varios programas de tareas, para su uso en diferentes modos de sistema. Por ejemplo, la Figura 4 muestra una representación esquemática de la arquitectura 400 de software para un sistema de aeronave con diferentes Modos (401, 402, 403) de Sistema correspondientes a las diferentes etapas de vuelo (preparación para el despegue, ascenso a altura de crucero, etc.).

Al desarrollar diseños TT con múltiples modos de sistema, es probable que tengamos una lista de pulsos diferente para cada modo de sistema.

Tenga en cuenta que, durante el diseño del sistema, es posible que debamos considerar el período de inicialización de la secuencia de tareas tanto cuando se inicializa el sistema como en cualquier momento en el que cambia el modo del sistema.

Tenga en cuenta también que, para un diseño de sistema TT de este tipo, podemos determinar en tiempo de diseño con precisión cuál será la secuencia de tareas cuando el sistema funcione: esto significa que, si es necesario, podemos determinar la secuencia de tareas incluso durante el período de inicialización de la secuencia de tareas (para conjuntos de tareas asincrónicas). Una forma conveniente de hacer esto es tratar el período de inicialización de la secuencia de tareas como un modo de sistema separado.

Tenga en cuenta que el tiempo de transición entre los modos del sistema puede no conocerse de antemano (porque, por ejemplo, el tiempo que tarda el avión que se muestra en la Figura 4 en alcanzar la altura de crucero variará con las condiciones meteorológicas), pero esto no altera el proceso de diseño o implementación. La característica clave de todos los diseños de TT es que las tareas, ya sean cooperativas o preventivas, cualquiera que sea el modo del sistema, siempre se liberan de acuerdo con un programa que se determina, valida y verifica cuando se diseña el sistema.

Se apreciará que sólo un modo de sistema puede estar activo en cualquier momento mientras el sistema informático está en funcionamiento: esto se denomina modo de sistema activo. Cuando está en el modo de sistema activo, el sistema informático ejecutará tareas de acuerdo con el programa de tareas activo. En un sistema informático, las diversas programaciones de tareas para los diversos modos del sistema pueden registrarse en varias representaciones de secuencia de tareas: la representación de secuencia de tareas correspondiente al modo de sistema activo se conoce como la representación de secuencia de tareas activa (o TSR activo).

Ahora consideramos cómo se implementan típicamente los diseños TT en los sistemas actuales. Puede encontrar más información sobre este tema en la Referencia 5 y la Referencia 6.

La Figura 5 muestra una representación 500 esquemática de los componentes clave en un programador de TTC. Primero, está la función SCH_Update () 502: está vinculada, en este ejemplo, a un temporizador 503 que se supone que genera pulsos periódicos, es decir, el temporizador interrumpe 504 cada milisegundo. La función SCH_Update () 502 es responsable de realizar un seguimiento del tiempo transcurrido. Dentro de la función main () 501 asumimos que hay funciones para inicializar el programador, inicializar las tareas y luego agregar las tareas al programa.

Por ejemplo, el siguiente código ilustra un ejemplo de cómo se pueden configurar tres tareas periódicas en un Procesador Principal usando un programador TTC estándar:

```
int main(void)
{
    SCH_TTC_Init(); // Set up the scheduler
    // Other init functions
    // ...
    // Add Task_A, Task_B and Task_C to the schedule
    // Parameters: Task name, task offset, task period
    SCH_TTC_Add_Task(Task_A, 0, 10);
    SCH_TTC_Add_Task(Task_B, 0, 15);
    SCH_TTC_Add_Task(Task_C, 0, 33);
    SCH_TTC_Start(); // Start the schedule
    while(1)
    {
        SCH_TTC_Dispatch_Tasks();
    }
    return 1;
}
```

Tenga en cuenta que la lista de códigos anterior muestra un conjunto de tareas sincrónicas.

La siguiente lista de códigos muestra un conjunto de tareas asincrónicas:

```
int main(void)
{
    SCH_TTC_Init(); // Set up the scheduler
    // Other init functions
    // ...
    // Add Task_A, Task_B and Task_C to the schedule
    // Parameters: Task name, task offset, task period
```

```
SCH_TTC_Add_Task(Task_A, 0, 10);
SCH_TTC_Add_Task(Task_B, 3, 15);
SCH_TTC_Add_Task(Task_C, 10, 33);
SCH_TTC_Start(); // Start the schedule
while(1)
{
    SCH_TTC_Dispatch_Tasks();
}
return 1;
}
```

En ambos listados de códigos presentados anteriormente (y en la Figura 5, la función main () 501), el proceso de liberar las tareas del sistema se lleva a cabo en la función SCH_Dispatch_Tasks (). El funcionamiento de una función SCH_Dispatch () típica se ilustra esquemáticamente en la Figura 6.

La Figura 6 muestra una descripción general 600 de una función de despachador de tareas que podría incorporarse en un programador TTC o TTH. En la Figura 6, el despachador comienza determinando 601 602 si hay alguna tarea que deba ejecutarse actualmente. Si la respuesta a esta pregunta es "sí", el despachador ejecuta la tarea 603. Este repite este proceso (verifique 601, 602, ejecute 603) hasta que no queden tareas pendientes de ejecución. Luego, el despachador pone el procesador en suspensión 604: es decir, coloca el procesador en un modo de ahorro de energía. El procesador permanecerá en este modo de ahorro de energía hasta que lo despierte la siguiente interrupción del temporizador: en este punto, se llamará nuevamente al temporizador ISR 502 (Figura 5), seguido de la siguiente llamada a la función 600 de despachador (Figura 6).

Haciendo referencia de nuevo a la Figura 5, debe notarse que hay una división deliberada entre el proceso de actualizaciones de temporizador 502 y el proceso de despacho de tareas (en main () 501, y descrito con más detalle en la Figura 6). Esta división significa que es posible que el programador ejecute tareas que tengan más de un intervalo de pulso sin perder pulsos del temporizador. Esto brinda una mayor flexibilidad en el diseño del sistema, al permitir el uso de un intervalo de pulso corto (que puede hacer que el sistema responda mejor) y tareas más largas (que pueden, por ejemplo, simplificar el proceso de diseño). Aunque esta flexibilidad está disponible, muchos (pero no todos) los sistemas TTC están diseñados para garantizar que no se estén ejecutando tareas cuando se produce una interrupción del temporizador: sin embargo, incluso en tales diseños, un error de tiempo de ejecución puede significar que una tarea tarda más en completarse. Debido a la naturaleza dinámica del programador, el sistema puede recuperarse de tales errores en tiempo de ejecución, siempre que el error no sea permanente.

La flexibilidad en el proceso de diseño y la capacidad de recuperarse de errores son dos razones por las que los diseños TT "dinámicos" (con un temporizador ISR separado y funciones de despacho de tareas) son generalmente preferidos a los diseños más simples en los que las tareas se despachan desde el temporizador ISR. Además, la separación de las funciones de despacho de tareas y de ISR también simplifica la creación de diseños TT con soporte para la prevención de tareas (a veces llamados diseños "híbridos activados por tiempo" - TTH).

Como ejemplo de un diseño de programador de TTH, véase la Figura 7. Esta Figura muestra una vista esquemática 700 de una implementación típica de un programador de TTH. Al igual que con el diseño TTC (en la Figura 5), el diseño TTH incorpora una función SCH_Update () 702 y una función main () 701 con un despachador de tareas. El diseño de TTH también vincula un temporizador 703 que genera interrupciones 704 periódicas a la función SCH_Update 702. La única diferencia entre la implementación de TTC en la Figura 5 y la implementación de TTH en la Figura 6 es que la implementación de TTH incluye una llamada a una tarea de prevención. (función) 705 dentro del temporizador ISR.

Véase la Referencia 5 y la Referencia 6 para obtener más información sobre la implementación de diferentes formas de programador TT.

Ahora consideramos cómo se implementan las tareas en un diseño TT típico.

La referencia 5 y la referencia 6 proporcionan ejemplos no limitativos de los tipos de tareas que se pueden ejecutar en los sistemas TT, por ejemplo, "transmisión de datos RS-232", "actualizaciones de visualización" y tareas de "control PID", incluidos detalles de implementación completos. Otros ejemplos de tareas pueden incluir leer datos de entrada, realizar cálculos y generar salidas.

El uso de una arquitectura TT permite al desarrollador garantizar que algunas o todas las tareas del sistema comiencen a ejecutarse en momentos determinados con precisión. Sin embargo, en algunos sistemas en tiempo real, aún se debe tener cuidado al diseñar e implementar las tareas para garantizar que se cumplan todos los requisitos de sincronización del sistema.

Para comprender por qué esto es necesario, considere el siguiente ejemplo de una tarea que puede ejecutarse en el procesador principal:

```
void Task_A(void)
{
    /* Task_A has a known WCET of A milliseconds */
    // Read inputs
    // Perform calculations
    /* Starting at t <= A ms */
    // Generate outputs

    /* Task_A completes within A milliseconds */
}
```

En este caso, se conoce el tiempo de ejecución del peor caso (WCET) de la tarea. El conocimiento de WCET es un requisito estándar para las tareas en sistemas integrados en tiempo real. En este caso sabemos (solo) que la tarea generará ciertas salidas dentro de A ms desde el inicio de la tarea (donde A es el WCET conocido de la tarea).

El listado siguiente muestra una implementación alternativa de la tarea:

```
void Task_A(void)
{
    /* Task_A has a known WCET of A milliseconds */
    /* Task_A is balanced */
    // Read inputs (KNOWN AND FIXED DURATION)
    // Perform calculations (KNOWN AND FIXED DURATION)
    /* Starting at t = A1 ms
       (measured from the start of the task),
       for a period of A2 ms, generate outputs */
    /* Task_A completes after (A1+A2) milliseconds */
}
```

En esta implementación alternativa, el código de la tarea se ha equilibrado. Un poco más formalmente, esto significa que conocemos los tiempos de ejecución del “peor caso” (WCET) y “mejor caso” (BCET) de esta tarea, y que BCET = WCET.

Cuando el código está equilibrado, es posible determinar con mayor precisión cuándo se realizarán las actividades de la tarea en particular (en un momento o momentos medidos con respecto al inicio de la tarea). En el ejemplo que se muestra arriba, las salidas de la tarea se generarán en un intervalo que comienza A1 ms después del inicio de la tarea y termina A2 ms después.

Por lo tanto, puede ser deseable equilibrar el código de tarea. Las técnicas de equilibrio incluyen la programación de ruta única (véase la Referencia 7, la Referencia 8, la Referencia 9 y la Referencia 10) y los retrasos en sándwich (Referencia 11).

Ahora describimos cómo se puede implementar la presente invención con el fin de mejorar la confiabilidad y seguridad de los sistemas informáticos que se han configurado para ejecutar tareas de acuerdo con un programa de tareas predeterminado.

La Figura 8 muestra un sistema 800 informático que comprende un procesador 802 monitor que actúa para determinar dónde un procesador 801 principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activo. Para facilitar esto: [i] el procesador 801 principal proporciona información sobre la siguiente tarea que pretende ejecutar al procesador 802 monitor a través de un enlace 803 de comunicación; y [ii] el procesador 802 monitor tiene acceso a la representación 804 812 813 de la secuencia de tareas activa (es decir, un registro legible por ordenador de la programación de la tarea activa): esta representación 804 812 813 de la secuencia de tareas activa puede almacenarse en alguna forma de memoria 804 interna y/o en alguna forma de dispositivo 812 813 externo de almacenamiento de datos.

En la Figura 8, el procesador principal y el procesador monitor pueden comprender cada uno, uno o más núcleos de procesador “blandos” o “duros” (que ejecutan algún software) y/o uno o más núcleos de hardware (que no ejecutan ningún software). El procesador 801 principal y el procesador 802 monitor se pueden implementar en un solo chip de procesador, como un microcontrolador o microprocesador listos para comercialización (COTS), procesador de señal digital (DSP), matriz de puerta programable en campo (FPGA), circuito integrado de aplicación específica (ASIC) o dispositivo similar.

Por ejemplo, el sistema informático puede implementarse utilizando un procesador de doble núcleo, con el procesador principal funcionando en un núcleo y el procesador monitor funcionando en el segundo núcleo. Como ejemplo alternativo, el sistema informático puede implementarse en una plataforma FPGA, con el procesador principal implementado por medio de un núcleo de procesador “suave” o “duro” (y algún software) y el procesador monitor implementado usando lógica FPGA (sin ningún tipo de software).

Sin embargo, preferiblemente, el procesador principal y el procesador monitor se implementarán en chips de procesador separados, reduciendo así las oportunidades de fallas de modo común en el proceso de detección de errores.

También preferiblemente, el Procesador 801 principal y el procesador 802 monitor estarán físicamente separados (por ejemplo, el procesador 801 principal puede implementarse en un chip de procesador que está ubicado en el compartimiento del motor en un vehículo de pasajeros y el procesador 802 monitor puede implementarse en un chip de procesador que se encuentra entre los asientos en el área de pasajeros), reduciendo así (más) las oportunidades de fallas de modo común en el proceso de detección de errores.

Se apreciará que, independientemente de las opciones de diseño e implementación seleccionadas para el procesador 801 principal y el procesador 802 monitor en un sistema en particular,

asegurando que el procesador 801 y el procesador 802 monitor emplean diferentes diseños de hardware y/o software, también pueden reducir las oportunidades de fallas en modo común en el proceso de detección de errores.

La Figura 8 también muestra el mecanismo (805, 806, 807) de control mediante el cual el procesador 802 monitor puede detener o reiniciar el procesador 801 principal y tomar otras acciones correctivas, si el procesador 802 monitor determina que el procesador 801 principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activo: en breve hablamos más sobre el mecanismo de control.

Preferiblemente, el procesador 801 principal comprende un programador activado por tiempo.

Este programador activado por tiempo puede (como ejemplo no limitativo) ser impulsado por una fuente periódica de interrupciones. Alternativamente (como otro ejemplo no limitativo), este programador activado por tiempo puede ser impulsado por una fuente no periódica de interrupciones.

Opcionalmente, el procesador 801 principal comprende un programador cooperativo activado por tiempo (TTC) o un programador híbrido activado por tiempo (TTH), similar a los diseños descritos en detalle en la Referencia 5.

Opcionalmente, una o más de las tareas que se ejecutan en el procesador 801 principal se han “equilibrado” para garantizar que el tiempo de ejecución de la tarea sea aproximadamente el mismo cada vez que se ejecuta la tarea.

Preferiblemente, el procesador 802 monitor tiene acceso a al menos una Representación 804 812 813 de Secuencia de Tareas predeterminada almacenada - por ejemplo - en la memoria 804 interna o en un dispositivo 812 813 externo de almacenamiento de datos, habiendo sido determinada y almacenada esta representación antes de que el sistema informático comience a funcionar.

Como ejemplo no limitativo, se pueden usar diferentes tipos de memoria 804 interna para almacenar la representación 804 812 813 de secuencia de tareas: por ejemplo, memoria RAM, ROM, Flash o EEPROM. También se pueden emplear otras formas de implementar tal dispositivo de memoria interna (por ejemplo, usando una colección de puertas adecuadas en un dispositivo FPGA).

Como ejemplo de la implementación de una representación 804 812 813 de secuencia de tareas, por favor véase la Figura 1. Para el conjunto de tareas que se muestra en la Figura 1 (102, 103, 104, 105), una posible representación 804 812 813 de la secuencia de tareas sería la siguiente:

```
// Task Sequence Representation
#define LENGTH_OF_TSR_G 4
static unsigned char TSR_G[LENGTH_OF_TSR_G] =
{
    0,
    2,
    1,
    3
};
```

En el ejemplo anterior, la matriz almacena una representación de las tareas mostradas en la Figura 1, donde la Tarea A 102 se representa con el ID de tarea 0, la Tarea B 103 se representa con el ID de tarea 1, y así sucesivamente.

Opcionalmente, el procesador 802 monitor crea y almacena una o más representaciones 804 812 813 de secuencia de tareas en, por ejemplo, un 804 interno o un dispositivo 812, 813, de memoria externa, estas representaciones se determinan observando el funcionamiento del procesador 801 principal en puntos durante la operación del sistema.

Opcionalmente, la recopilación de información de la secuencia de tareas se llevará a cabo cuando el sistema informático ingrese a un modo de sistema en particular por primera vez.

Alternativamente, el procesador 802 monitor creará una o más representaciones 804 812 813 de secuencia de tareas durante la operación del sistema ejecutando un algoritmo de programación en el procesador 802 monitor que es similar al algoritmo de programación que se ejecuta en el procesador 801 principal.

A menudo habrá una representación 804 812 813 de secuencia de tarea que corresponde a cada modo de sistema diferente en el procesador 801 principal.

Cada representación 804 812 813 de secuencia de tarea puede almacenarse en uno o más dispositivos 804 812 813 de almacenamiento de datos.

Opcionalmente, habrá dos o más copias de una o más representaciones 804 812 813 de secuencia de tareas, estas copias se mantendrán en uno o más dispositivos 804 812 813 de almacenamiento de datos. Opcionalmente, las diferentes copias de la representación 804 812 813 de secuencia de tarea se comparará en puntos durante la operación del sistema y, si se encuentra diferente, este evento puede tratarse como un error de datos. Opcionalmente, dicho error de datos se manejará de la misma manera que se maneja un error de secuencia de tareas.

Opcionalmente, habrá tres o más copias de una o más representaciones 804 812 813 de secuencia de tareas: estas copias pueden guardarse en uno o más dispositivos 804 812 813 de almacenamiento de datos. Opcionalmente, las diferentes copias de la representación 804 812 813 de secuencia de tarea se comparará en puntos durante la operación del sistema y, si se encuentra diferente, una decisión sobre la secuencia correcta puede basarse en una decisión mayoritaria; cuando no sea posible una decisión mayoritaria, este evento puede tratarse como un error de datos. Opcionalmente, dicho error de datos se manejará de la misma manera que se maneja un error de secuencia de tareas.

Se apreciará que, independientemente de las opciones de diseño e implementación que se seleccionen para el procesador 801 principal y el procesador 802 monitor en un sistema particular, se asegura que el procesador 801 principal y el procesador 802 monitor empleen diferentes enfoques para determinar qué tarea debe ser ejecutada a continuación por el procesador 801 principal puede reducir las oportunidades de fallas de modo común en el proceso de detección de errores. Por ejemplo, una combinación efectiva puede ser para usar [i] un procesador 801 principal que comprende un programador (dinámico) TTC o TTH, por medio del cual determinará qué tarea ejecutar a continuación en base a cálculos realizados en tiempo de ejecución, y combinará esto con [ii] un procesador 802 monitor que comprende un conjunto de representaciones de secuencia de tareas estáticas (donde las representaciones de secuencia de tareas se determinaron en tiempo de diseño, antes de que el sistema comenzara a ejecutarse).

El enlace 803 de comunicación entre el procesador 801 principal y el procesador 802 monitor se puede implementar utilizando una conexión en serie cableada, basada en un protocolo estándar (por ejemplo, SPI, I2C, RS-232, RS-422, RS-485, CAN, etc.) o un protocolo personalizado. Alternativamente, el enlace 803 de comunicación entre el procesador 801 principal y el procesador 802 monitor puede implementarse usando una interfaz paralela cableada (con, por ejemplo, uno o más pines I/O de procesador usados para representar el estado actual del procesador principal). Alternativamente, el enlace 803 de comunicación entre el procesador 801 principal y el procesador 802 monitor se puede implementar usando un área de memoria a la que ambos procesadores tienen acceso. Alternativamente, el enlace 803 de comunicación entre el procesador 801 principal y el procesador 802 monitor se puede implementar usando un protocolo inalámbrico.

Opcionalmente, el procesador monitor, por medio del mecanismo de control, hará que el procesador principal se detenga o reinicie, en situaciones en las que el procesador monitor determina que el procesador principal está a punto de ejecutar una tarea que no está de acuerdo con el horario de tareas activo.

Opcionalmente, el proceso de detener el procesador principal en situaciones en las que el procesador monitor determina que el procesador principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de la tarea activa, comprenderá además un intento de colocar el procesador principal en un estado seguro antes de detener el procesador principal. Tal estado seguro puede ser uno en el que, por ejemplo, los pines de salida del procesador principal se establecen en niveles que garantizan que cualquier equipo externo que esté bajo el control del sistema informático esté apagado.

El mecanismo de control puede comprender una salida 805 de control del sistema y/o un enlace 807 de reinicio y/o un enlace de comunicación B 806.

Opcionalmente, como parte del mecanismo de control, el sistema informático tendrá una salida 805 de control del sistema que permite al sistema controlar uno o más dispositivos o sistemas externos y/o el propio sistema informático.

Opcionalmente, la salida 805 de control del sistema comprenderá uno o más pines de salida en el procesador monitor, y dichos pines pueden conectarse además a uno o más relés electromecánicos u otros accionadores o dispositivos similares.

Opcionalmente, la salida 805 de control del sistema permitirá que el sistema informático desconecte el procesador 801 principal de una fuente de alimentación, deteniendo así el procesador 801 principal (y haciéndolo inoperativo). Opcionalmente, la salida 805 de control del sistema permitirá que el sistema informático desconecte el procesador 801 principal de una fuente de alimentación temporalmente, "haciendo cíclica la energía" en el procesador 801 principal y provocando que se reinicie. Opcionalmente, la salida 805 de control del sistema se utilizará para desconectar el sistema informático de su propia fuente de alimentación. Opcionalmente, la salida 805 de control del sistema se utilizará para desconectar el sistema informático de otros sistemas. Opcionalmente, la salida 805 de control del sistema se utilizará para apagar los dispositivos que están bajo el control del sistema informático. Opcionalmente, la salida 805 de control del sistema permitirá que el sistema informático desconecte uno o más dispositivos externos de una fuente de alimentación, haciendo que tales dispositivos no funcionen. Opcionalmente, la salida 805 de control del sistema se utilizará para habilitar dispositivos que están bajo el control del sistema informático. Opcionalmente, la salida 805 de control del sistema se utilizará para habilitar sistemas de seguridad en dispositivos que están bajo el control del sistema informático.

Como ejemplo no limitativo, la Figura 9 ilustra el uso de la salida de control del sistema 905 en un diseño 900 en el que el procesador 901 principal se utiliza para controlar un robot 903 industrial. El sistema informático comprende un procesador 901 principal vinculado a un procesador 902 monitor separado físicamente por medio de un enlace 907 de comunicación que se implementa (en este ejemplo) por medio del protocolo estándar RS-422. En este diseño 900, la salida 905 de control del sistema se utiliza para apagar la fuente 904 de alimentación al robot 903 en caso de que se detecte un error de secuencia de tareas en el procesador 901 principal. De esta manera, se impide que el robot 903 funcione peligrosamente en caso de una falla en el procesador 901 principal.

Opcionalmente, como parte del mecanismo de control, habrá un enlace 807 de reinicio entre el Procesador 802 Monitor y el procesador 801 principal. Opcionalmente, este enlace 807 de reinicio se implementará como una conexión entre un pin de salida en el procesador 802 monitor y un pin de reinicio en el procesador principal. Alternativamente, este enlace 807 de reinicio puede implementarse como una conexión entre un pin de salida en el Procesador 802 monitor y un pin relacionado con interrupciones en el Procesador Principal. Alternativamente, este enlace 807 de reinicio puede implementarse como una conexión entre un pin de salida en el Procesador 802 Monitor y un pin de entrada en el Procesador Principal. Alternativamente, el enlace 807 de reinicio puede ser cualquier medio de activar un reinicio en el procesador principal.

Opcionalmente, como parte del Mecanismo de Control, habrá un Enlace de Comunicación B 806 entre el Procesador 802 Monitor y el Procesador Principal. Opcionalmente, el enlace de comunicación B 806 entre el Procesador 802 monitor y el procesador 801 principal puede implementarse utilizando un protocolo en serie cableado. Alternativamente, el enlace de comunicación B 806 entre el procesador 802 monitor y el procesador 801 principal puede implementarse usando un protocolo paralelo cableado. Alternativamente, el enlace de comunicación B 806 entre el procesador 802 monitor y el procesador 801 principal puede implementarse usando un área de memoria a la que ambos procesadores tienen acceso. Alternativamente, el enlace de comunicación B 806 entre el procesador 802 monitor y el procesador 801 principal puede implementarse de forma inalámbrica.

Opcionalmente, por medio del Mecanismo de Control, el Procesador 802 Monitor puede reiniciar el Procesador 801 principal, si este determina que el Procesador 801 principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activo.

Opcionalmente, por medio del Mecanismo de Control, el Procesador 802 Monitor puede detener o reiniciar el Procesador Principal, si este determina que el Procesador 801 principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activo.

Opcionalmente, por medio del Mecanismo de Control, el Procesador 802 Monitor pasará un mensaje al Procesador 801 principal solicitando que el Procesador 801 principal se reinicie o se detenga, si determina que el Procesador 801 principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activo. Se apreciará que tal solicitud está destinada a darle al Procesador 801 principal la oportunidad de realizar un "apagado ordenado", si esto es posible. También se apreciará que, si el Procesador 801 principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activa, entonces esto puede ser sintomático de problemas o fallas importantes en el Procesador Principal. Si, por lo tanto, el procesador 801 principal no cumple con el restablecimiento de la solicitud de detención anterior dentro de un período predeterminado, entonces, opcionalmente, por medio del mecanismo de control, el Procesador 802 Monitor hará, por ejemplo, un ciclo de energía al procesador 801 principal

(provocando que se reinicie) o desconecte la energía del procesador 801 principal por completo (provocando que se detenga).

5 Opcionalmente, por medio del Mecanismo de Control, el Procesador 802 Monitor puede reiniciar o detener el Procesador 802 Monitor si no recibe informes de actividad sobre una o más tareas del Procesador 801 principal durante un período predeterminado.

10 Opcionalmente, por medio del Mecanismo de Control, el Procesador 802 monitor puede reiniciar o detener el Procesador 801 principal si no recibe informes de ninguna actividad de tarea desde el Procesador 801 principal durante un período predeterminado.

Opcionalmente, el Procesador 802 Monitor puede ignorar los reportes de actividad de tareas del Procesador 801 principal en momentos particulares durante la operación del sistema.

15 Opcionalmente, el Procesador 802 Monitor está configurado para permitir el funcionamiento continuo del Procesador 801 principal siempre que el número de errores detectados esté por debajo de un umbral predeterminado.

20 Opcionalmente, el Procesador 802 Monitor está configurado para permitir el funcionamiento continuo del Procesador 801 principal siempre que la tasa de errores detectados esté por debajo de un umbral predeterminado.

Opcionalmente, el procesador 801 principal puede no enviar información sobre la siguiente tarea a ejecutar para cada tarea en el programa de tareas predeterminado.

25 Opcionalmente, el Procesador 802 Monitor señalará una alarma si determina que el Procesador 801 principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activo. Opcionalmente, se pueden emplear diversas formas de dispositivos de señalización audibles y/o visuales como un dispositivo 808 de señal de error, como un pequeño zumbador o un dispositivo LED o LCD, o una forma de conexión por cable o inalámbrica a alguna forma de estación de monitorización con o sin personal.

30 Opcionalmente, el Procesador 802 Monitor registrará un error si determina que el Procesador 801 principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activo.

35 Opcionalmente, se pueden usar diferentes tipos de memoria 804 interna para implementar un registro 809 de error: por ejemplo, memoria RAM, Flash o EEPROM. También se podrían emplear otras formas de implementar tal registro de errores (por ejemplo, usando una colección de puertas adecuadas en un dispositivo FPGA).

40 Opcionalmente, se pueden usar diferentes tipos de memoria 809 externa para implementar un registro 809 de error: por ejemplo, memoria RAM, Flash o EEPROM. También se podrían emplear otras formas de implementar tal registro de errores (por ejemplo, usando una colección de puertas adecuadas en un dispositivo FPGA, o un "disco duro").

45 Opcionalmente, el sistema informático puede tener una entrada 810 de reinicio del sistema, cuyo uso hace que el procesador 801 principal comience sus operaciones desde un estado inicial conocido. La entrada 810 de reinicio del sistema puede tomar la forma de un interruptor o dispositivo similar que se encuentra en o cerca del sistema informático. Alternativamente, la entrada 810 de reinicio del sistema puede implementarse a través de una interfaz cableada o inalámbrica que permite que el sistema se reinicie a distancia.

50 Opcionalmente, el sistema informático puede tener una entrada 811 de detención del sistema, cuyo uso hace que el sistema deje de funcionar. Opcionalmente, el sistema se detendrá en un estado seguro. La entrada 811 de detención del sistema puede adoptar la forma de un interruptor o dispositivo similar que se encuentra en o cerca del sistema informático. Alternativamente, la entrada 811 de detención del sistema puede implementarse a través de una interfaz cableada o inalámbrica que permite que el sistema se reinicie a distancia.

55 La Figura 10 muestra una representación esquemática de una función de despachador de tareas que podría emplearse en el procesador 801 principal mostrado en la Figura 8. Este despachador de tareas debe compararse con la versión estándar que se muestra en la Figura 6: para ser claros: la Figura 6 no tiene el beneficio de la presente invención.

60 El despachador 1000 de tarea mostrado en la Figura 10 comienza determinando 1001 si hay alguna tarea que deba ejecutarse actualmente. Si la respuesta a esta pregunta es "sí" 1002, el despachador notifica al sistema monitor acerca de la tarea que está a punto de ejecutar 1003 antes de ejecutar la tarea 1004. Este repite este proceso (verifique 1001, 1002, notifique 1003, ejecute 1004) hasta que no queden tareas pendientes de ejecución. El despachador luego coloca el procesador en un modo 1005 de ahorro de energía.

65 En la Figura 11 se muestra una posible implementación de la otra mitad del proceso de monitorización introducido en la presente invención.

El proceso de monitorización comienza inicializando el Procesador 1101 Principal.

Opcionalmente, la inicialización del Procesador 801 principal se realizará mediante el Mecanismo de Control. Por ejemplo, el Procesador 802 Monitor puede reiniciar el procesador 801 principal por medio del enlace 807 de reinicio. Reiniciar el procesador 801 principal de esta manera puede, por ejemplo, implicar mantener un pin de reinicio en el procesador 801 principal en estado bajo durante un breve período de tiempo (típicamente unos pocos milisegundos) y luego soltando este pin, asumiendo que el pin de reinicio está "activo bajo". Alternativamente, el Procesador 802 monitor puede, por ejemplo, hacer cíclica la energía al procesador 801 principal (haciendo que se reinicie), por medio de la salida 805 de control del sistema. Alternativamente, el Procesador 802 monitor puede, por medio del enlace de comunicación B 806, enviar un mensaje al procesador 801 principal solicitando que el procesador 801 principal se reinicie.

El procesador monitor puede emplear el enlace de comunicación B para proporcionar información al procesador principal cuando el procesador principal se está configurando después de un reinicio. Tal enlace puede utilizar para proporcionar información al Procesador 801 principal sobre el estado requerido del sistema en el momento de la inicialización. Esta información podría proporcionarse si el sistema se ha reiniciado porque se ha detectado un error. La información proporcionada puede comprender el modo del sistema (en el momento en que se detectó el error). Opcionalmente, la información proporcionada también puede comprender el valor de pulso del sistema (en el momento en que se detectó el error). Mediante el uso de esta información, el Procesador 802 monitor puede ayudar a restaurar el procesador 801 principal a un estado muy similar al estado antes de que ocurriera el error. Alternativamente, el procesador 802 monitor puede estar preconfigurado para proporcionar información de inicialización diferente en caso de un error específico, para garantizar que el procesador principal funcione de forma segura en estas circunstancias.

Cuando el procesador 802 monitor proporciona información al procesador 801 principal sobre el estado requerido del sistema en el momento de la inicialización, esta información puede comprender mensajes entregados a través del enlace de comunicación B 806: el proceso de inicialización del sistema puede comprender entonces la operación de inicialización del sistema llevada a cabo al inicio de la función principal () 501 701.

Opcionalmente, el Procesador 802 monitor puede activar un proceso de inicialización y luego esperar un período fijo antes de continuar.

Opcionalmente, el procesador 801 principal puede enviar un mensaje (a través del enlace 803 de comunicación) para indicar que ha completado cualquier inicialización.

En el siguiente proceso de la Figura 11 (1102), el Procesador 802 monitor selecciona o recopila la Representación de secuencia de tareas activa.

Como se señaló anteriormente, la Representación de la secuencia de tareas que es utilizada por el Procesador 802 monitor se habrá determinado (y almacenado) preferiblemente antes de que el sistema informático comience a funcionar. El proceso 1102 implicará entonces seleccionar esta representación.

Alternativamente, la representación de la secuencia de tareas que utiliza el proceso 802 de monitor puede, por ejemplo, recopilarse en tiempo de ejecución registrando la secuencia de tareas proporcionada por el procesador 801 principal en uno o más puntos cuando el sistema está operativo. Opcionalmente, la recopilación de información de la secuencia de tareas (y la creación de la Representación de la secuencia de tareas activa) se llevará a cabo cuando el sistema informático ingrese a un modo de sistema particular por primera vez.

Opcionalmente, la representación de la secuencia de tareas (si se almacena antes de que el sistema informático comience a funcionar, o se genere cuando el sistema informático está operativo) se almacenará en un dispositivo de almacenamiento de datos interno 804 o externo 812 813.

Como se señaló anteriormente, puede haber varias Representaciones de Secuencia de Tareas diferentes correspondientes a varios Modos del Sistema diferentes. Opcionalmente, estas diferentes representaciones de secuencia de tareas pueden almacenarse en diferentes dispositivos de almacenamiento de datos. Opcionalmente, el Procesador 802 monitor seleccionará la Representación de Secuencia de Tareas activa (correspondiente al Modo de Sistema activo) 1102. A continuación, decimos más sobre la posible representación de los diferentes Modos de Sistema.

Opcionalmente, el procesador 802 monitor esperará entonces 1103 a que el procesador 801 principal complete un período de inicialización de secuencia de tareas correspondiente al modo de sistema activo.

Opcionalmente, el procesador 802 monitor ignorará las salidas del procesador 801 principal durante el período de inicialización de la secuencia de tareas correspondiente al modo de sistema activo.

Una vez que el procesador 801 principal se ha inicializado 1101, la representación de la secuencia de tareas se ha seleccionado 1102 (y, opcionalmente, se ha recopilado 1102) y la inicialización del modo del sistema se completa 1103, el procesador 802 monitor comienza el proceso 1104 de monitorización.

El proceso 1104 de monitorización implica esperar a que el Procesador 801 principal proporcione información sobre la siguiente tarea que pretende ejecutar, y el Procesador 802 monitor verifica que la información de la tarea proporcionada por el Procesador 801 principal coincida con la siguiente entrada en la representación de la secuencia de tareas activa que se seleccionó previamente 1102.

A continuación, se muestra una posible implementación de la función para verificar la secuencia de tareas. Tenga en cuenta que el código debe leerse junto con el código de la matriz TSR_G [] presentado anteriormente.

```
int SCH_Check_Task_Seq(const unsigned char TASK_ID)
{
    static int Tick_list_index = 0;
    int Return_value = 0;
    if (TSR_G[Tick_list_index] != TASK_ID)
    {
        // Task sequence is WRONG
        Return_value = 1;
    }
    // Task sequence is CORRECT
    Return_value = 0;
    // Increment position in array
    if (++Tick_list_index == LENGTH_OF_TSR_G)
    {
        Tick_list_index = 0;
    }
    return Return_value;
}
```

Durante el proceso 1104 de supervisión, se identificará un error de secuencia de tareas 1106 si la ejecución de la tarea propuesta (proporcionada por el Procesador 801 principal) no coincide con la siguiente entrada en la Representación 804 812 813 de secuencia de tareas activa.

Opcionalmente durante el proceso 1104 de monitorización, el tiempo de llegada de la información de la secuencia de tareas desde el procesador 801 principal puede ser verificado 1105 y - si se encuentra que es inconsistente con los límites de tiempo predeterminados - puede identificarse un error de tiempo de la secuencia de tareas. Además, opcionalmente, se puede identificar un error de tiempo de secuencia de tareas si el procesador 802 monitor no recibe informes de actividad sobre una o más tareas del procesador 801 principal durante un período predeterminado. Además, opcionalmente, se puede identificar un error de tiempo de secuencia de tareas si el procesador 802 monitor no recibe informes de ninguna actividad de tareas desde el procesador 801 principal durante un período predeterminado.

Además de manejar cualesquiera errores 1105 de temporización de secuencia de tareas y/o errores 1106 de secuencia de tareas, el procesador 802 monitor también manejará cualquier solicitud para cambiar el modo 1107 desde el procesador 801 principal.

El siguiente ejemplo de código no limitativo ilustra cómo podrían representarse dos secuencias de tareas diferentes (que representan dos modos de sistema diferentes).

```
// Task sequence representation 1
#define SYSTEM_MODE_ID1 200
#define LENGTH_OF_TSR1_G 4
static unsigned char TSR1_G[LENGTH_OF_TSR1_G] =
{
    0,
    2,
    1,
    3
};

// Task sequence representation 2
#define SYSTEM_MODE_ID2 201
#define LENGTH_OF_TSR2_G 5
static unsigned char TSR2_G[LENGTH_OF_TSR2_G] =
{
    0,
    2,
    4,
    1,
    3
};
```

En el ejemplo anterior, como en los ejemplos anteriores, las matrices que almacenan las representaciones de secuencia de tareas contienen ID de tarea, con valores de ID de tarea que varían de 0 a 5 en este caso. Aquí, las dos matrices de tareas diferentes (TSR2_G [] y TSR2_G []) representan las diferentes secuencias de tareas para dos modos de sistema posibles: estos modos en sí tienen ID de modo de sistema (representados como SYSTEM_MODE_ID1 con un valor de 200, y SYSTEM_MODE_ID2 con un valor de 201 en este ejemplo).

Además, asumimos (a los efectos de este ejemplo no limitativo solamente) que el sistema puede operar con hasta 200 posibles tareas del sistema, con ID de tareas que varían en valor de 0 a 199. En este caso, el procesador 801 principal podría reportar la siguiente información al procesador 802 monitor a través del enlace 803 de comunicación:

```
200
0
2
1
3
0
2
1
3
201
0
2
4
1
3
```

En el ejemplo anterior, los valores 200 no son ID de tarea válidos, pero son ID de modo de sistema válidos. Esto refleja el hecho de que, al comienzo de la secuencia anterior, el procesador 801 principal está ejecutando tareas desde la representación de la secuencia de tareas con ID 200. Después de que haya pasado un tiempo, el procesador 801 principal indica que el modo del sistema ha cambiado ahora (por ejemplo, como resultado de entradas de un usuario, o cambios en el entorno operativo) y que ha comenzado a ejecutar tareas desde la Representación de Secuencia de Tareas con ID 201.

Opcionalmente, durante el proceso 1104 de monitorización, se puede comprobar el tiempo de cualquier cambio en el modo del sistema y, si se encuentra que es inconsistente con los valores predeterminados, se puede identificar un error de tiempo del modo del sistema. Opcionalmente, se identificará un error de sincronización del modo del sistema si, por ejemplo, el modo del sistema cambia más rápido o más lentamente de lo que los límites predeterminados sugerirían que es probable. Además, opcionalmente, se identificará un error de temporización del modo del sistema si el procesador monitor no recibe reportes de los cambios del modo del sistema del procesador 801 principal durante más de un período predeterminado.

Opcionalmente, durante el proceso 1104 de monitorización, se comprobará la secuencia de cualquier cambio en el modo del sistema y, si se encuentra que es inconsistente con los valores predeterminados, se puede identificar un error de secuencia del modo del sistema.

Opcionalmente, en el caso de que se identifique 1105 un error de tiempo de secuencia de tareas, o se identifique 1105 un error de tiempo de modo de sistema, o se identifique 1106 un error de secuencia de tareas, o se identifique 1106 un error de secuencia de modo de sistema, el procesador 802 monitor señalará 1109 el estado del sistema (por ejemplo, a través de un dispositivo 808 de señal de error) y/o registrará el estado del sistema (por ejemplo, en un registro 809 de error), y/o aplicará una o más operaciones de control del sistema por medio de la salida 805 de control del sistema.

Opcionalmente, en el caso de que se identifique 1105 un error de tiempo de secuencia de tareas, o se identifique 1105 un error de tiempo de modo de sistema, o se identifique 1106 un error de secuencia de tareas, o se identifique 1106 un error de secuencia de modo de sistema, el procesador 802 monitor ignorará este error, si el número de ocurrencias del error particular está por debajo de un umbral predeterminado.

Opcionalmente, en el caso de que se identifique 1105 un error de tiempo de secuencia de tareas, o se identifique 1105 un error de tiempo de modo de sistema, o se identifique 1106 un error de secuencia de tareas, o se identifique 1106 un error de secuencia de modo de sistema, el procesador 802 monitor ignorará este error, si la tasa de ocurrencias del error particular está por debajo de un umbral predeterminado.

Opcionalmente, el procesador 802 monitor también manejará cualquier solicitud para detener el sistema 1108 informático. Dichas solicitudes para detener el sistema informático pueden surgir a través de la entrada 811 de detención del sistema. Si se recibe una solicitud de detención del sistema, el procesador 802 monitor puede 1109 señalar el estado del sistema (por ejemplo, a través de un dispositivo 808 de señal de error) y/o registrar el estado del sistema (por ejemplo, en un registro 809 de error) y/o aplica una o más operaciones de control del sistema por medio de la salida 805 de control del sistema.

El algoritmo 1100 ilustrado en la Figura 11 procesa cualquier entrada de la entrada 810 de reinicio del sistema o la entrada 811 de detección del sistema en los procesos 1108, 1109 y 1110. En el caso de una entrada 810 de reinicio del sistema o una entrada 811 de detección del sistema, el procesador 802 monitor puede señalar 1109 el estado del sistema (por ejemplo, a través de un dispositivo 808 de señal de error) y/o registrar el estado del sistema (por ejemplo, en un registro 809 de error), y/o aplicar una o más operaciones de control del sistema por medio de la salida 805 de control del sistema. En el caso de una entrada 811 de detención del sistema, el procesador 802 monitor puede 1109 detener el procesador 801 principal, por medio (por ejemplo) del enlace 807 de reinicio. Detener el procesador 801 principal de esta manera puede, por ejemplo, implicar mantener un pin de reinicio en el procesador principal en un estado bajo (y bloquearlo en este estado), asumiendo que el pin de reinicio está "activo bajo".

En el caso de una entrada 810 de reinicio del sistema, el procesador 802 monitor puede 1110 reiniciar el procesador 801 principal por medio del enlace 807 de reinicio. Reiniciar el procesador 801 principal de esta manera puede, por ejemplo, implicar mantener un pin de reinicio en el procesador 801 principal en estado bajo durante un breve período (normalmente pocos milisegundos) y luego soltando este pin, asumiendo que el pin de reinicio está "activo bajo". Alternativamente, el procesador 802 monitor puede, por ejemplo, hacer cíclica la energía al procesador 801 principal (haciendo que se reinicie), por medio de la salida 805 de control del sistema. Alternativamente, el procesador 802 monitor puede enviar un mensaje al procesador 801 principal solicitando que el procesador 801 principal se reinicie por sí mismo, mediante el enlace de comunicación B 806.

El procesador 802 monitor cesará sus operaciones después de completar los procesos establecidos en 1110.

Tenga en cuenta que, como se explicó anteriormente, los diseñadores de sistemas TT a menudo eligen equilibrar el código de la tarea, para garantizar que el comportamiento de temporización del sistema sea temporalmente determinista. Opcionalmente, el uso de código balanceado puede emplearse en aplicaciones de la presente invención.

El siguiente ejemplo no limitante ilustra por qué el uso del equilibrio de código puede ser beneficioso cuando se emplea un programador TTH (o cualquier otro programador que implique la prevención de tareas).

Por favor, véase de nuevo la Figura 3. La Figura 3 primero muestra 301 un conjunto de dos tareas cooperativas donde la Tarea A 303 y/o la Tarea B 304 pueden ser prevenidas periódicamente por la Tarea P 305 de alta prioridad.

La secuencia de tareas completa para el diagrama 301 superior en la Figura 3 podría listarse como sigue: Tarea P, Tarea A, Tarea P, Tarea B, Tarea P, Tarea A, Tarea P, Tarea B.

Si el código para todas estas tareas está equilibrado, entonces esta secuencia siempre será la misma (cada vez que se ejecute esta secuencia de tareas).

Si el código no está equilibrado, entonces podemos, por ejemplo, obtener la secuencia de tareas que se muestra en la mitad inferior de la Figura 3 302 durante la ejecución del sistema. En este caso 302, la secuencia de tareas sería la siguiente: Tarea P, Tarea A, Tarea B, Tarea P, Tarea P, Tarea A, Tarea P, Tarea B. Sin equilibrio de código, este diagrama 302 inferior puede interpretarse como un error de secuencia de tareas (que puede no ser lo que se pretendía).

Hay más opciones de detención y reinicio del sistema disponibles para el desarrollador.

Por ejemplo, en el caso de un error, el procesador 802 monitor puede solicitar primero, a través del enlace de comunicación B 806, que el procesador 801 principal realice una operación ordenada de apagado, como se describe (por medio de un ejemplo no limitante) en la Figura 12 (1200). Si el procesador 801 principal continúa ejecutándose en estas circunstancias, el procesador 802 monitor puede entonces intentar forzar al procesador 801 principal a detenerse, por medio del enlace 807 de reinicio. Además, opcionalmente, el procesador 802 monitor puede cortar la energía del procesador 801 principal (por medio de la salida 805 de control del sistema) en estas y otras circunstancias similares.

La Figura 13 ilustra de forma esquemática otra posible opción de implementación para la presente invención. En la implementación 1300 mostrada en la Figura 13, el Procesador 1302 monitor no almacena una representación de la programación de tareas predeterminada, sino que, en cambio, calcula esta programación en tiempo de ejecución. En estas circunstancias, el procesador 1302 monitor puede ejecutar una versión de un programador activado por tiempo controlado por uno o más "pulsos" del temporizador. Este programador activado por tiempo puede ser un programador cooperativo activado por tiempo (TTC) o un programador híbrido activado por tiempo (TTH): el tipo de programador elegido probablemente (pero no necesariamente) se elegirá para que coincida con el que se ejecuta en el procesador 1301 principal.

La Figura 14 ilustra en forma 1400 esquemática la operación del software que podría ejecutarse en el procesador 1302 monitor ilustrado en la Figura 13.

Como ejemplo, la lista de códigos a continuación muestra un programador TTC que podría emplearse en el Procesador 1301 principal que se usa junto con el software que se muestra esquemáticamente en la Figura 14.

```
int main (void)
{
    SCH_TTC_Init(); // Set up the scheduler
    // Other init functions
    // ...
    // Add Task_A, Task_B and Task_C to the schedule
    // Parameters: Task name, task offset, task period
    SCH_TTC_Add_Task(Task_A, 0, 10);
    SCH_TTC_Add_Task(Task_B, 0, 15);
    SCH_TTC_Add_Task(Task_C, 0, 33);
    SCH_TTC_Start(); // Start the schedule
    while (1)
    {
        SCH_TTC_Dispatch_Tasks();
    }
    return 1;
}
```

Parte del código correspondiente que se ejecuta en el procesador 1302 monitor podría tomar la forma que se muestra en la lista a continuación:

```
int main(void)
{
    SCH_TTC_Monitor_Init(); // Set up the scheduler
    // Other init functions
    // ...
    // Add Task_A, Task_B and Task_C to the schedule
    // Parameters: Task name, task offset, task period
    SCH_TTC_Model_Add_Task(Task_A, 0, 10);
    SCH_TTC_Model_Add_Task(Task_B, 0, 15);
    SCH_TTC_Model_Add_Task(Task_C, 0, 33);
    Next_task_ID = SCH_TTC_Model_Dispatch_Tasks();
}
```

En este ejemplo, se usará una versión de la función de despachador (como se usa en el procesador 1301 principal y se describe en detalle, por ejemplo, en la Referencia 5), cuando sea necesario, para determinar qué tarea debe ejecutarse a continuación.

La Figura 15 ilustra en forma esquemática otra posible opción de implementación para la presente invención. En la implementación 1500 mostrada en la Figura en la que el proceso 1502 principal y el proceso 1503 monitor se ejecutan en un solo procesador 1501 y el proceso 1503 monitor emplea un área de memoria 1505 interna para almacenar información sobre la representación de secuencia de tareas predeterminada.

La Figura 16 ilustra en forma esquemática la operación del software que podría ejecutarse en el proceso 1503 monitor ilustrado en la Figura 15.

En la Figura 16, el proceso 1600 de monitorización implica la inicialización del proceso 1601 principal.

Opcionalmente, la inicialización del proceso 1502 principal se llevará a cabo mediante el mecanismo de control. Por ejemplo, el proceso 1503 monitor puede, por ejemplo, hacer que el proceso 1502 principal (y todo el procesador 1501) se reinicie provocando el desbordamiento de un temporizador de "vigilancia". Alternativamente, el proceso 1503 monitor puede enviar un mensaje al proceso 1502 principal solicitando que el proceso 1502 principal se reinicie por sí mismo, por medio del enlace de comunicación B 1508.

Opcionalmente, el proceso 1503 monitor proporcionará información al proceso 1502 principal sobre el estado requerido del sistema en el momento de la inicialización. Esta información podría proporcionarse si el sistema se ha reiniciado porque se ha detectado un error. La información proporcionada puede comprender el modo del sistema (en el momento en que se detectó el error). Opcionalmente, la información proporcionada también puede comprender el valor de pulso del sistema (en el momento en que se detectó el error). Mediante el uso de esta información, el proceso 1503 monitor puede ayudar a restaurar el proceso 1502 principal a un estado muy similar al estado antes de que ocurriera el error. Alternativamente, el proceso 1503 monitor puede estar preconfigurado para proporcionar información de inicialización diferente en el caso de un error, para asegurar que el proceso principal funciona de forma segura en estas circunstancias.

Cuando el proceso 1503 monitor proporciona información al proceso 1502 principal sobre el estado requerido del sistema en el momento de la inicialización, esta información puede comprender mensajes entregados a través del enlace de comunicación B 1508: el proceso de inicialización del sistema puede comprender entonces la operación de inicialización del sistema llevada a cabo al inicio de la función main () 501, 701.

Opcionalmente, el proceso 1503 monitor puede activar un proceso de inicialización y luego esperar un período fijo antes de continuar.

Opcionalmente, el proceso 1502 principal puede enviar un mensaje (a través del enlace 1504 de comunicación) para indicar que ha completado cualquier inicialización.

El proceso 1600 de monitorización (mostrado en la Figura 16) implica entonces la selección (o recopilación) de los detalles de la secuencia de tareas para el modo 1602 operativo actual. El proceso 1503 monitor entonces espera (si es necesario) hasta que se completa 1603 el período de inicialización del modo, antes de revisar la información (proporcionada por el proceso 1502 principal) sobre la siguiente tarea y/o el modo del sistema que se ejecutará 1604. Si hay, por ejemplo, una demora inesperada al proporcionar información sobre la tarea o el modo, el proceso 1503 monitor puede "entrar el tiempo de espera" 1605. En ausencia del comportamiento de tiempo de espera, el proceso 1503 monitor tomará medidas si la información del proceso 1502 principal sobre la siguiente tarea y/o el modo del sistema que se ejecutará no es la prevista 1606: esto puede ir seguido de un cambio del modo 1607 del sistema y el procesamiento de las solicitudes 1608 de detención o reinicio del sistema. En caso de comportamiento de tiempo de espera o la recepción de información del proceso 1502 principal que sugiere que el proceso 1502 principal está a punto de realizar una acción que no esté de acuerdo con el programa de tareas activo, el proceso 1503 monitor puede 1609 [i] señalar y/o registrar el estado del sistema, y/o [ii] detener el proceso objetivo, y/o [iii] aplicar operaciones apropiadas de control del sistema a través de la salida 1506 de control del sistema. El proceso 1503 monitor puede entonces 1610 reiniciar el sistema informático.

En toda la memoria descriptiva, a menos que el contexto exija lo contrario, los términos "comprende" o "incluye", o variaciones tales como "comprende" o "que comprende", "incluye" o "que incluye" se entenderá que implica la inclusión de un número entero o grupo de números enteros, pero no la exclusión de ningún otro número entero o grupo de números enteros.

Referencias

1. Keith M. Klug, Steven R. Tugenberg "Functional lockstep arrangement for redundant processors", US Patent US5226152 A, 1993.

2. Nahmsuk Oh, Philip P. Shirvani and Edward J. McCluskey, "Control Flow Checking by Software Signatures", Technical Report CRC-TR00-4, Center for Reliable Computing, Stanford University, 2000.
- 5 3. M. J. Pont and H.L.R. Ong "Using watchdog timers to improve the reliability of TTCS embedded systems", in Hruby, P. and Soressen, K. E. [Eds.] Proceedings of the First Nordic Conference on Pattern Languages of Programs, September, 2002 ("VikingPloP 2002"), pp.159-200. Published by Microsoft Business Solutions. ISBN: 87-7849-769-8, 2003.
- 10 4. J.-T. Leung and M. Merril. "A note on preemptive scheduling of periodic real-time tasks", Information Processing Letters, 3(11), 1980.
5. M. J. Pont, "Patterns for Time-Triggered Embedded Systems", Addison-Wesley / ACM Press, 2001.
- 15 6. M. J. Pont, "Embedded C", Addison-Wesley, 2002.
7. P. Puschner and A. Burns, "Writing temporally predictable Code," in Proceedings of the seventh International Workshop on Object-Oriented Real-Time Dependable Systems, 2002.
- 20 8. P. Puschner, "Is WCET Analysis a non-problem? Towards new Software and Hardware architectures," in 2nd International Workshop on Worst Case Execution Time Analysis, Vienna, Austria, June 2002.
9. R. Kirner and P. Puschner, "Discussion of Misconceptions about WCET Analysis," in 3rd Euromicro International workshop on WCET Analysis, 2003.
- 25 10. A. K. Gendy and M. J. Pont "Towards a generic 'Single Path Programming' solution with reduced power consumption", in International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2007, Las Vegas, Nevada, USA, 2007.
- 30 11. M. J. Pont, S. Kurian, and R. Bautista-Quintero, "Meeting Real-time Constraints Using "Sandwich Delays"", TPLOP,LNCS, pp. 94-102, 2009.

REIVINDICACIONES

1. Un sistema informático activado por tiempo (800) que comprende:

- 5 un procesador (801) principal que comprende un programador activado por tiempo adaptado para ejecutar una o más tareas de acuerdo con uno o más programas de tareas predeterminados que se determinan, validan y verifican cuando se diseña el sistema, y solo uno de ellos, un programa de tareas activo, estará activo en cualquier momento; y
 - 10 un Procesador (802) monitor adaptado para determinar de antemano si el procesador (801) principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activo usando una representación (804) de secuencia de tareas que comprende un registro de una secuencia de ejecuciones de tarea que se espera que siga el procesador (801) principal en el programa de tareas activos; y
 - 15 un enlace (803) de comunicación para pasar información sobre futuras ejecuciones de tareas entre el procesador (801) principal y el procesador (802) monitor; y
 - 20 un mecanismo (807) de control mediante el cual el procesador (802) monitor puede detener o reiniciar el procesador (801) principal y apagar, deshabilitar, habilitar o configurar los dispositivos conectados que están bajo el control del procesador (801) principal, si el procesador (802) monitor determina que el procesador (801) principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activo.
2. Un sistema (800) informático de acuerdo con la reivindicación 1, en el que el procesador (801) principal y el procesador (802) monitor comprenden cada uno, uno o más núcleos de procesador "blandos" o "duros" (que ejecutan algún software) y/o uno o más núcleos de hardware (que no ejecutan ningún software).
- 25 3. Un sistema (800) informático de acuerdo con cualquier reivindicación anterior, en el que el procesador (801) principal y el procesador (802) monitor se implementan ambos por medio de un solo chip de procesador.
- 30 4. Un sistema (800) informático de acuerdo con cualquier reivindicación anterior, en el que el procesador (801) principal y el procesador (802) monitor se implementan mediante chips de procesador separados.
5. Un sistema (800) informático de acuerdo con la reivindicación 4, en el que el procesador (801) principal y el procesador (802) monitor están separados físicamente.
- 35 6. Un sistema (800) informático de acuerdo con cualquier reivindicación anterior, en el que el procesador (802) monitor tiene acceso a al menos una representación (804) de secuencia de tareas predeterminada, habiendo sido determinada y almacenada esta representación (804) de secuencia de tarea un momento antes de que el sistema informático comience a funcionar.
- 40 7. Un sistema (800) informático de acuerdo con cualquier reivindicación anterior, en el que el procesador (802) monitor crea una o más representaciones (804) de secuencia de tareas durante la operación del sistema.
- 45 8. Un sistema (800) informático de acuerdo con la reivindicación 7, en el que el procesador (802) monitor crea una o más representaciones (804) de secuencia de tareas durante la operación del sistema, registrando la secuencia de tareas del procesador (801) principal en puntos durante la operación del sistema.
9. Un sistema (800) informático de acuerdo con cualquier reivindicación anterior, en el que cada representación (804) de secuencia de tareas se almacena en uno o más dispositivos de almacenamiento de datos.
- 50 10. Un sistema (800) informático de acuerdo con cualquier reivindicación anterior, en el que el procesador (802) monitor no procesa información sobre la siguiente tarea a ejecutar para cada tarea en el programa de tareas activo.
11. Un sistema (800) informático de acuerdo con cualquier reivindicación anterior, en el que el mecanismo (807) de control comprende una salida de control del sistema y/o un enlace de reinicio y/o un enlace de comunicación B.
- 55 12. Un sistema (800) informático de acuerdo con la reivindicación 11, en el que el procesador (802) monitor intenta colocar el procesador (801) principal en un estado seguro antes de detener el procesador (801) principal.
- 60 13. Un sistema (800) informático de acuerdo con cualquier reivindicación anterior, en el que el procesador (802) monitor señala una alarma si determina que el procesador (801) principal está a punto de ejecutar una tarea que no está de acuerdo con el programa de tareas activo.

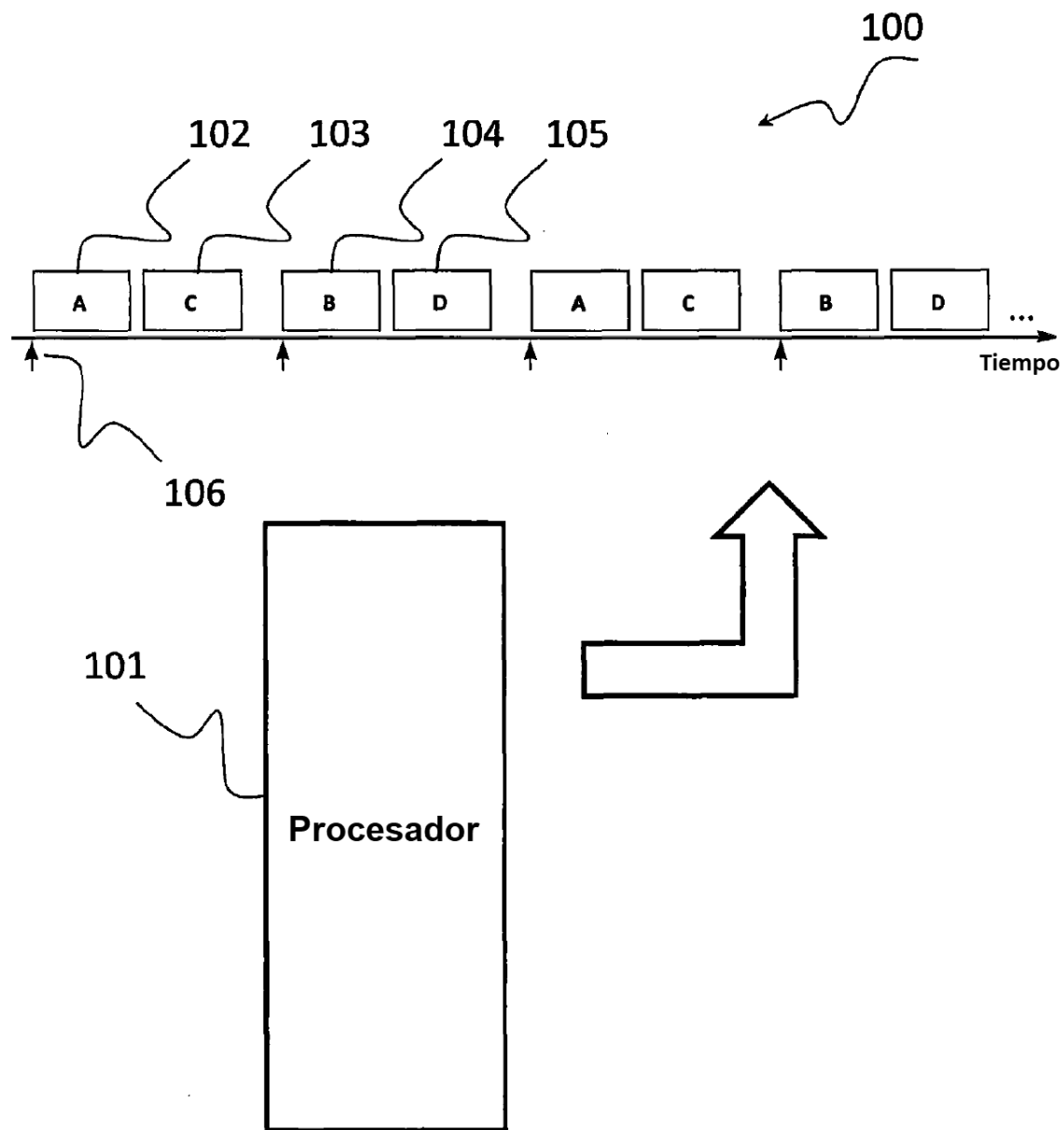


Figura 1

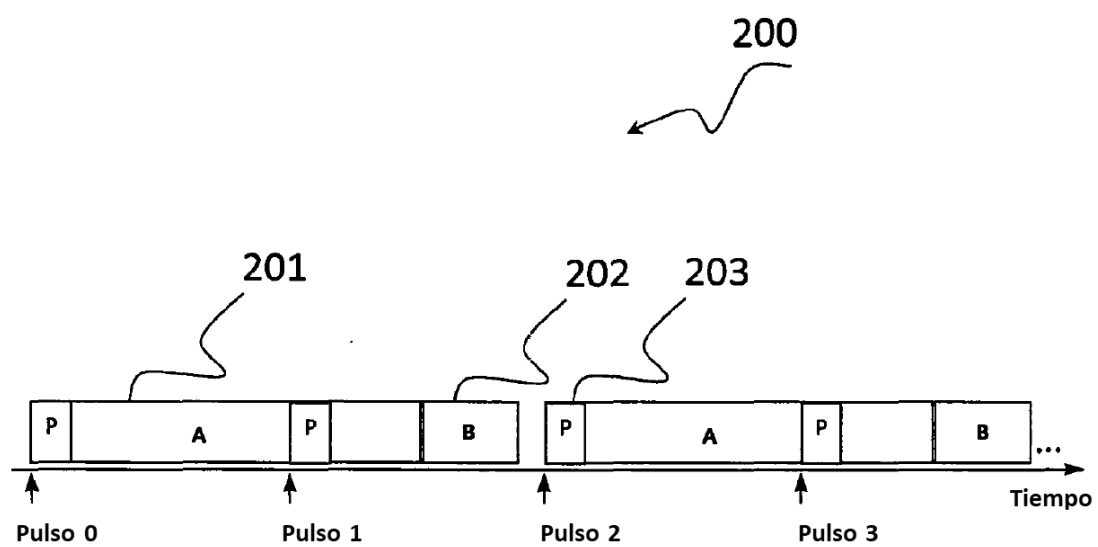


Figura 2

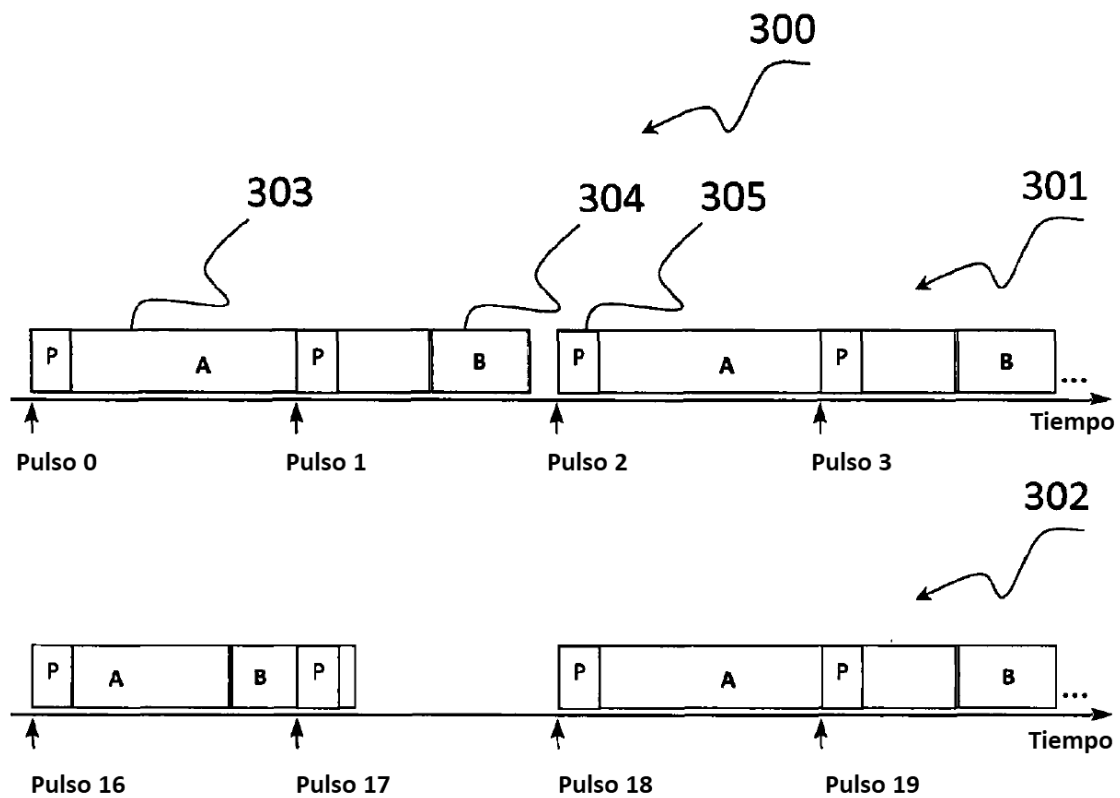


Figura 3

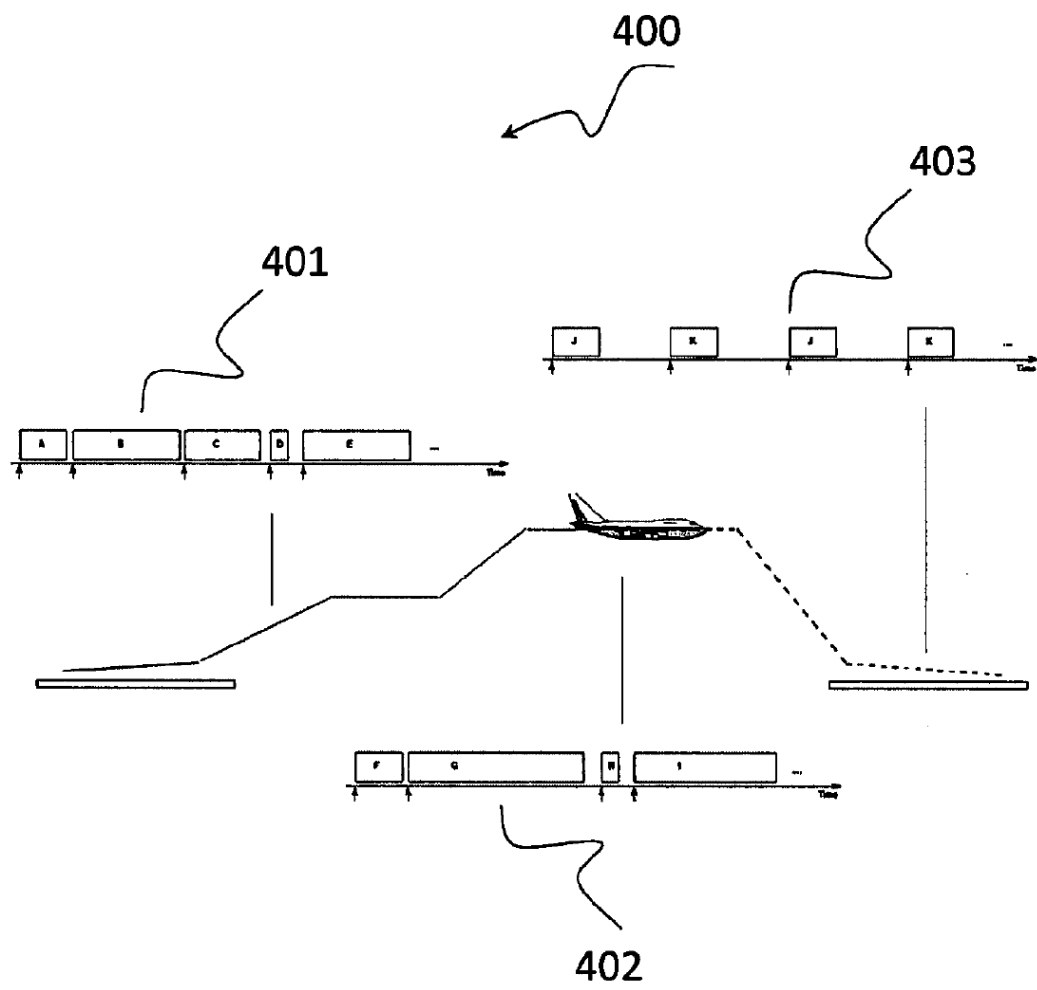


Figure 4

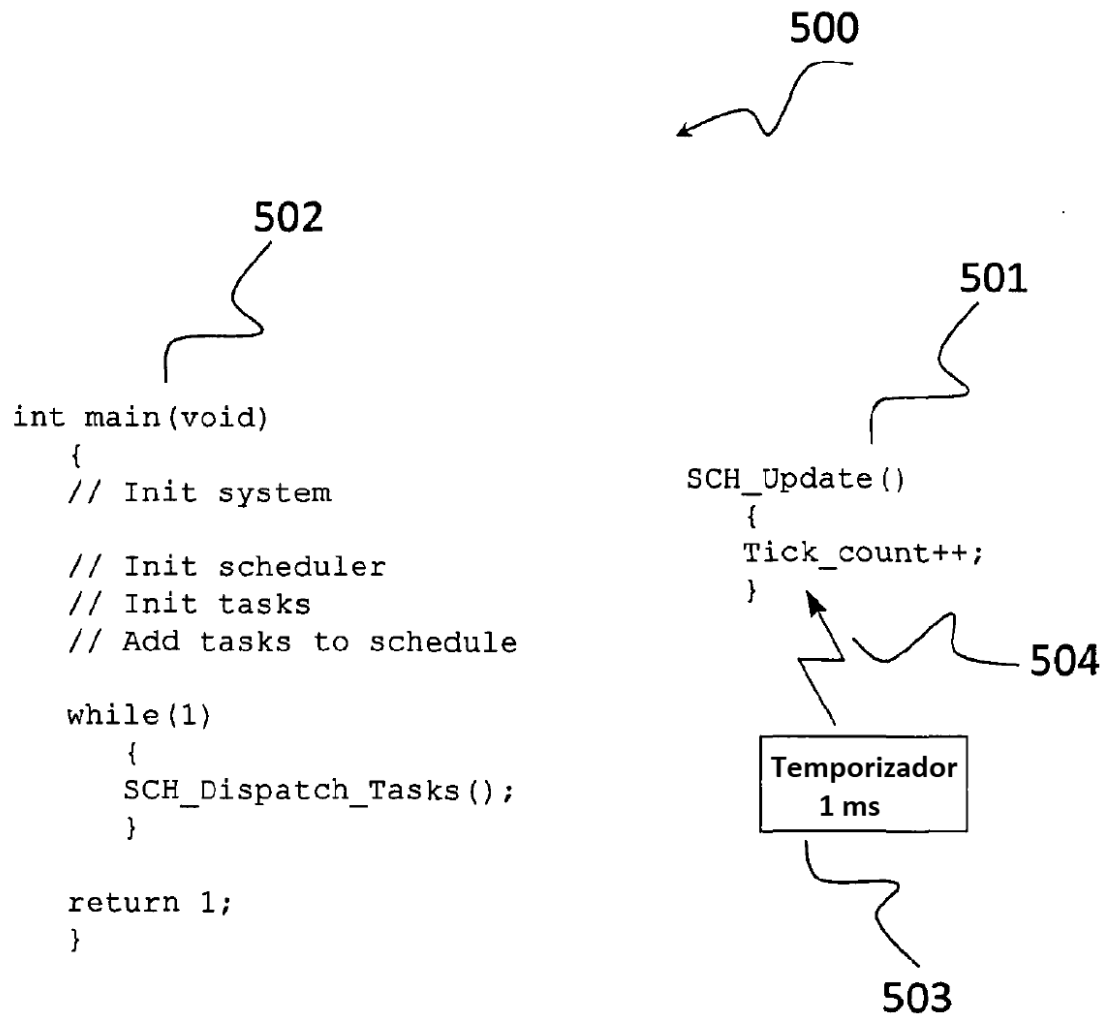


Figura 5

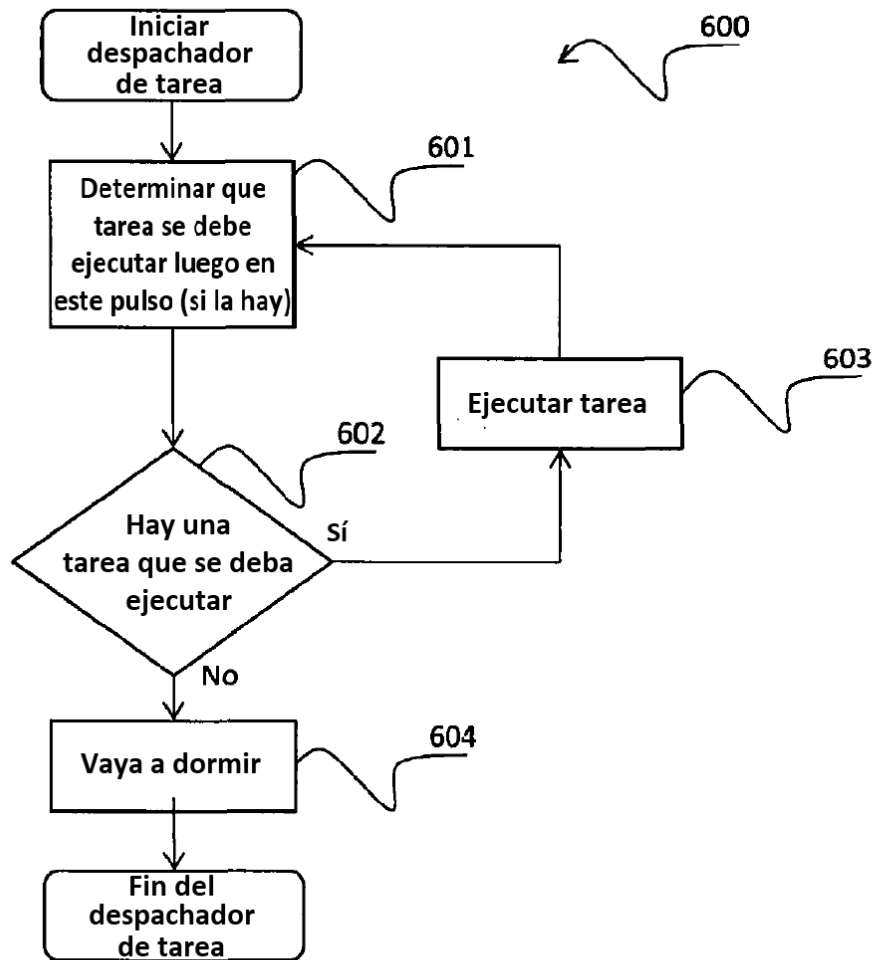


Figura 6

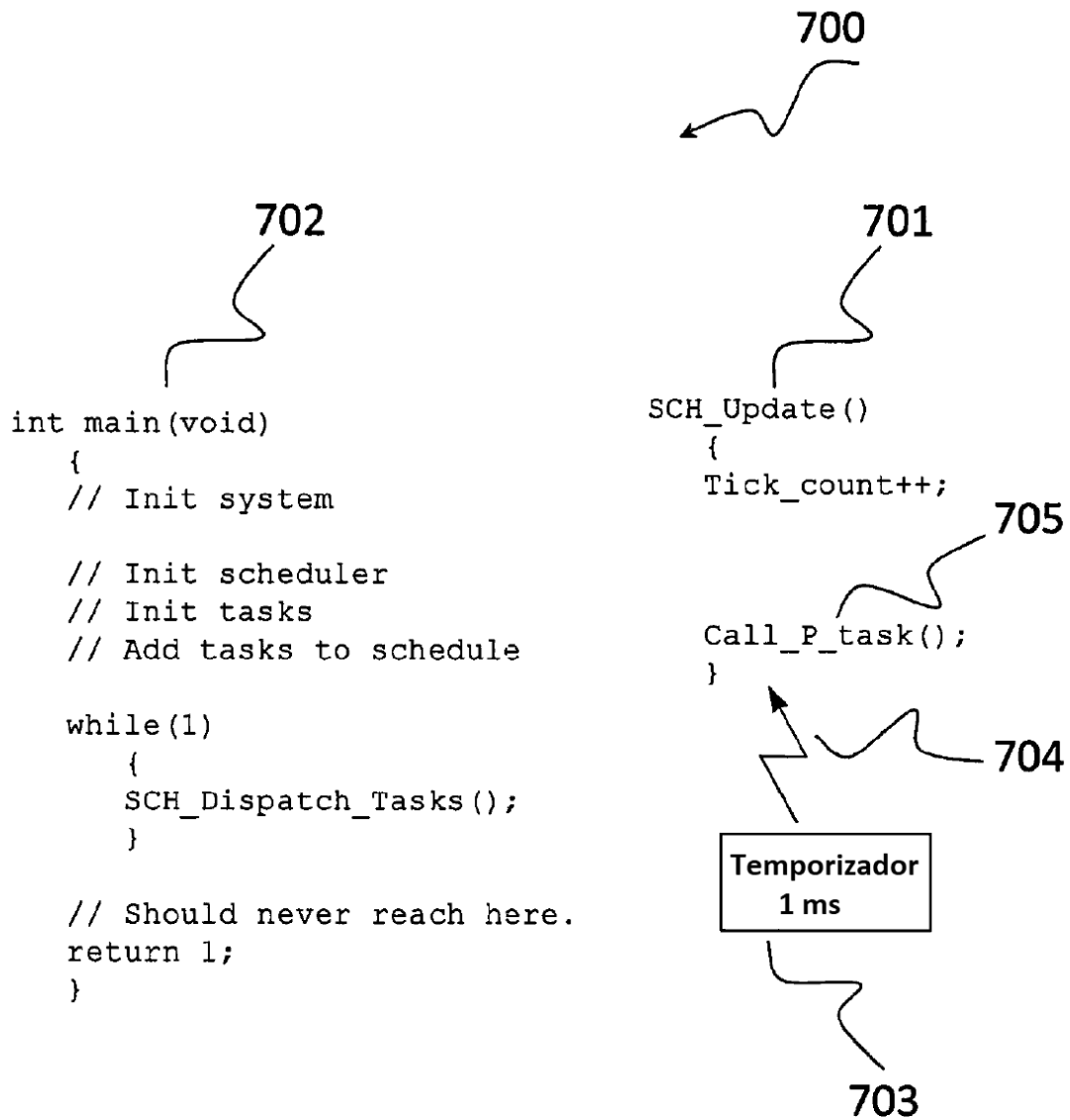


Figura 7

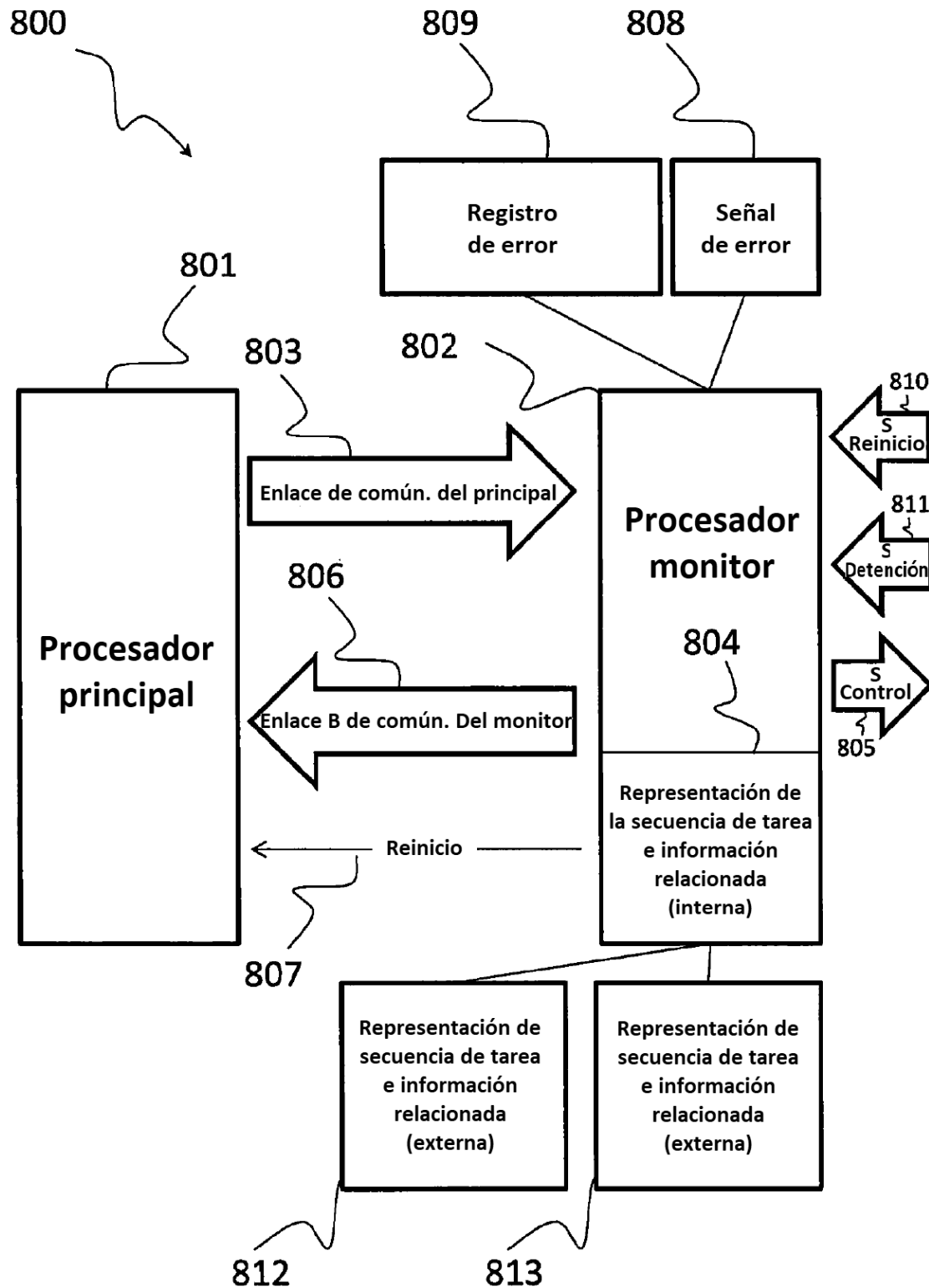


Figura 8

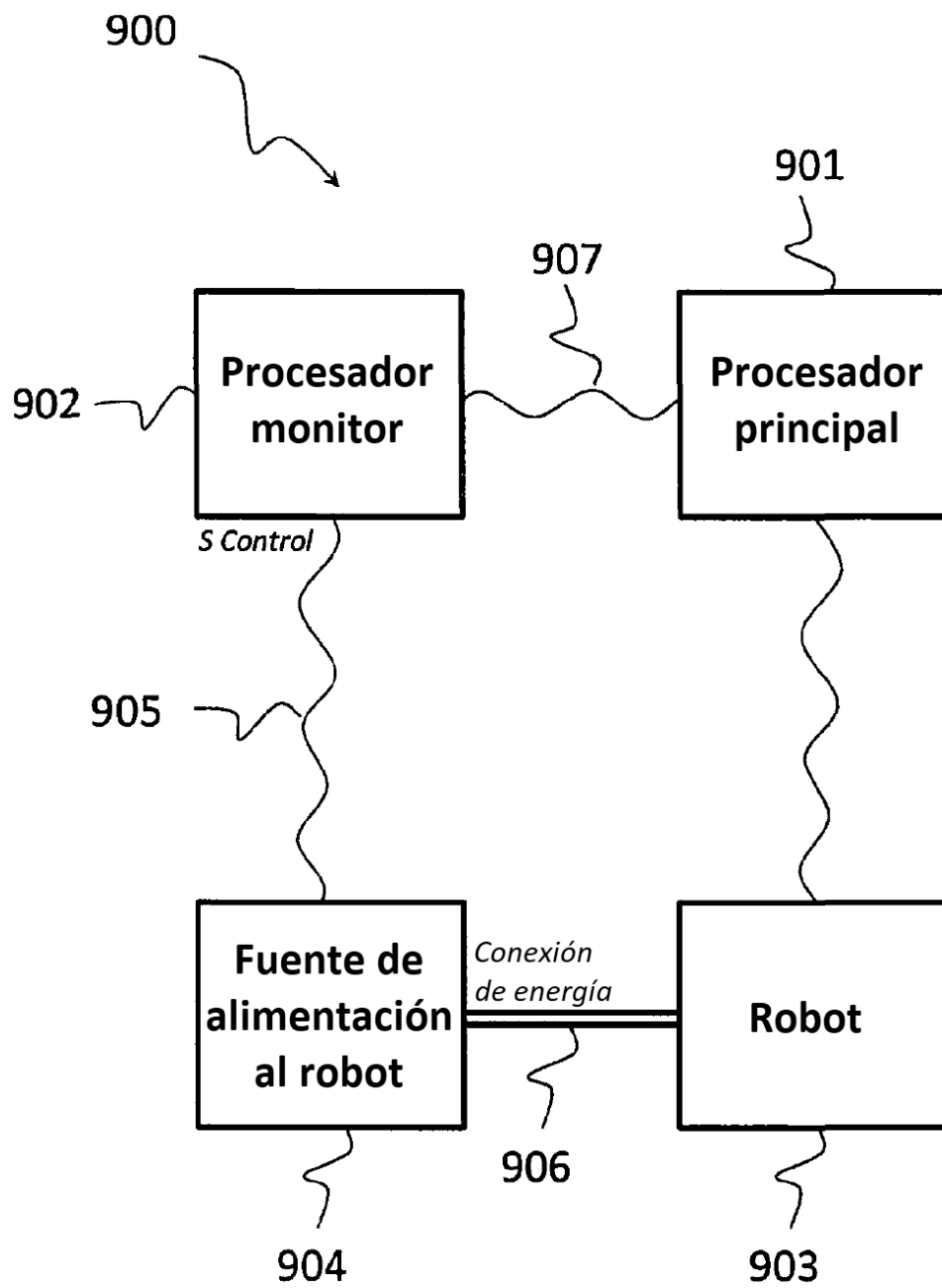


Figura 9

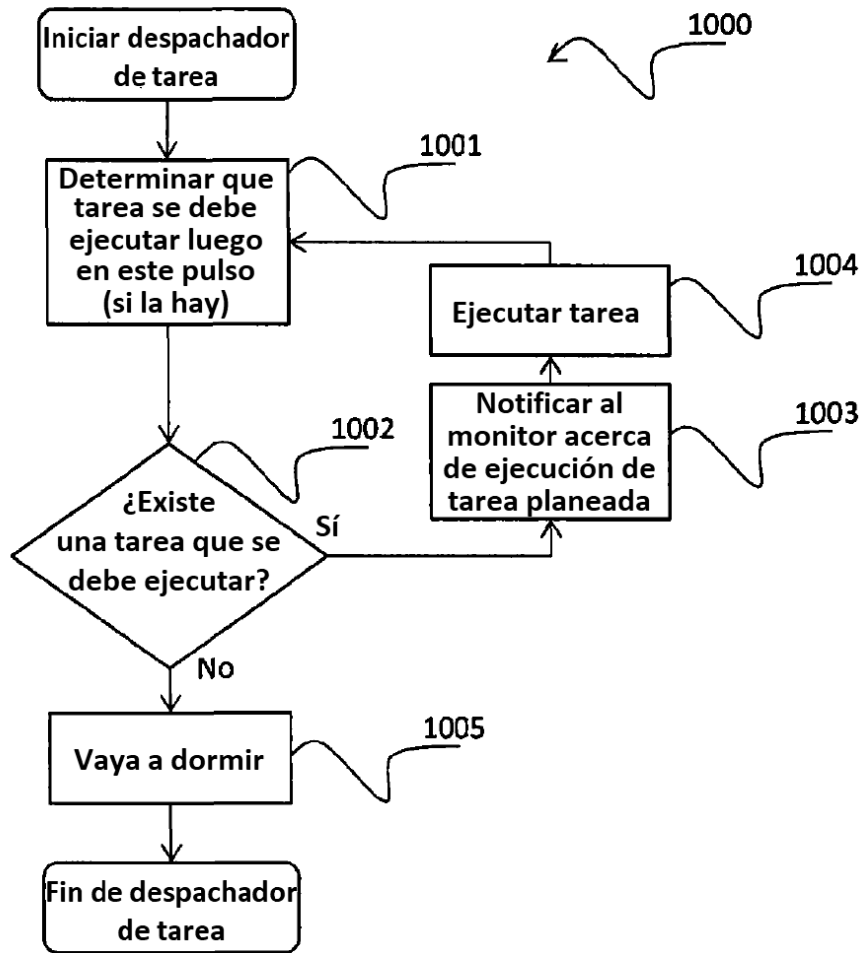


Figura 10

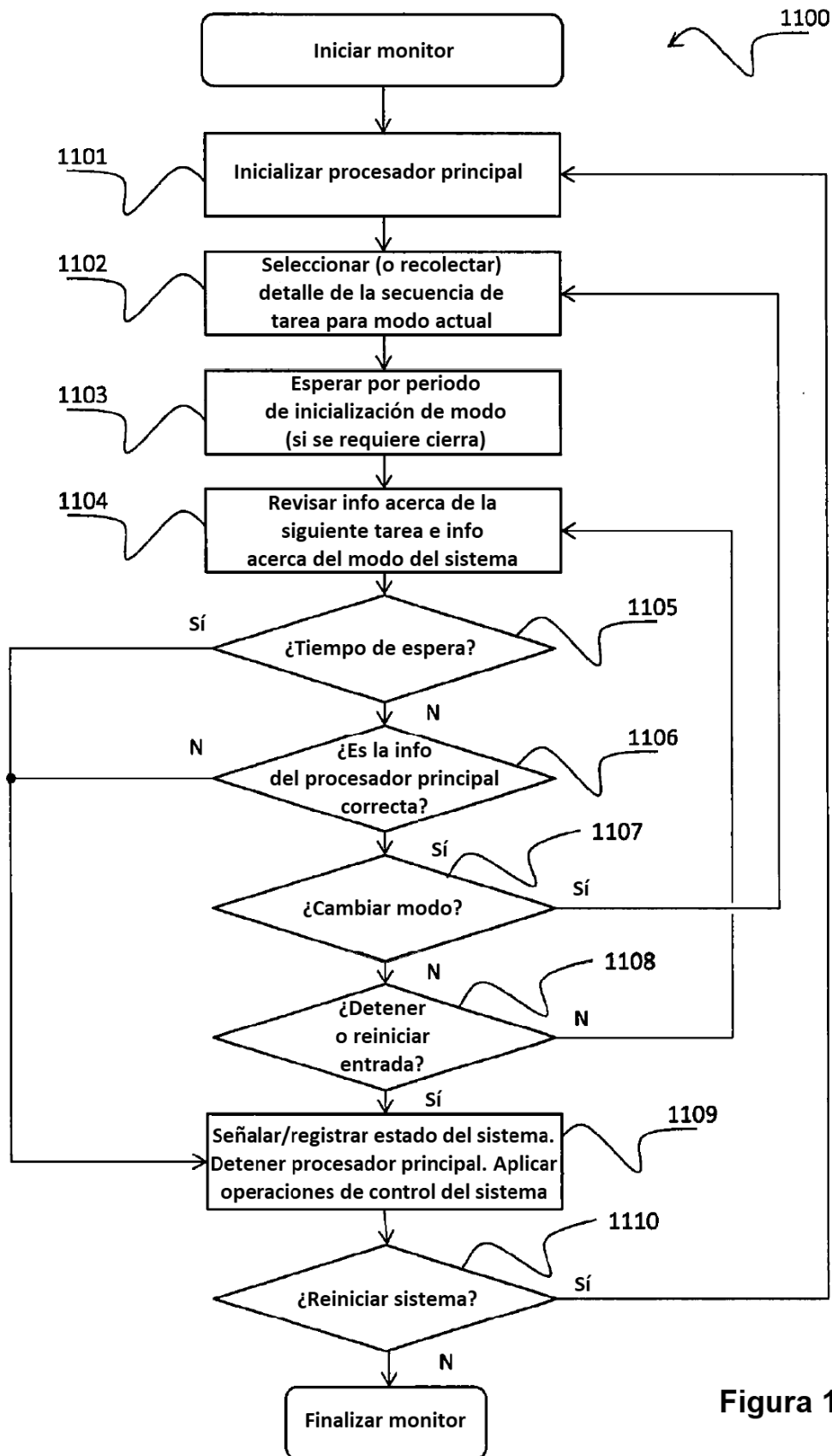


Figura 11

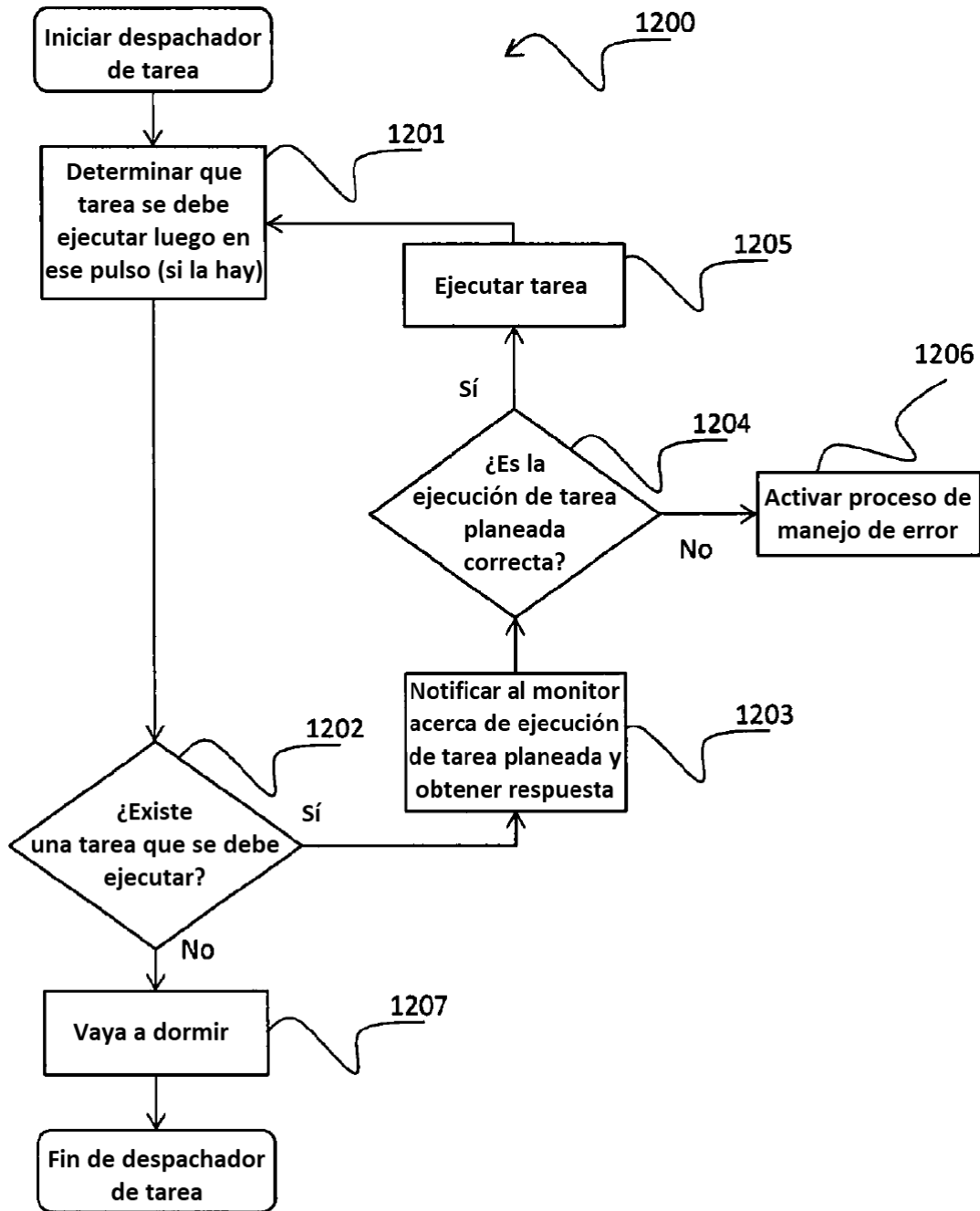


Figura 12

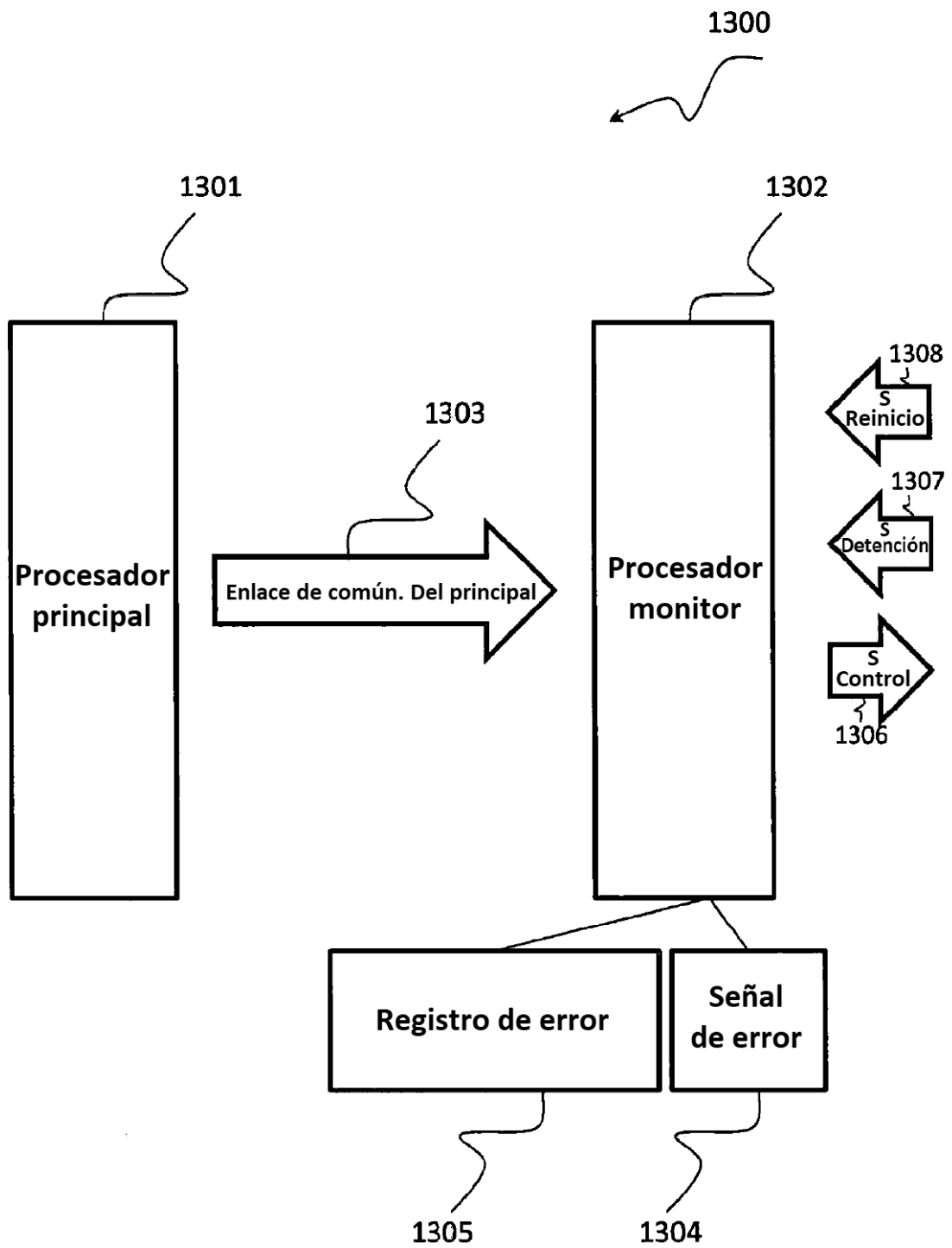


Figura 13

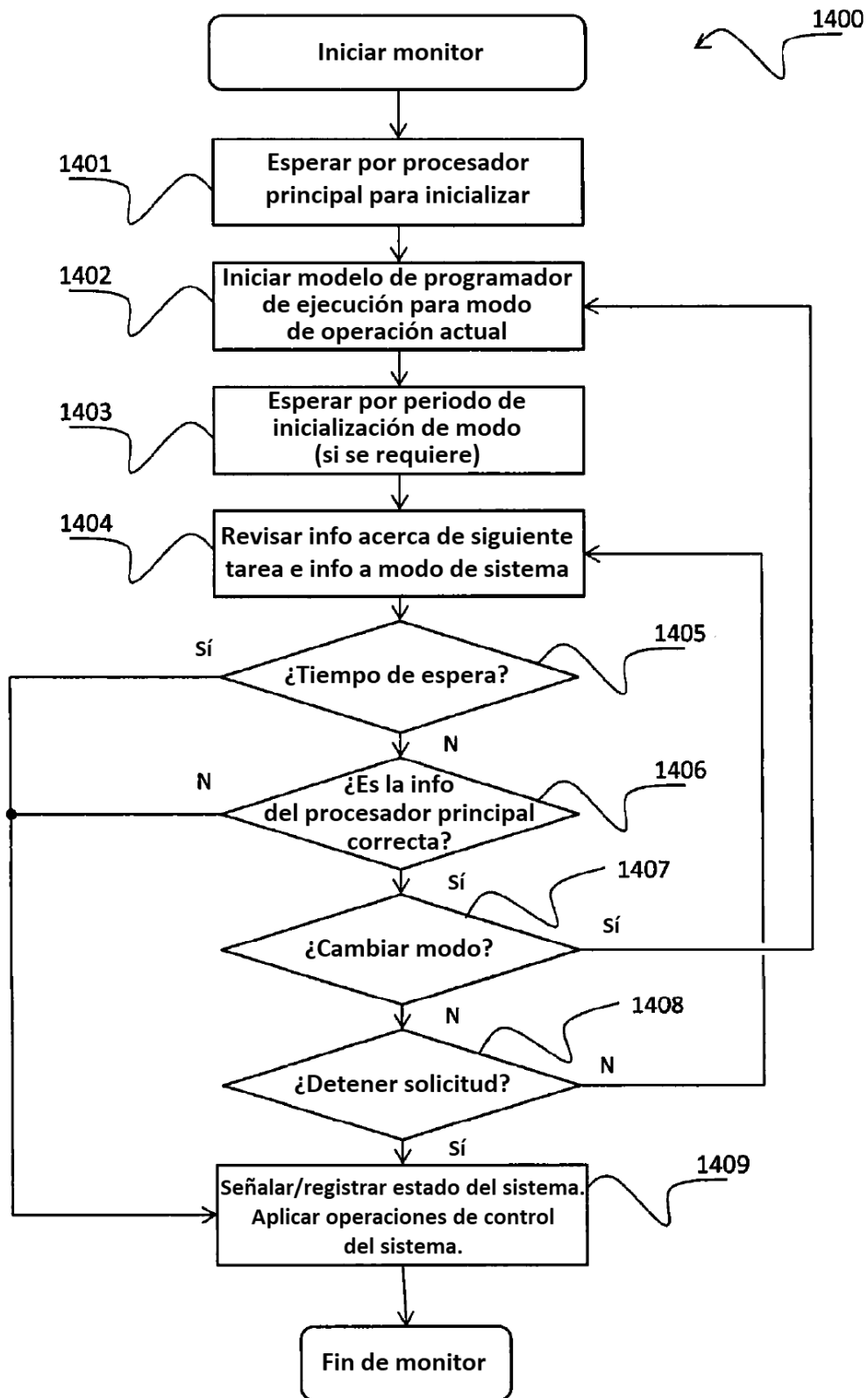


Figura 14

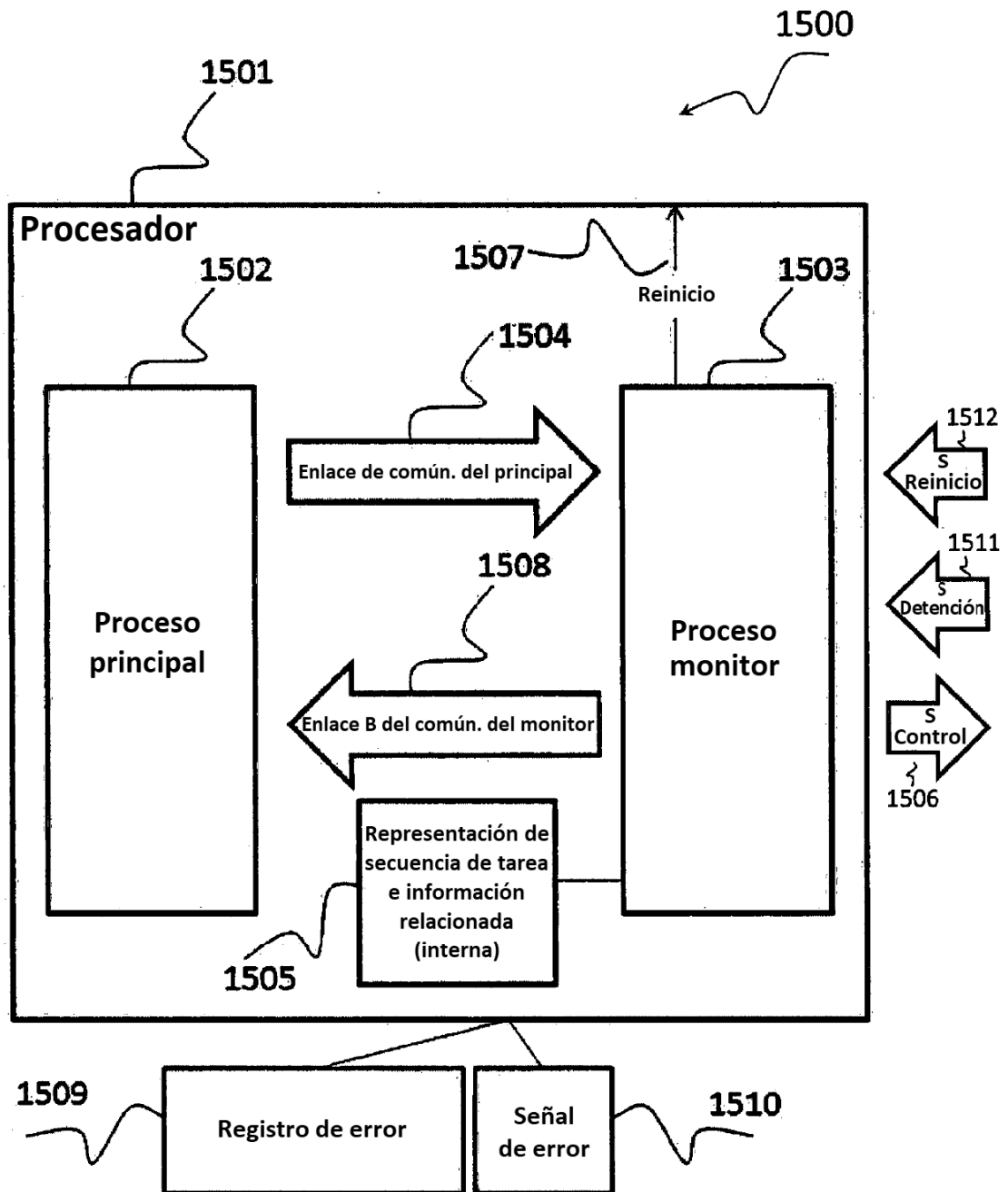


Figura 15

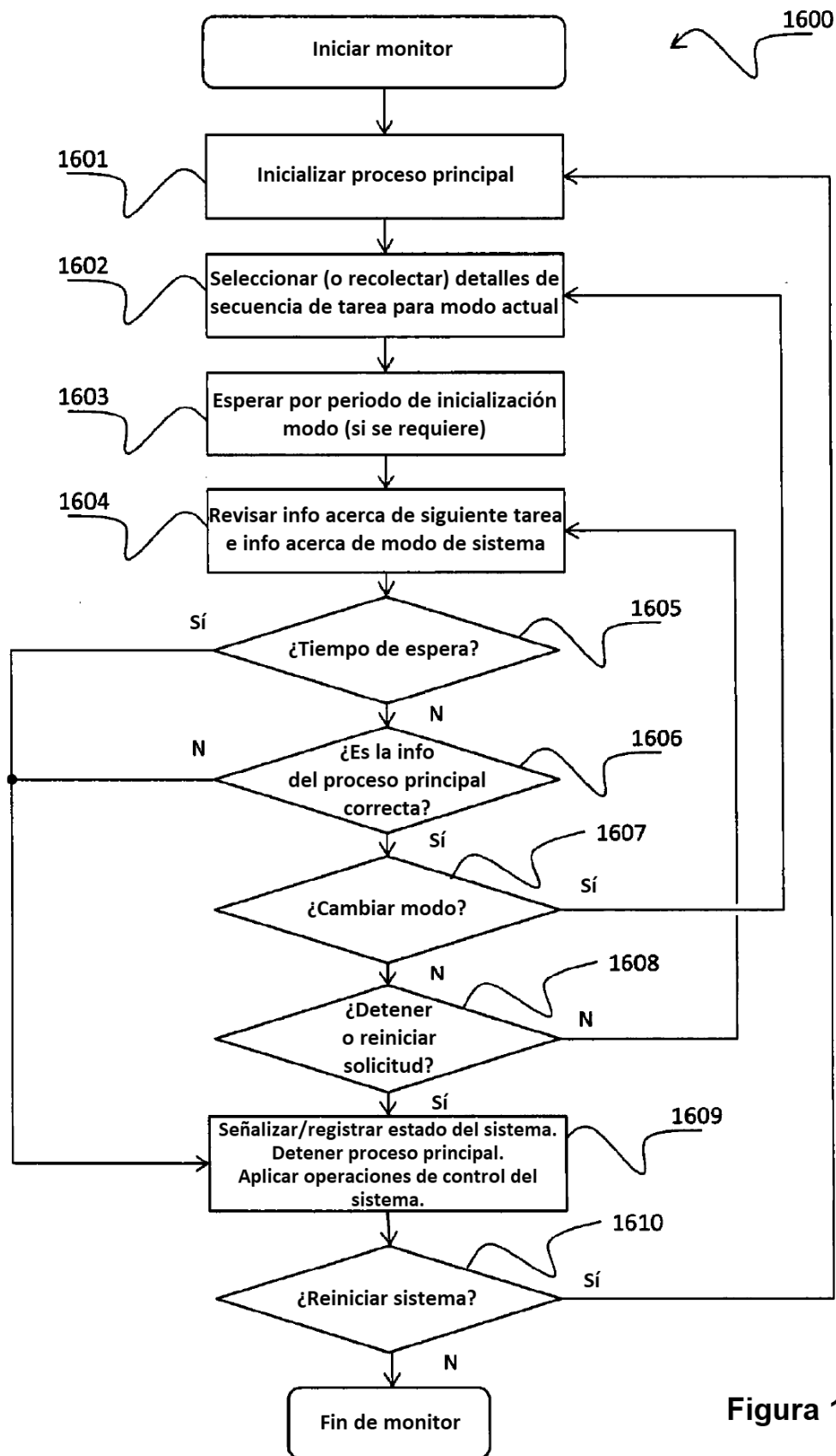


Figura 16