(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 414 435**

(51) Int. CI.:

G06F 21/00 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **29.10.2002 E 02790306 (1)**

(97) Fecha y número de publicación de la concesión europea: **10.04.2013 EP 1449084**

(54) Título: **Ejecución controlada de un programa por un soporte portátil de datos**

(30) Prioridad:

16.11.2001 DE 10156394

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
19.07.2013

(73) Titular/es:

GIESECKE & DEVRIENT GMBH (100.0%)
PRINZREGENTENSTRASSE 159
81677 MÜNCHEN, DE

(72) Inventor/es:

WEISS, DIETER

(74) Agente/Representante:

TORNER LASALLE, Elisabet

DESCRIPCIÓN

Ejecución controlada de un programa por un soporte portátil de datos.

La invención versa, en general, acerca del campo técnico de la ejecución de programas por un soporte portátil de datos que comprende una memoria de programas con al menos un núcleo de procesador para ejecutar este programa. Un soporte portátil de datos de este tipo puede ser, en particular, una tarjeta de chip en diversas formas o un módulo de chip. Más en particular, la invención versa acerca de la ejecución controlada de programas para detectar interferencia o ataques y para iniciar una gestión de errores adecuada.

En muchas técnicas de ataque a soportes portátiles de datos, la ejecución normal de los programas es alterada por influencias externas. Las interferencias de este tipo pueden ser causadas en particular por impulsos de tensión, el efecto del calor o el frío, campos eléctricos o magnéticos, ondas electromagnéticas o radiación de partículas. Por ejemplo, es posible manipular el contenido de los registros en el núcleo de un procesador mediante destellos de luz sobre el chip semiconductor, dejado al descubierto, del soporte portátil de datos. El contador de programa puede ser modificado potencialmente por tal ataque lumínico para hacer que el soporte portátil de datos dé salida a datos confidenciales de la memoria intermedia de entrada/salida en contra de la programación almacenada en el mismo.

Se conoce un dispositivo para controlar un motor de cierre de puertas en un vehículo por la solicitud de patente europea EP 1 056 012 A2 en el que un procesador de control ejecuta una pluralidad de subprogramas uno tras otro, incrementándose un contador tras cada subprograma. Tras la ejecución de los subprogramas se realiza una comprobación de si el estado del contador corresponde al número de subprogramas.

Sin embargo, este dispositivo conocido está relacionado únicamente con un sistema instalado permanentemente en el vehículo. No se considera un uso más general, en particular en un soporte portátil de datos. Además, en este dispositivo solo se proporciona una secuencia estrictamente secuencial de todos los subprogramas. Por lo tanto, la enseñanza conocida a raíz del documento EP 1 056 012 A2 no es adecuada para aplicaciones más complejas, en las que subprogramas individuales han de ser saltados de forma opcional.

Se conoce un procedimiento para monitorizar la ejecución de subrutinas por el documento US 5.732.272 A, sirviendo el procedimiento, en particular, para el seguimiento temporal y la evaluación del rendimiento, respectivamente. En principio, se monitoriza la ejecución de una subrutina realizando una "anotación de inicio" al inicio y realizando una "anotación de salida" al final de la ejecución del programa. En el caso de que ya no pueda ejecutarse correctamente la "anotación de salida", se proporcionan medidas de salvaguardia, que siguen garantizando el correcto registro de la ejecución de la subrutina. Debido a su complejidad, la solución conocida es de idoneidad limitada para sistemas que tienen recursos limitados.

El documento US 6.353.924 A describe un procedimiento para el seguimiento detallado de la ejecución de programas para detectar fallos. Se añaden respectivos elementos de "código de instrumentación" a los bloques individuales del programa que ha de ser objeto de seguimiento, comprendiendo los elementos de "código de instrumentación" información de identificación de bloques por medio de la cual se detecta la ejecución de bloque respectivo. La solución propuesta es adecuada solo en un grado limitado para determinar rápidamente tentativas de manipulación durante la ejecución del programa.

Por el documento WO 00/54155, se conoce un procedimiento para monitorizar la correcta ejecución de un programa por medio de una unidad de monitorización conectada entre una memoria de programas y un procesador. Se asigna un respectivo valor numérico a cada instrucción del programa, determinando y calculando la unidad de monitorización el valor numérico antes de la ejecución por el procesador. El valor de clave calculada de la respectiva operación previa se incluye en la operación de cálculo de clave calculada. Se compara un valor final característico resultante de clave calculada con un valor de referencia. El procedimiento es seguro, pero requiere que se lleven a cabo operaciones de cálculo comparativamente costosas para generar valores de clave calculada.

En consecuencia, el objeto de la presente invención es evitar, al menos parcialmente, los problemas de la técnica anterior y crear un procedimiento para la ejecución controlada de programas por un soporte portátil de datos, así como un soporte portátil de datos que proporcione una protección fiable contra manipulaciones del valor del contador de programa y que también sean adecuados para su uso en la ejecución de programas complejos. En particular, en las realizaciones preferentes ha de lograrse una alta protección con poco gasto.

Según la invención, este objeto se logra, total o parcialmente, mediante un procedimiento con las características de la reivindicación 1y con un soporte portátil de datos con las características de la reivindicación 10. Las reivindicaciones dependientes están relacionadas con realizaciones preferentes de la invención.

La invención parte del concepto básico de controlar la correcta ejecución de programas por medio de un contador de estado que es arrastrado durante la ejecución del programa. El programa tiene una pluralidad de secciones controladas cuyo procesamiento lleva en cada caso a un cambio en el contador de estado. No es preciso que las secciones controladas coincidan con subprogramas ni módulos de programa. La lectura del contador de estado se

compara con uno más valores admisibles en al menos un punto de comprobación del programa y, en caso de diferencia, tiene lugar una gestión de errores adecuada.

Según la invención se proporciona al menos la posibilidad de que el programa comprenda al menos una instrucción de salto, por la que se saltan y, por lo tanto, no se ejecutan una o más secciones controladas. En este caso, la lectura del contador de estado cambia en conjunción con la ejecución de la instrucción de salto, como si las secciones controladas se hubiesen ejecutado. El valor del contador de estado puede ser entonces comprobado en un punto posterior en el tiempo independientemente de la ejecución de la instrucción de salto. La flexibilidad en la configuración del programa que ha de ejecutarse puede así aumentar considerablemente sin que tengan que establecerse limitaciones con respecto a la seguridad.

El contador de estado puede estar configurado como un registro en el núcleo de procesador o como una zona de memoria en la memoria de trabajo del soporte portátil de datos. En las realizaciones preferentes de la invención el contador de estado se configura como un contador diferencial de estado cuyo valor resulta de la diferencia entre dos contadores implementados en términos de soporte físico que se denominan contador base y contador activo. Se impide así que el contador de estado adopte una única secuencia de valores durante la ejecución del programa conforme avanza. Por lo tanto, el uso de un contador diferencial de estado hace que resulte más difícil que el procedimiento usado sea objeto de espionaje y aumenta la seguridad del mismo. Preferentemente, el contador base es inicializado de manera regular, porque se configura a la respectiva lectura del contador activo.

En principio, la estructura del programa que ha de ejecutarse no está sujeta a ninguna limitación. Sin embargo, en configuraciones preferentes, que se contemplan en particular para tarjetas de chip o módulos de chip, la estructura básica del programa es un bucle que comprende una sección inicial, una pluralidad de secciones controladas y una sección de acción.

Preferentemente, la sección inicial incluye la lectura de entrada de una instrucción externa, las secciones controladas están relacionadas con el procesamiento de la instrucción, y en la sección de acción se emite una respuesta. En este caso, el punto de comprobación, o uno de los puntos de comprobación cuando hay una pluralidad de puntos de comprobación, está situado preferentemente entre la última sección controlada y la sección de acción, de modo que solo se emite la respuesta cuando la comprobación del contador de estado no ha dado como resultado alguna indicación de interferencia con la ejecución del programa. Una característica general de las realizaciones preferentes de la invención, con independencia de dicha estructura del programa, es proporcionar un punto de comprobación antes de cada acción del soporte portátil de datos orientada al exterior.

Preferentemente, el programa comprende uno o más módulos que han de ejecutarse uno tras otro cada uno de los cuales contiene, a su vez, al menos una sección controlada. En algunas realizaciones, cada módulo consiste en una sección controlada y, opcionalmente, en una comprobación de si los módulos subsiguientes han de ejecutarse o saltarse. En otras realizaciones, al menos algunos de los módulos contienen una pluralidad de secciones controladas.

Se logra un aumento adicional en el grado de monitorización si al menos uno de los módulos se configura como un módulo controlado con el menos un punto de comprobación interno al módulo. El valor del contador de estado, ya sea desde el inicio de la ejecución del programa o desde el inicio de la actual ejecución del bucle puede ser investigado en busca de su coincidencia con uno o más valores predeterminados en este punto de comprobación interno al módulo, o puede comprobarse el cambio en el contador de estado desde el inicio de la ejecución del módulo. En el segundo caso, la lectura del contador de estado o de un valor relacionado con el mismo, por ejemplo el estado del contador activo, se introduce preferentemente en memoria intermedia en un contador base local antes del primer cambio en el contador de estado del módulo.

Asimismo, en configuraciones adicionales, puede introducirse un contador local dentro de un módulo o dentro de una sección, monitorizando el contador local la correcta ejecución del programa dentro del módulo o dentro de la sección independientemente del contador global de estado. El contador local puede ser configurado según las técnicas descritas en lo que antecede en conjunción con el contador global de estado, siendo posible una construcción como un contador local diferencial.

Preferentemente, cada instrucción de salto a través de la que puedan saltarse secciones controladas está acoplada a correspondientes instrucciones para la adaptación del contador de estado. Sin embargo, también se proporcionan configuraciones de la invención en las que las instrucciones individuales de salto no están acompañadas por una adaptación del contador de estado. En este caso, se define habitualmente una pluralidad de estados admisible de contador en el punto de comprobación o los puntos de comprobación. Sin embargo, dado que la precisión de la monitorización se ve reducida por esta medida, es usada preferentemente solo para el mencionado contador local de estado.

El soporte portátil de datos según la invención se construye preferentemente como una tarjeta de chip o un módulo de chip. En configuraciones preferentes, tiene características correspondientes a las que acaban de describirse y/o características mencionadas en las reivindicaciones de procedimiento dependientes.

Características, ventajas y objetos adicionales de la invención resultan evidentes a partir de la siguiente descripción detallada de una pluralidad de realizaciones de muestra y de realizaciones alternativas. Se hace referencia a los dibujos esquemáticos, en los que:

la Fig. 1 es un diagrama de bloques de unidades funcionales de un soporte portátil de datos según una realización de la invención,

la Fig. 2 es un diagrama de flujo de la ejecución del programa de la realización de la Fig. 1,

la Fig. 3 es un diagrama de flujo de una variación de configuración de la Fig. 2,

la Fig. 4 es un diagrama de flujo de la ejecución del programa en un módulo controlado según una variación de configuración adicional de la Fig. 2, y

la Fig. 5 es un diagrama de flujo de la ejecución del programa en una sección que comprende una pluralidad de subsecciones controladas.

El soporte portátil 10 de datos ilustrado en la Fig. 1 está construido en la realización presente como una tarjeta de chip. Tiene una interfaz 12 para el intercambio de datos acotados, sin contacto o con él, con el entorno, un núcleo 14 de procesador con un contador 16 de programa y una memoria 18. Habitualmente, el núcleo 14 de procesador, la memoria 18 y los componentes de la interfaz 12 están integrados en un único chip de semiconductores.

La memoria 18 tiene una pluralidad de regiones configuradas en diferentes tecnologías de memoria. En la realización de la Fig. 1, estas son una memoria 20 de solo lectura configurada, por ejemplo, como una ROM (memoria de solo lectura) programada por máscara, una memoria no volátil 22 de lectura-escritura configurada en tecnología FLASH y una memoria volátil 24 de lectura-escritura configurada como RAM (memoria de acceso aleatorio). En términos de concepto, la memoria 18 contiene una memoria 26 de programas situadas en las regiones 20 y 22 y una memoria 28 de trabajo en las regiones 22 y 24.

La memoria 26 de programas contiene una pluralidad de programas, de los cuales uno se indica en la Fig. 1 y tiene el número de referencia 30. En la memoria 28 de trabajo se reservan ubicaciones de memoria para una pluralidad de contadores 32 y, más precisamente, en la realización de la Fig. 2, para un contador de estado ZZ, en la realización de la Fig. 3 para un contador activo AZ y un contador base BZ, en la modificación de la Fig. 4 para el contador de estado ZZ y un contador base local LBZ, y en la modificación de la Fig. 5 para el contador de estado ZZ y el contador LZ. En configuraciones alternativas, estos contadores también pueden ser configurados, total o parcialmente, como registros del núcleo 14 de procesador en vez de en la memoria 28 de trabajo, o estar almacenados en unidades adicionales del soporte portátil 10 de datos no mostradas en la Fig. 1.

En la presente realización, el contador 16 de programa del núcleo 14 de procesador puede ser influenciado por destellos de luz dirigidos sobre el chip de semiconductores. No obstante, para descartar que de ello resulte un ataque potencial contra los datos gestionados en el núcleo de procesador, el contador de estado ZZ en la ejecución del programa según la Fig. 2 se arrastra durante la ejecución del programa como un mecanismo de control independiente del contador 16 de programa y, en cada caso, comprueba la coherencia con un estado de contador predeterminado antes de ejecutar una acción dirigida al exterior.

En la realización de la Fig. 2, el programa 30 tiene la estructura de un bucle sin fin que comprende una sección inicial 34, una pluralidad de secciones controladas 36.1, 36.2, ..., 36.N designadas colectivamente en lo sucesivo en el presente documento como 36.x, un punto 38 de comprobación y una sección 40 de acción. Una instrucción que haya de ser procesada por el soporte portátil 10 de datos es leída de entrada, por ejemplo, a través de la interfaz 12 en la sección inicial 34. Las secciones controladas 36.x están relacionadas con etapas de procesamiento de la instrucción ejecutadas una tras otra en las que, por ejemplo, se verifica en primer lugar la validez de la instrucción; luego tiene lugar una comprobación de autorización; después tiene lugar una comprobación de la ejecutabilidad de la instrucción, por ejemplo mediante una comprobación de si los datos solicitados están presentes; y, por último, se ejecuta la instrucción de que, por ejemplo, los datos solicitados se escriban en una memoria intermedia de salida. En la presente realización se transmite una respuesta a través de la interfaz 12 en la sección 40 de acción y, en consecuencia, se señala ya sea la ejecución con éxito de la instrucción o el fallo en la ejecución de la instrucción.

Pueden ocurrir situaciones de error en las funciones de las secciones 36.x individualmente controladas citadas a título de ejemplo en lo que antecede que hacen superflua la ejecución de las secciones controladas 36.x subsiguientes en la ejecución del programa. Por ejemplo, si una instrucción no debe ser ejecutada por falta de autorización, las etapas ulteriores de procesamiento son superfluas. Por lo tanto, hay comprobaciones 42.1, 42.2, etc. de saltos de salida, designadas colectivamente 42.x en lo sucesivo, en las que se realiza una comprobación de si las secciones controladas 36.x que siguen en la secuencia prescrita han de ser ejecutadas o saltadas. Cada comprobación 42.x de salto de salida es una instrucción de salto condicional que comprueba el valor de una bandera de salto de salida. A su vez, la bandera de salto de salida está puesta por la respectiva sección controlada 36.x precedente para indicar que debe tener lugar un salto de salida. Por supuesto, una comprobación 42.x de salto de

salida solo precisa estar asociada con una sección controlada 36.x cuando el valor de la bandera de salta de salida pueda cambiar en esta sección controlada 36.x.

Para comprobar la ejecución del programa independientemente del estado del contador 16 de programa y, por lo tanto, poder reconocer un ataque lumínico, se usa el contador de estado ZZ. El contador de estado ZZ se configura a un valor inicial por medio de una instrucción 44 de inicialización en conjunción con la ejecución de la sección inicial 34. En la realización de la Fig. 2, este valor es cero. Cada sección controlada 36.x también contiene una instrucción 46.x en la que se altera el contador de estado ZZ, concretamente incrementado en una unidad en la presente realización. La instrucción se ejecuta entonces dentro de cada sección controlada 36.x solo cuando han concluido las operaciones que han de ser llevadas a cabo por la respectiva sección 36.x. Preferentemente, la instrucción 46.x es la última instrucción de cada sección controlada 36.x en cada caso.

Por lo tanto, en general el contador de estado ZZ, suponiendo que no tenga lugar ningún salto, contiene un valor durante la ejecución del programa correspondiente al número de secciones controladas 36.x ejecutadas. En la secuencia ejemplar de la Fig. 2, este número está designado con el valor N. Por lo tanto, tiene lugar una comprobación 48 en el punto 38 de comprobación en el que se verifica si la lectura del contador de estado ZZ tiene el único valor admisible, N. Si esto es así, se ejecuta la sección 40 de acción. Si el contador de estado ZZ tiene un valor diferente, la ejecución del programa se bifurca a una rutina 50 de gestión de errores.

Como gestión 50 de errores pueden tener lugar, por ejemplo, la reposición total del soporte portátil 10 de datos o, incluso, con una incidencia reiterada del error, el bloqueo del soporte portátil 10 de datos. También puede enviarse a través de la interfaz 12 un mensaje de error adecuado. En cualquier caso, la gestión 50 de errores debería garantizar que ningún dato del soporte portátil 10 de datos ya proporcionado potencialmente para su salida pase al exterior a través de la interfaz 12.

Según la secuencia descrita previamente, el contador de estado ZZ no tendría el valor admisible N si un salto de una o más secciones controladas 36.6 hubiese sido causado por una de las comprobaciones 42.x de salto de salida. Para garantizar el correcto funcionamiento del programa aun en este caso, se lleva a cabo una correspondiente instrucción 52.1, 52.2, designadas colectivamente en lo sucesivo 52.x, de corrección siempre que se salten secciones controladas 36.x debido a una comprobación 42.x de salto de salida. El contador de estado ZZ es alterado en cualquier instrucción 52.x de corrección como si las secciones controladas 36.x saltadas se hubiesen ejecutado. En el presente caso, el contador de estado ZZ aumenta en cada instrucción 52.x de corrección el número de secciones 36.x saltadas. Por lo tanto, el contador de estado ZZ tiene el único valor admisible N al llegar al punto 38 de comprobación aunque haya tenido lugar un salto de salida durante la ejecución del bucle.

En la presente realización, el contador de estado ZZ fue inicializado con el valor cero e incrementado una unidad durante el procesamiento de cada sección controlada. En configuraciones altemativas se proporcionan otros valores de inicialización y/u otros procedimientos para alterar el contador de estado. Por ejemplo, el contador de estado ZZ puede ser inicializado en la instrucción 44 con el valor 1 o N u otro valor. También es posible, en el caso de que, tras una comprobación 42.x de salto de salida, se salten secciones controladas 36.x, que el contador de estado ZZ siga contando sin corrección y, en vez de ello, disminuir el valor N con el número de secciones 36.x saltadas. Las instrucciones 52.1, 52.2 de corrección son, entonces, N := 1 y N := 2, etc. En las instrucciones 46.x puede realizarse cualquier operación matemática; por ejemplo, la resta de una unidad o el desplazamiento o la rotación bit a bit. Por supuesto, el valor admisible en la comprobación 48 debe adaptarse en consecuencia en estos casos.

La Fig. 3 muestra una variación adicional de la secuencia de la Fig. 2. El contador de estado ZZ no está aquí directamente implementado como un contador 32 en la memoria 28 de trabajo, sino que se proporcionan dos contadores 32 en la memoria 28 de trabajo, concretamente el contador activo AZ y el contador base BZ. Esta configuración del contador de estado ZZ como un contador diferencial evita la situación en la que cada estado del contador 16 de programa corresponde a valores fijos y siempre idénticos de los contadores 32 en la memoria 28 de trabajo.

La secuencia mostrada en la Fig. 3 comienza con la etapa 54 de inicialización, en la que el contador activo AZ se configura con un valor inicial, en la presente realización el valor cero. Tras la etapa 54 de inicialización, la estructura de programa ilustrada en la Fig. 3 es idéntica a la de la Fig. 2, por cuya razón en la Fig. 3se usan los mismos números de referencia que en la Fig. 2. Las diferencias consisten en que el contador base BZ se configura en la instrucción 44 con el respectivo estado del contador activo AZ. Por lo tanto, el estado del contador base BZ es la "marca cero" en cualquier bucle a partir de la que empieza a contar el contador activo AZ.

Según la Fig. 3, el contador activo AZ se cambia en las instrucciones 46.x y 52.x de la misma manera que el contador de estado ZZ en la Fig. 2. Por último, la diferencia entre el contador activo AZ y el contador base BZ es comparada en la comprobación 48 con el valor admisible N. Preferentemente, se evita una comparación con el valor siempre constante N en la programación real de la comprobación 48 porque, por ejemplo, inicialmente se calcula un resultado provisional BZ + N y este resultado provisional es comparado entonces con el valor del contador activo AZ.

El contador activo AZ no se repone de una ejecución del bucle a la siguiente, sino que la "marca cero", definida por el contador base BZ, meramente se establece de nuevo en la instrucción 44. Así, los valores de los contadores AZ y

BZ cambian de una ejecución del bucle a otra ejecución del bucle, de modo que se hace muy difícil espiar el procedimiento de la Fig. 3. En variaciones del procedimiento de la Fig. 3, el valor del contador activo AZ puede reponerse de manera arbitraria después de la comprobación 48. Los papeles de los contadores 32 en la memoria 28 de trabajo, que, respectivamente, forman el contador activo AZ y el contador base BZ, también pueden intercambiarse de vez en cuando. La Fig. 2 y la Fig. 3 muestran a título de ejemplo un módulo 56 que combina las dos secciones controladas 36.1 y 36.2 y las dos comprobaciones asignados 42.1 y 42.2 de salto. La Fig. 4 muestra este módulo 56 de la Fig. 2 en una variación adicional que comprende una comprobación local adicional del cambio en el contador de estado ZZ durante la ejecución del módulo 56. Según la Fig. 4, la lectura del contador de estado ZZ se guarda con este propósito al inicio de la ejecución del módulo en un contador base local LBZ.

Tras el procesamiento de las secciones controladas 36.1 y 36.2 y de las correspondientes comprobaciones 42.1 y 42.2 de salto, la ejecución del programa alcanza el punto 60 de comprobación interno al módulo. Aquí, en la comprobación 62 se verifica si el cambio en el contador de estado ZZ con respecto al valor LBZ guardado al inicio del módulo corresponde al número M de secciones controladas ejecutadas entretanto. En el presente ejemplo, M tiene el valor 2. Si el cambio interno al módulo en el contador de estado ZZ tiene el valor admisible, la ejecución del programa prosigue; si no, se ejecuta una rutina interna al módulo para la gestión 64 de errores.

También puede usarse la variación de configuración de la Fig. 4 en el procedimiento según la Fig. 3 con un contador de estado diferencial. En este caso, en la etapa 58 se almacena ya sea el valor del contador activo AZ o la diferencia AZ - BZ en el contador base local LBZ y la diferencia entre este valor o esta diferencia y el valor del contador base local LBZ se forma en la comprobación 62.

Aunque durante la comprobación del cambio interno al módulo en la Fig. 4 el contador de estado ZZ avanzaba, también es posible introducir un contador adicional local de estado dentro de un módulo 56 o dentro de una sección controlada 36.x. La Fig. 5 muestra una secuencia ejemplar de esta variación de configuración en la que una sección controlada 36.x de la Fig. 2 es protegida una vez más por un contador local de estado LZ.

Según la Fig. 5, el contador local de estado LZ se inicializa en la etapa 66 con el valor cero. Sigue la ejecución de una pluralidad, tres en el presente ejemplo, de subsecciones controladas 68.1, 68.2, 68.3, que son designadas colectivamente 68.6. El contador local LZ se incrementa en cada una de las subsecciones controladas 68.x en una respectiva instrucción 70.x. La segunda subsección controlada 68.2 comprende un salto de salida condicional, ilustrado en la Fig. 5 mediante una flecha discontinua, por medio del cual se saltan la etapa 70.2 de incremento y la tercera subsección 68.3.

Tras procesar la última subsección controlada 68.3, la ejecución del programa llega a un punto 72 de comprobación local. Con fines de comprobación, se ejecuta una comprobación 74 en la que el estado del contador local LZ se compara con los valores admisibles 1 y 3. Si hay coincidencia con uno de estos valores, la ejecución del programa prosigue con la instrucción 46.x para cambiar el contador de estado global ZZ; si no, tiene lugar una gestión 76 de errores.

Las anteriores variaciones, ya descritas en conjunción con el contador de estado ZZ, también son posibles para implementar el contador local LZ de la Fig. 5. En particular, el contador local LZ también puede formarse como un contador diferencial. Si el procedimiento de la Fig. 5 se usa en combinación con la configuración de la Fig. 3, solo tiene que adaptarse la asignación 46.x.

La Fig. 5 ilustra la comparación de un contador con una pluralidad de valores admisibles que es necesaria cuando se proporcionan diferentes trayectos de ejecución del programa sin las correspondientes instrucciones 52.x de corrección. En algunas realizaciones de la invención, también puede usarse una comparación de este tipo con una pluralidad de valores admisibles para el contador de estado global ZZ. Sin embargo, esta configuración reduce la fiabilidad del control y, por lo tanto, solo resulta preferible para contadores de estado locales LZ. Por el contrario, el salto de salida en la segunda subsección controlada 68.2 también puede ejecutarse en variantes adicionales mediante una instrucción adecuada de corrección. En este caso, en la comprobación 74 solo es admisible un único estado del contador local LZ, concretamente el estado 3 del contador.

REIVINDICACIONES

1. Un procedimiento para la ejecución controlada de un programa por un soporte portátil (10) de datos que comprende una memoria (26) de programas, un núcleo (14) de procesador y al menos un contador de estado (ZZ), en el que la memoria (26) de programas comprende al menos un programa (30) ejecutable por el núcleo (14) de procesador y con una pluralidad de secciones controladas (36.x) y en el que:

- la lectura del contador de estado (ZZ) se altera durante la ejecución de cada sección controlada (36.x) para reflejar el procesamiento de la respectiva sección controlada (36.x),

caracterizado porque

- durante la ejecución de una instrucción (42.x) de salto, como consecuencia de la cual se saltan una o más secciones controladas (36.x), se altera el contador de estado (ZZ) como si las secciones controladas (36.x) saltadas se hubiesen ejecutado, y

- la lectura del contador de estado (ZZ) se compara con al menos un valor admisible en al menos un punto (38, 60, 72) de comprobación, en la que, si hay coincidencia, prosigue la ejecución del programa y, en el caso de una diferencia, tiene lugar una gestión (50, 64, 76) de errores.

2. El procedimiento según la reivindicación 1 caracterizado porque el contador de estado (ZZ) está configurado como un contador diferencial de estado cuyo valor de contador corresponde a la diferencia entre los valores de un contador activo (AZ) y de un contador base (BZ).

3. El procedimiento según las reivindicaciones 1 o 2 caracterizado porque el programa (30) está configurado como un bucle que comprende una sección inicial (34), la pluralidad de secciones controladas (36.x) y una sección (40) de acción, situándose el punto (38) de comprobación o uno de los puntos de comprobación entre la última sección controlada (36.N) y la sección (40) de acción.

4. El procedimiento según las reivindicaciones 2 y 3 caracterizado porque el contador base (BZ) está configurado con el respectivo valor del contador activo (AZ) antes de la ejecución de las secciones controladas (36.x).

5. El procedimiento según cualquiera de las reivindicaciones 1 a 4 caracterizado porque el programa (30) comprende al menos un módulo (56) que contiene una o más de las secciones controladas (36.x).

6. El procedimiento según la reivindicación 5 caracterizado porque el módulo (56) es un módulo controlado en el que el cambio del contador de estado (ZZ) dentro del módulo (56) se compara en un punto (60) de comprobación con al menos un valor admisible.

7. El procedimiento según la reivindicación 6 caracterizado porque, antes del respectivo primer cambio del contador de estado (ZZ) en el módulo controlado (56), la lectura del contador de estado (ZZ) o de un valor relacionado con el mismo se introduce en memoria intermedia en un contador base local (LBZ).

8. El procedimiento según cualquiera de las reivindicaciones 1 a 7 caracterizado porque al menos una de las secciones controladas (36.x) comprende una pluralidad de subsecciones controladas (68.x), cambiando el valor de un contador local (LZ) durante la ejecución de cada subsección controlada (68.x) para reflejar el procesamiento de la respectiva subsección controlada (68.x), y comparándose el valor del contador local (LZ) en al menos un punto (72) de comprobación con al menos un valor admisible.

9. El procedimiento según cualquiera de las reivindicaciones 1 a 8 caracterizado porque el procedimiento se usa para repeler ataques lumínicos en los que el valor de un contador (16) de programa del núcleo (14) de procesador es manipulado por influencia de la luz.

10. Un soporte portátil (10) de datos, en particular una tarjeta de chip o un módulo de chip, que comprende una memoria (26) de programas, un núcleo (14) de procesador y al menos un contador de estado (ZZ), comprendiendo la memoria (26) de programas al menos un programa (30) ejecutable por el núcleo (14) de procesador y con una pluralidad de secciones controladas (36.x), caracterizado porque la memoria (26) de programas comprende instrucciones para hacer que el núcleo (14) de procesador ejecute un procedimiento según cualquiera de las reivindicaciones 1 a 9.

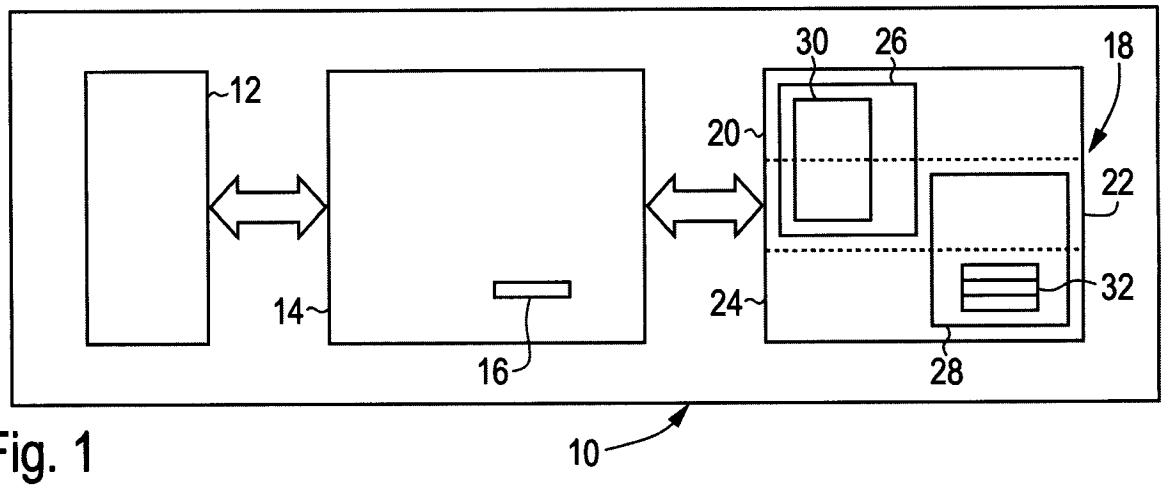
12
14
16
30
26
18
20
22
24
32
28
10

Fig. 1

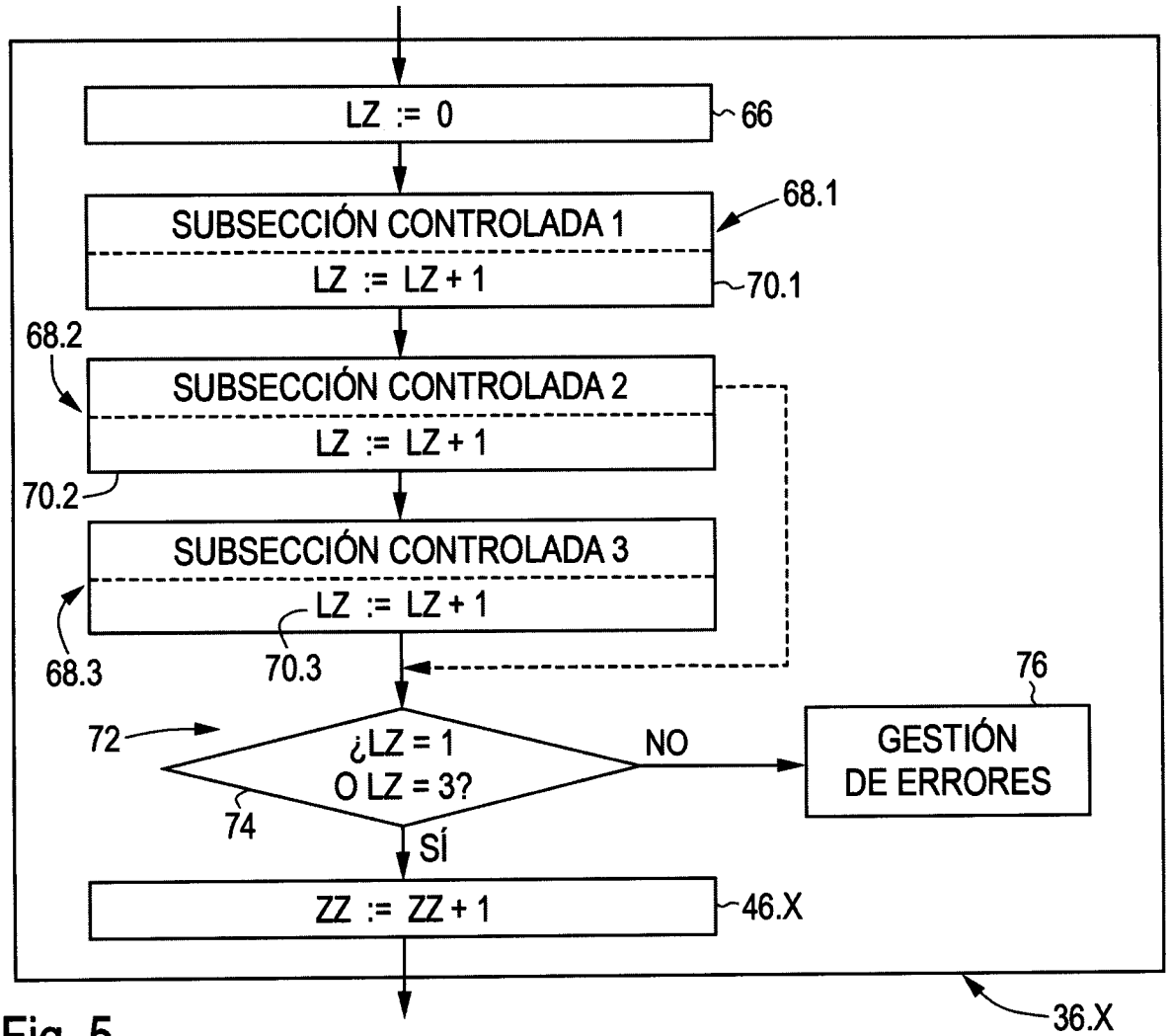
LZ := 0
66
SUBSECCIÓN CONTROLADA 1
LZ := LZ + 1
68.1
70.1
68.2
SUBSECCIÓN CONTROLADA 2
LZ := LZ + 1
70.2
SUBSECCIÓN CONTROLADA 3
LZ := LZ + 1
68.3
70.3
76
72
¿LZ = 1
O LZ = 3?
NO
GESTIÓN
DE ERRORES
74
SÍ
ZZ := ZZ + 1
46.X
36.X

Fig. 5

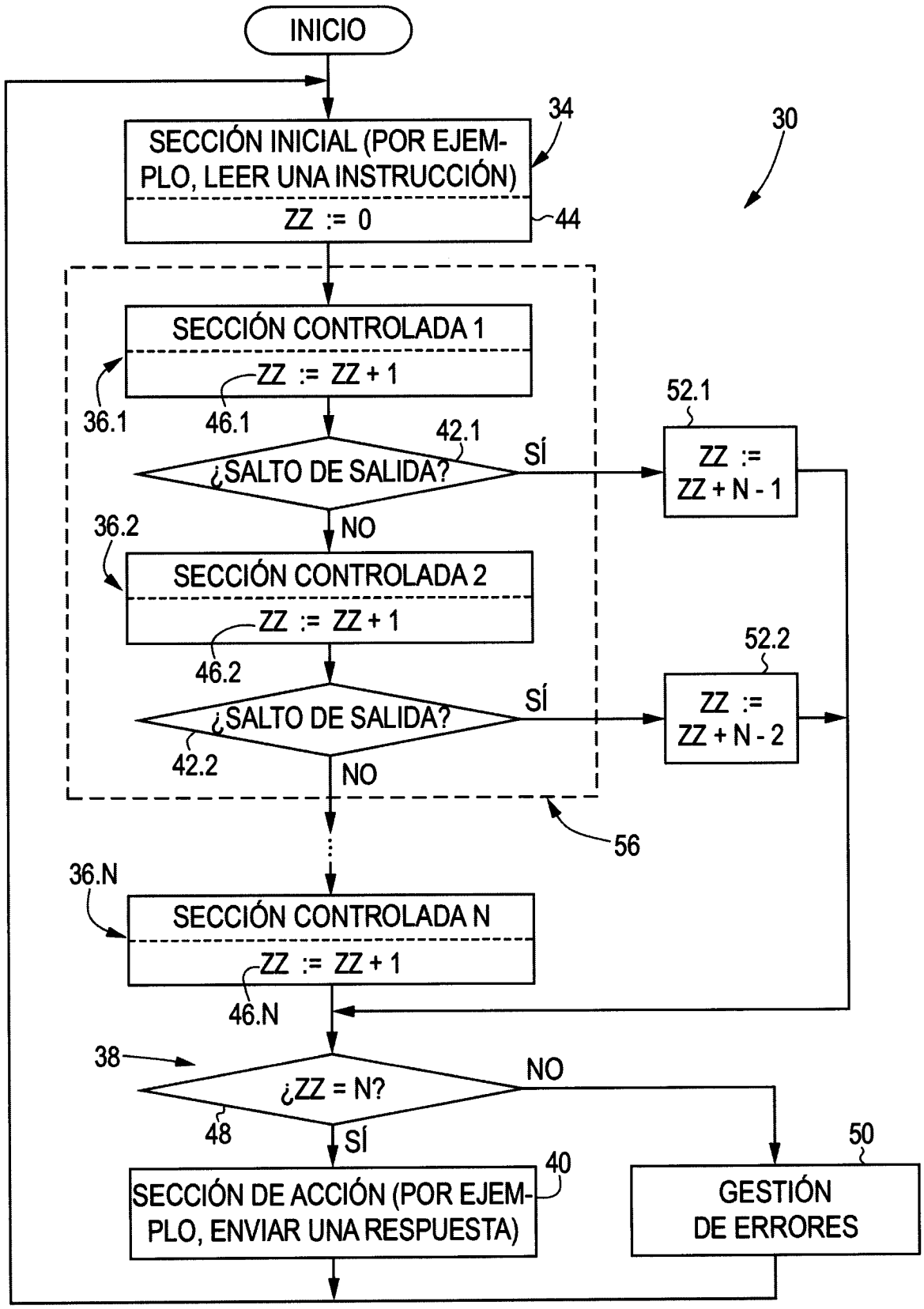
INICIO
34
30
SECCIÓN INICIAL (POR EJEMPLO, LEER UNA INSTRUCCIÓN)
ZZ := 0
44
SECCIÓN CONTROLADA 1
ZZ := ZZ + 1
36.1
46.1
42.1
¿SALTO DE SALIDA?
SÍ
52.1
ZZ := ZZ + N - 1
NO
36.2
SECCIÓN CONTROLADA 2
ZZ := ZZ + 1
46.2
52.2
¿SALTO DE SALIDA?
SÍ
ZZ := ZZ + N - 2
42.2
NO
56
36.N
SECCIÓN CONTROLADA N
ZZ := ZZ + 1
46.N
38
¿ZZ = N?
NO
48
SÍ
50
40
SECCIÓN DE ACCIÓN (POR EJEMPLO, ENVIAR UNA RESPUESTA)
GESTIÓN DE ERRORES

Fig. 2

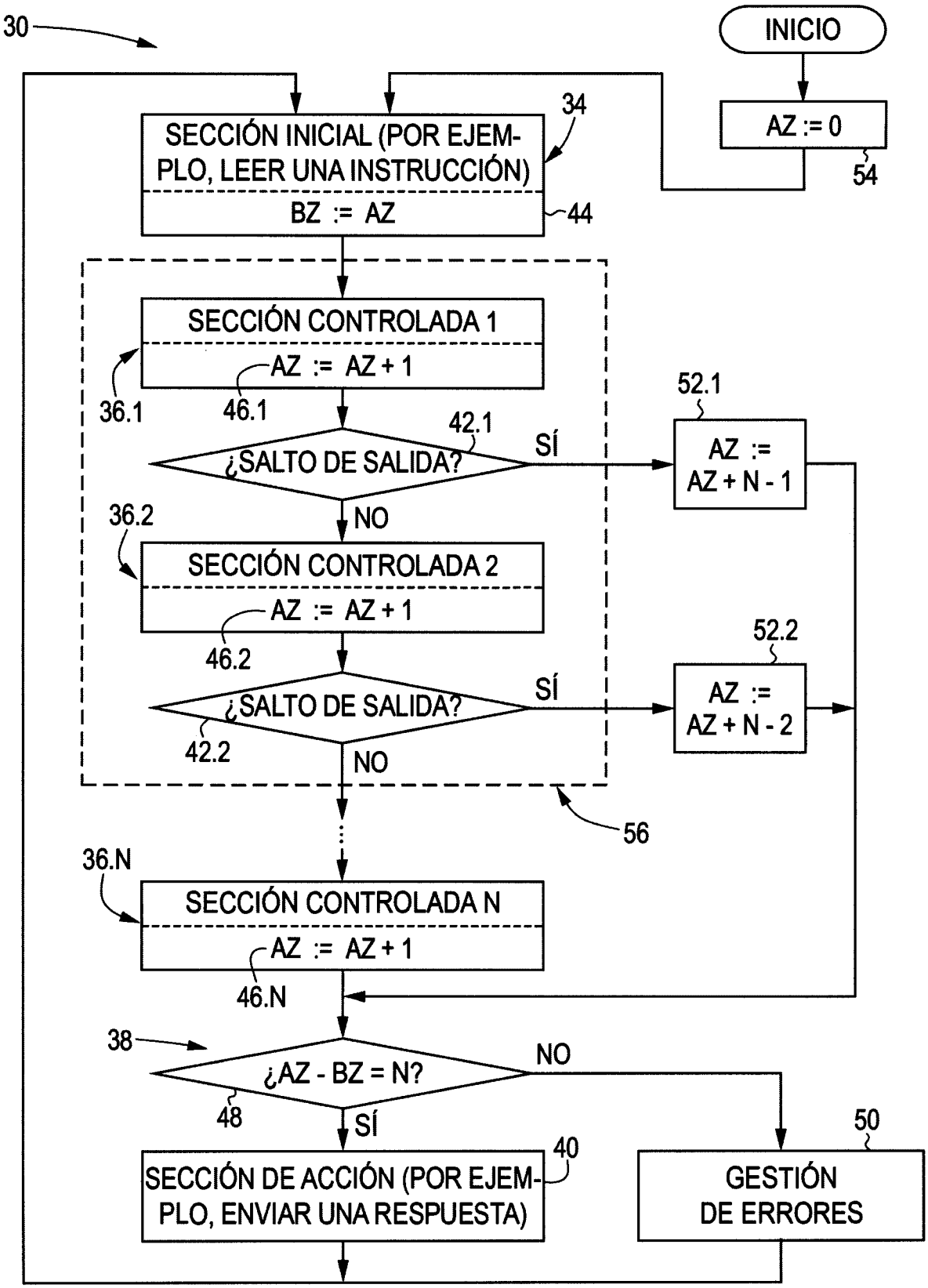
30
INICIO
AZ := 0
54
34
SECCIÓN INICIAL (POR EJEMPLO, LEER UNA INSTRUCCIÓN)
BZ := AZ
44
SECCIÓN CONTROLADA 1
AZ := AZ + 1
36.1
46.1
42.1
¿SALTO DE SALIDA?
SÍ
52.1
AZ := AZ + N - 1
NO
36.2
SECCIÓN CONTROLADA 2
AZ := AZ + 1
46.2
52.2
¿SALTO DE SALIDA?
SÍ
AZ := AZ + N - 2
42.2
NO
56
36.N
SECCIÓN CONTROLADA N
AZ := AZ + 1
46.N
38
¿AZ - BZ = N?
NO
48
SÍ
50
40
SECCIÓN DE ACCIÓN (POR EJEMPLO, ENVIAR UNA RESPUESTA)
GESTIÓN DE ERRORES

Fig. 3

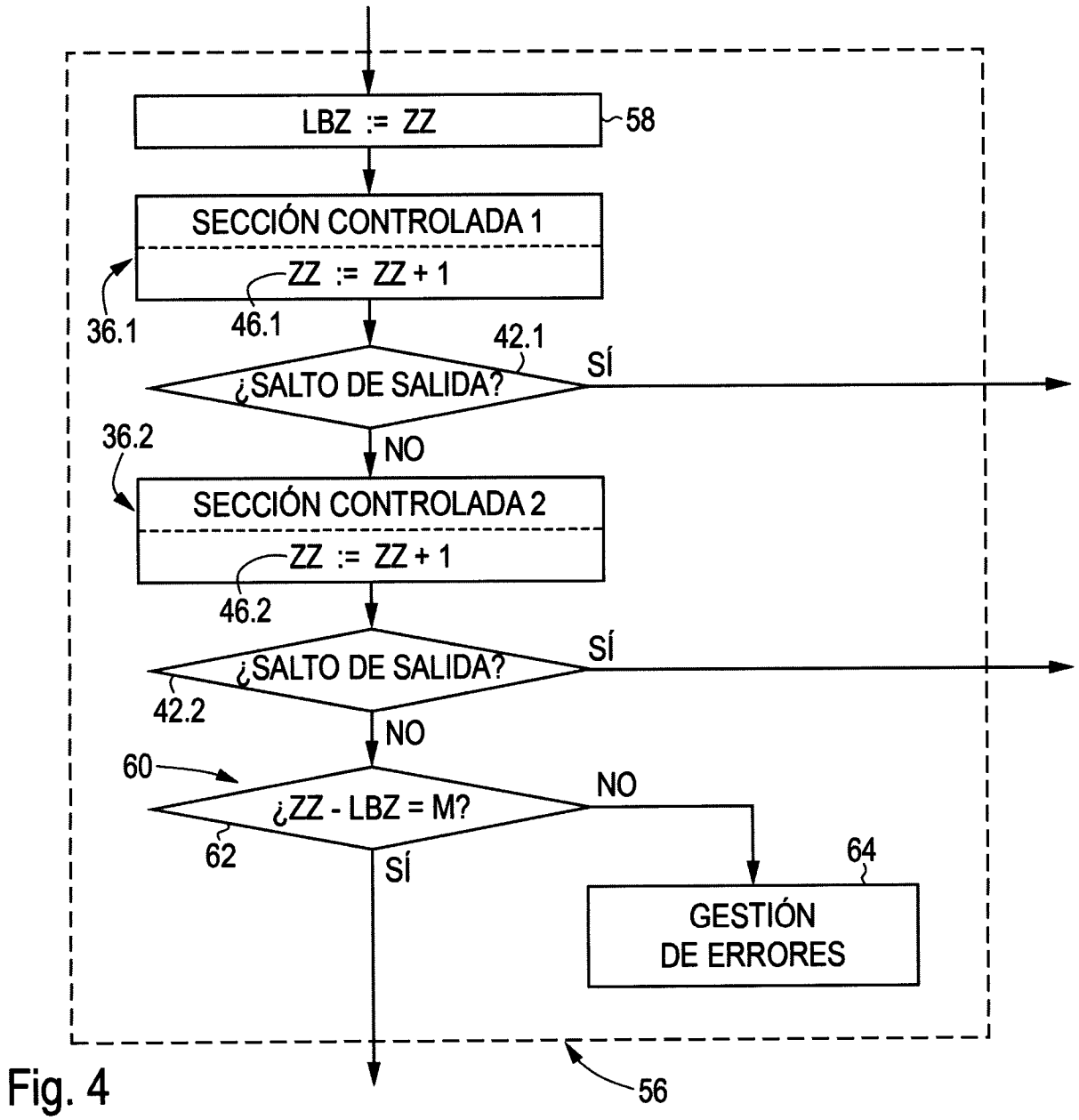
LBZ := ZZ
58
SECCIÓN CONTROLADA 1
ZZ := ZZ + 1
36.1
46.1
42.1
¿SALTO DE SALIDA?
SÍ
NO
36.2
SECCIÓN CONTROLADA 2
ZZ := ZZ + 1
46.2
¿SALTO DE SALIDA?
SÍ
42.2
NO
60
¿ZZ - LBZ = M?
NO
62
SÍ
64
GESTIÓN
DE ERRORES
56

Fig. 4