(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 708 499**

(51) Int. CI.:

G06F 9/44 (2008.01)
G06F 8/41 (2008.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **02.02.2006 E 10014002 (9)**

(97) Fecha y número de publicación de la concesión europea: **05.12.2018 EP 2330502**

(54) Título: **Dispositivo y método de soporte de la generación de código de programa, dispositivo y método de ejecución del programa, y dispositivo y método de procesamiento de la compresión del código de programa y programa para el mismo**

(30) Prioridad:

03.02.2005 JP 2005028122
03.02.2005 JP 2005028123

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
09.04.2019

(73) Titular/es:

MITSUBISHI DENKI KABUSHIKI KAISHA (100.0%)
7-3, Marunouchi 2-chome Chiyoda-ku
Tokyo 100-8310, JP

(72) Inventor/es:

ITO, TAKAHIRO;
SUZUKI, SHIGEKI;
OCHIAI, YOSHIKO;
KUSHIRO, NORIYUKI y
KOIZUMI, YOSHIAKI

(74) Agente/Representante:

ELZABURU, S.L.P

DESCRIPCIÓN

Dispositivo y método de soporte de la generación de código de programa, dispositivo y método de ejecución del programa, y dispositivo y método de procesamiento de la compresión del código de programa y programa para el mismo

Campo técnico

La presente invención se refiere a un dispositivo de soporte de la generación de código de programa para realizar el soporte para generar un código de programa y similar. En concreto, el dispositivo es para optimizar un código de programa de acuerdo con un dispositivo o similar que ejecuta y procesa el código de programa. También, la invención se relaciona con un dispositivo, método, y similar de ejecución del programa para realizar el proceso de ejecución en base al código de programa. Además, la invención se relaciona con un método y dispositivo de procesamiento de compresión del código de programa para realizar el dispositivo y el método.

Antecedentes de la técnica

En un electrodoméstico (un acondicionador de aire, un refrigerador, una lavadora, un dispositivo de iluminación de lámpara luminiscente, o similar), etc., para controlar el funcionamiento del mismo, se incorpora un dispositivo de ejecución del programa (de aquí en adelante tal como un electrodoméstico, un adaptador de electrodoméstico, o similar es referido como un aparato incorporado). En este momento, para que el dispositivo de ejecución del programa realice un proceso de ejecución, se requieren unos datos predeterminados lo que es llamado un código de programa. Este código de programa se almacena, por ejemplo, en medios de almacenamiento avanzados tales como un ROM (Memoria de Sólo Lectura) que se proporciona en el aparato incorporado, y el dispositivo de ejecución del programa lee (introduce) el código de programa almacenado en la ROM para realizar el proceso de ejecución. Después, se propone también un dispositivo para el soporte de la generación de dicho código de programa (por ejemplo, consulte el Documento 1 de Patente).

Además, existe un dispositivo de ejecución del programa para realizar un proceso de compresión del código de programa para reducir la cantidad de datos que deberían ser almacenados y para reducir la capacidad de almacenamiento de los medios de almacenamiento tales como la ROM tanto como sea posible, suprimiendo así los costes relacionados con los medios de almacenamiento (por ejemplo, consulte el Documento 2 de Patente). De esta manera, la reducción en la cantidad de datos del código de programa será, en los días venideros, conveniente para el caso en el que los aparatos incorporados estén mutuamente conectados a través de una red para enviar y recibir, por ejemplo, señales que contienen los datos del código de programa.

Documento 1 de Patente: Publicación de Solicitud de Patente No Examinada Japonesa Nº 9-16382.

Documento 2 de Patente: Publicación de Solicitud de Patente No Examinada Japonesa Nº 2002-318696.

Los artículos “A practical tool kit for making portable compilers”, de A.A. Tanenbaum et al., Communications of the ACM, septiembre 1983, y “An architecture for the direct execution of the Forth programming language” de J.R. Hayes et al., ASPLOS Proceedings, Agosto 1987, ACM, Nueva York, EE.UU., discuten aspectos de optimización de código de programa, incluyendo el uso de diccionarios para reemplazar un código con un código equivalente más compacto.

Descripción de la invención

Problemas a ser resueltos por la invención

En la presente memoria, los entornos en los que se usa el código de programa no son necesariamente uniformes y varían en gran medida dependiendo de, por ejemplo, el aparato incorporado (en concreto, el dispositivo de ejecución del programa). Por tanto, se desea generar un código de programa de acuerdo con una característica del aparato incorporado o similar. En concreto, si la cantidad de datos relacionada con el código de programa se puede establecer pequeña, por solo eso, es posible reducir la capacidad de almacenamiento de la ROM o de una memoria intermedia para el almacenamiento temporal en el momento de realizar el proceso de ejecución en el dispositivo de ejecución del programa. También, en los días venideros, es conveniente también dicha reducción en el caso en que los aparatos incorporados estén mutuamente conectados a través de la red para enviar y recibir, por ejemplo, las señales que contienen el código de programa. Sin embargo, el dispositivo de soporte descrito anteriormente simplemente genera el código de programa.

También, en los casos habituales, cuando el código de programa está comprimido, se disminuye la velocidad del proceso de ejecución del dispositivo de ejecución del programa. En vista de lo anterior, se adopta un método de acumulación de código de programa expandido en los medios de almacenamiento temporales (memorias intermedias) que están compuestos de una RAM (Memoria de Acceso Aleatorio) o similar. Por esta razón, se requiere una gran capacidad RAM.

Según aspectos de la presente invención se proporcionan un dispositivo de procesamiento de compresión de un código de programa, un método de procesamiento de compresión de un código de programa y un programa de un método de procesamiento de compresión de un código de programa de acuerdo con las reivindicaciones adjuntas.

[Fig. 1] Un diagrama que ilustra un sistema alrededor de un dispositivo de soporte de la generación de código de programa.

[Fig. 2] Un diagrama que ilustra un ejemplo de una pantalla cuyo medio 1 de entrada de especificación se presenta en el medio 20 de presentación.

[Fig. 3] Un diagrama que ilustra un ejemplo de un código de ejecución.

[Fig. 4] Diagramas que ilustran un ejemplo de una regla de optimización.

[Fig. 5] Diagramas que ilustran otro ejemplo de la regla de optimización.

[Fig. 6] Un diagrama que ilustra un dispositivo de soporte de la generación de código de programa según la Realización 2.

[Fig. 7] Un diagrama que ilustra un ejemplo de una pantalla que presenta un medio 6 de entrada de la condición del aparato incorporado.

[Fig. 8] Un diagrama que ilustra datos de ejecución comprimidos.

[Fig. 9] Un diagrama que ilustra un proceso de compresión del código de programa según la Realización 1.

[Fig. 10] Un diagrama que ilustra un dispositivo de ejecución del programa según una Realización 2 de la presente invención.

[Fig. 11] Un diagrama de flujo que describe las operaciones en el dispositivo de ejecución del programa.

[Fig. 12] Diagramas que ilustran un código de ejecución comprimido según la Realización 3.

Referencias numéricas

1 medio de entrada de especificación, 2 medio de generación de código, 3 medio de evaluación de código, 4, 4-1 medios de optimización de código, 4A unidad de análisis de código, 4B unidad de búsqueda de condiciones, 4C, 4D unidad de optimización, 4E unidad de discriminación de datos de proceso y comando, 4F unidad de conversión, 5 medio de salida, 6 medio de entrada de la condición del aparato incorporado, 10 medio de almacenamiento, 10A unidad de diccionario, 10B unidad de almacenamiento de reglas de optimización, 20 medio de presentación, 30 medio de entrada, 40 dispositivo de impresión, 50 dispositivo de lectura y escritura en memoria, 101 medio del proceso de control, 101A unidad de discriminación de datos de proceso y comando, 101B unidad de conversión, 102 medio de almacenamiento, 103, medio de comunicación, 121 medio de determinación del tipo de código, 122 medio de extensión de código, 123 medio de ejecución, 124 medio de almacenamiento del código de programa, 125 medio de almacenamiento de memoria intermedia.

Mejor manera de llevar a cabo la invención

Ejemplo 1

La Fig. 1 es un diagrama que ilustra un sistema alrededor de un dispositivo de soporte de la generación de código de programa según el ejemplo 1 de la presente invención. El dispositivo de soporte de la generación de código de programa está compuesto por el medio 1 de entrada de especificación, el medio 2 de generación de código, el medio 3 de evaluación de código, el medio 4 de optimización de código, el medio 5 de salida, el medio 10 de almacenamiento, el medio 20 de presentación, y el medio 30 de entrada.

La Fig. 2 es un diagrama que ilustra un ejemplo de la pantalla que presenta el medio 1 de entrada de especificación en el medio 20 de presentación. El medio 1 de entrada de especificación presenta, por ejemplo, una pantalla que permite a un operador introducir una especificación en el medio 20 de presentación, y procesar los datos que se introducen desde el medio 30 de entrada como la especificación. Los datos así procesados resultan datos procesados que constituyen un código de ejecución. En este momento, ECHONET (Conservación de Energía y Red de Asistencia Domiciliaria: ECHONET es una marca del consorcio ECHONET) se refiere a un estándar de una red (circuito de comunicación) para realizar un control mientras que se hace cooperar a los electrodomésticos los unos con los otros como se muestra en la Fig. 1. Según esta realización, se genera un código de programa para realizar un proceso de ejecución de electrodomésticos, adaptadores, y similares, que se conectan en base al estándar. La relación entre los elementos (objetos) de cada especificación y los datos de proceso se describirá más adelante.

La Fig. 3 es un diagrama que ilustra un ejemplo de un código de ejecución. El medio 2 de generación de código genera un código de programa en base a los datos procesados por el medio 1 de entrada de especificación. En la Fig. 3, se describe como datos de texto. El código de programa está compuesto de uno o una pluralidad de códigos de ejecución. El código de ejecución es una unidad en la que los procesos de ejecución se realizan de manera secuencial, por ejemplo, mediante un intérprete de un dispositivo de ejecución del programa que incluye el aparato incorporado. En esta realización, se usa Forth como el lenguaje que se usa para el código de programa. Forth es un lenguaje basado en la notación Polaca inversa en la que el análisis de la sintaxis es simple o innecesario. Según esta realización, el código de ejecución está compuesto por los datos de un comando (instrucción) que representan un proceso que el dispositivo de ejecución del programa debería ejecutar (de aquí en adelante referido como un comando; En Forth, referido como una palabra) y los datos para un proceso en base al comando (de aquí en adelante referido como datos de proceso), y se describe en el orden de los datos de proceso y el comando. En este momento, cada comando y datos de su contenido de proceso se definen (asocian) y registran (lo cual es llamado diccionario) anteriormente. Por esta razón, si datos tales como una cadena de caracteres descrita en el código de ejecución se registra en el diccionario, los datos son un comando, y si los datos no están registrados, los datos son datos de proceso. También, además de los comandos anteriormente determinados, se puede registrar el contenido del proceso del comando en el diccionario definiendo el contenido del proceso de comando. Según esta realización, este diccionario se almacena en el medio 10 de almacenamiento como una unidad 10A de diccionario. En este caso, el código de programa se genera con el uso de Forth, pero el código de programa al que la presente invención se puede aplicar no se limita a lo anterior.

Aquí, el medio 2 de generación de código genera un código de programa para la evaluación de modo que el operador pueda realizar fácilmente una evaluación cuando el código de programa se presenta en el medio 20 de presentación en un proceso mediante el medio 3 de soporte de la evaluación de código en una etapa posterior. Por ejemplo, se realiza la generación de modo que la cadena de caracteres del código de ejecución no sea demasiado larga, o que un comando básico, que no está definido de manera única, se use para la generación, facilitando así un seguimiento.

Se describe el comando RGST_EPC para ejecutar y procesar un registro para la información de propiedad del aparato incorporado. El id de objeto (id_obj) es un id añadido a un objeto del aparato incorporado. La propiedad ECHONET (epc) es un código de una propiedad (por ejemplo, un estado de establecimiento del volumen de aire en un acondicionador de aire o similar), que se puede establecer de acuerdo con el aparato incorporado. El valor del código es definido por el estándar ECHONET. También, se preparan 7 tipos para los tipos de datos (tipo) de la propiedad (determinados de manera automática cuando se determinan el aparato incorporado y la propiedad). La regla de acceso (regla) representa escribir, leer, y la autorización o desautorización de notificación. La disponibilidad o no disponibilidad del anuncio de cambio de estado (anno) es un índice (bandera) que indica si, cuando es cambiado el estado de la propiedad, se emite una notificación o no a través de la red. El tamaño de los datos (tamaño) es para definir un tamaño de datos de la propiedad, y el valor máximo del tamaño se establece como 8640 bytes en el estándar ECHONET.

El medio 3 de evaluación de código realiza el soporte para el operador para evaluar el código de programa (el código de ejecución) que es generado por el medio 2 de generación de código u otro medio. Por ejemplo, para que el operador evalúe el código de programa a través de la comprobación, depuración, o similar del código de programa así generado, el código de programa se presenta en la pantalla del medio 20 de presentación. Además, el dispositivo de ejecución del programa tiene un intérprete que se usa al realizar el proceso de ejecución, haciendo así posible realizar el proceso de ejecución del código de ejecución así generado. Como resultado, el operador puede comprobar el proceso de ejecución que se realiza en base al código de programa (el código de ejecución).

El medio 4 de optimización de código optimiza el código de programa en base a la regla de optimización almacenada en una unidad 10B de almacenamiento de reglas de optimización del medio 10 de almacenamiento. El medio 4 de optimización de código se compone de una unidad 4A de análisis de código, una unidad 4B de búsqueda de condiciones, y una unidad 4C de optimización. La regla de optimización es generada, por ejemplo, mediante el establecimiento de una regla de conversión dentro de los datos para que el dispositivo de ejecución del programa pueda realizar de manera efectiva el proceso de ejecución en base al código de programa, además de compilar una pluralidad de códigos de ejecución, generar un nuevo código de ejecución, etc. La regla de optimización se describirá de manera adicional más adelante.

La unidad 4A de análisis de código analiza el código de programa de entrada. En la presente memoria, la unidad 4A de análisis de código tiene un contador para contar el número consecutivo de códigos de ejecución en una parte en la que los códigos de ejecución que tienen el mismo comando para ejecutar y procesar el mismo contenido aparecen de manera consecutiva. La unidad 4B de búsqueda de condiciones realiza una búsqueda en base a la condición de conversión. Esto es, se determina si existe o no una parte que coincide con la condición de conversión mediante el cotejo del código de programa analizado con la condición de conversión que constituye la regla de optimización. En este momento, es referido el contador de la unidad 4A de análisis de código. La unidad 4C de optimización genera un nuevo código de ejecución (finalmente, un código de programa) con respecto a la parte determinada por la unidad 4B de búsqueda de condiciones que coincide con la condición de conversión.

En este momento, la unidad 4A de análisis de código, la unidad 4B de búsqueda de condiciones, y la unidad 4C de optimización pueden estar físicamente compuestas por medios independientes, pero en la presente memoria, por ejemplo, los procesos de las respectivas unidades se realizan mientras los procesos de los respectivos medios son ejecutados por un dispositivo de procesamiento de control basado en una CPU. Se debería observar que lo mismo aplica a la relación entre el medio 1 de entrada de especificación, el medio 2 de generación de código, el medio 3 de evaluación de código, el medio 4 de optimización de código, y el medio 5 de salida, y los respectivos medios pueden ser independientes los unos de los otros. Sin embargo, según esta realización, como el dispositivo de procesamiento de control ejecuta los procesos de los respectivos medios, se realizan los procesos del dispositivo de soporte de la generación de código de programa.

El medio 5 de salida convierte el código de programa en datos de un formato de acuerdo con el correspondiente dispositivo de salida y realiza un proceso para provocar que el dispositivo de salida emita. Por ejemplo, en caso de que el dispositivo de salida es un dispositivo 40 de impresión, el código de programa se convierte en datos que son impresos por el dispositivo de impresión en forma de código de barras, código QR (un código de dos dimensiones: el código QR es una marca registrada de DENSO WAVE INCORPORATED), o similar, y se realiza un proceso para imprimir los datos en un medio de impresión. Por esta razón, por ejemplo, sin enviar una señal proporcionando un costoso dispositivo de comunicación que es de uso exclusivo en el aparato incorporado y conectando directamente al dispositivo de comunicación, es posible almacenar el código de programa enviando una señal que incluye los datos al aparato incorporado con el uso de una cámara, que se proporciona en un teléfono móvil, de una tecnología de comunicación infrarroja, o similar. Como resultado, incluso cuando se usa un medio de bajo precio tal como un medio de papel, el código de programa almacenado en el aparato incorporado puede mantenerse actualizado. Además, por ejemplo, se puede usar un dispositivo 50 de lectura y escritura en memoria como dispositivo de salida para realizar el intercambio de datos a través de un medio de almacenamiento electrónico. También, el dispositivo de comunicación puede funcionar como dispositivo de salida. Se debería observar que el medio 10 de almacenamiento almacena, además del diccionario almacenado en la unidad 10A de diccionario y de la regla de optimización almacenada en la unidad 10B de almacenamiento de reglas de optimización, programas y similares para el soporte de la generación de código de programa a ser realizado por el medio respectivo, así como los datos para presentar una pantalla en el medio 20 de presentación.

Las Fig. 4 son diagramas que ilustran un ejemplo de una regla de optimización. Por ejemplo, como se muestra en la Fig. 4(a), la unidad 10B de almacenamiento de las reglas de optimización almacena una regla como la regla de optimización 'si la descripción del código de programa se hace para realizar de manera continua el proceso de ejecución del comando RGST_EPC (el registro de una propiedad ECHONET es el contenido del proceso) varias veces, el comando se sustituye por el comando RGST_EPCS con los contenidos del proceso en el que los registros de la pluralidad de propiedades ECHONET son ejecutadas y procesadas de una sola vez'. La Fig. 4(b) ilustra los contenidos como datos en un formato de texto (descripción). Un contenido antes de una flecha representa una condición de conversión en el momento en que la unidad 4B de búsqueda de condiciones realiza una búsqueda, y el contenido después de la flecha representa los contenidos de conversión en el momento en que la unidad 4C de optimización realiza la conversión (en la presente memoria, n es igual o mayor que 2 según el proceso es realizado varias veces. Esta n es contada por la unidad 4A de análisis de código). En un código de ejecución relacionado con el comando RGST_EPCS, con respecto a una propiedad ECHONET a ser registrada, los datos de proceso se hacen disponiendo conjuntos de id de objeto (id_obj), propiedad ECHONET (epc), tipo de datos (tipo), disponibilidad o no disponibilidad de anunciamiento de cambio de estado (anno) y tamaño de datos (tamaño) por el número de registros repetidos y asignando el número de repeticiones. Este comando RGST_EPCS se registra también en la unidad 10A de diccionario. De esta manera, los códigos de ejecución se compilan en otro código de ejecución para reducir la cantidad de datos del código de programa, logrando así la optimización. El proceso de ejecución se realiza mientras se compilan una pluralidad de códigos de ejecución, mediante lo cual es posible aumentar la velocidad de proceso.

Las FIG. 5 son diagramas que ilustran otro ejemplo de regla de optimización. Por ejemplo, en las Fig. 5, se almacena un regla 'una consecución de asignaciones con respecto a los diferentes elementos en la misma secuencia se sustituye por el comando ESTABLECER_C_CONJUNTO en donde se pueden asignar valores de manera colectiva' en la unidad 10B de almacenamiento de reglas de optimización. La Fig. 5(b) ilustra el contenido como datos en formato texto (descripción). En la presente memoria, carácter representa un valor asignado, c-dir representa una dirección de la parte asignada, y desplazamiento representa un desplazamiento a partir de la dirección inicial. [ ] esto se añade a la c-dir de la condición de conversión en el momento en que la unidad 4B de búsqueda de condiciones realiza una búsqueda y el contenido de la conversión en el momento en que la unidad 4C de optimización realiza una conversión representa que los contenidos de las c-dir (cadenas de caracteres) son los mismos. Además, n es igual o mayor que 2 y es contado por la unidad 4A de análisis de código. Por ejemplo, '1 MEM 0 + C! ' es un proceso de ejecución para asignar 1 en una parte en el byte número 0 a partir de la dirección MEM. En un código de ejecución relacionado con el comando ESTABLECER_C_CONJUNTO, los datos de proceso se hacen organizando los valores asignados (carácter) por n veces (el número de repeticiones), ordenando 'MEM' que representa una dirección y un desplazamiento de n veces, y asignando el número de repeticiones. También, se registra el comando ESTABLECER_C_CONJUNTO en la unidad 10A de diccionario.

Como se describe anteriormente, según la realización 1, la unidad 4A de análisis de código realiza el análisis en el código de programa generado por el medio 2 de generación de código además de en el medio 4 de optimización de

código, la unidad 4B de búsqueda de condiciones busca la parte que coincide con la condición de conversión en base a la regla de optimización almacenada en la unidad 10B de almacenamiento de reglas de optimización, y la unidad 4C de optimización realiza una conversión para realizar la optimización para la parte que coincide con la condición de conversión que es determinada por la unidad 4B de búsqueda de condiciones para generar el nuevo código de ejecución (el código de programa). Por tanto, el dispositivo de ejecución del programa que tiene el aparato incorporado puede realizar la optimización para facilitar el proceso de ejecución o similar. En concreto, se compila la parte en la que los códigos de ejecución basados en el mismo comando aparecen de manera consecutiva para generar el nuevo código de ejecución, y así se puede reducir la cantidad de datos del programa completo y los procesos se pueden realizar de manera colectiva al mismo tiempo en el proceso de ejecución así como, es posible aumentar la velocidad del proceso. En este momento, el código de programa relacionado a la generación por el medio 2 de generación de código se genera de tal manera que el operador realiza fácilmente la evaluación cuando el medio 3 de soporte de la evaluación de código presenta el código de programa en el medio 20 de presentación, mediante lo cual el operador puede realizar fácilmente la comprobación, corrección, y similar. Además, el código de programa es convertido en datos de acuerdo con el dispositivo de salida por el medio 5 de salida, y por lo tanto es posible registrar los códigos de programa correspondientes a los diversos formatos de los dispositivos de salida. En concreto, si se hace la conversión en los datos para imprimir el código de barras, el código QR, o para imprimir algo similar, el registro del código de programa se puede realizar en el medio de bajo precio. También, por ejemplo, los medios de comunicación generales tales como un teléfono móvil se pueden usar para enviar la señal que incluye el código de programa al aparato incorporado.

Ejemplo 2

La Fig. 6 es un dibujo que ilustra un dispositivo de soporte de la generación de código de programa según el Ejemplo 2 de la presente invención. El entorno en el que se mantiene (almacena) el código de programa en el aparato incorporado y el entorno en el que el dispositivo de ejecución del programa que incorpora el aparato ha realizado el proceso de ejecución en base al código de programa varía dependiendo del tipo de dispositivo. En concreto, para la generación del código de programa, una capacidad de almacenamiento de una ROM para mantener (almacenar) el código de programa y una capacidad de almacenamiento usable de una RAM (Memoria de Acceso Aleatorio) durante el proceso de ejecución o similar. En vista de lo anterior, según esta realización, se establece y se refleja una condición para el aparato incorporado (un entorno del proceso de ejecución basado en el código de programa) en el momento de la generación del código de programa. En la Fig. 6, una parte que tiene la misma referencia numérica que la del dispositivo de soporte de la generación de código de programa de la Fig. 1 realiza una operación de proceso similar a la del medio descrito en la realización 1, por lo que se omite esta explicación.

La Fig. 7 es un dibujo que ilustra un ejemplo de una pantalla que incorpora un medio 6 de entrada de la condición del aparato incorporado en el medio 20 de presentación. El medio 6 de entrada de la condición del aparato incorporado presenta, por ejemplo, una pantalla que permite a un operador introducir una característica, una condición, y similar del aparato incorporado en el medio 20 de presentación. El medio de entrada de la condición del aparato incorporado realiza un proceso de configuración de una señal de instrucción que es una entrada a partir del medio 30 de entrada y transmite la señal al medio 4 de optimización de código y al medio 5 de salida. En la Fig. 7, es posible establecer la capacidad de almacenamiento de la ROM para mantener el código de programa (capacidad de almacenamiento de código), la capacidad de almacenamiento usable de la RAM (capacidad RAM usable), si se realiza o no el proceso de compresión del código de programa (compresión binaria), si el código de programa se saca como medio de impresión o se saca como medio de almacenamiento electrónico tal como una tarjeta de memoria (medio de salida), y si se realiza o no la conversión en código QR (el código QR es una marca registrada de DENSO WAVE INCORPORATED) cuando se selecciona el medio de impresión (conversión en código QR).

El medio 4-1 de optimización de código según esta realización es diferente del medio 4 de optimización en que el medio 4-1 de optimización de código se proporciona con una unidad 4D de optimización para realizar, además del proceso de la unidad 4C de optimización, la determinación en base a las capacidades de almacenamiento de la ROM y de la RAM del aparato incorporado que son entrada en el medio 6 de entrada de la condición del aparato incorporado, en lugar de la unidad 4C de optimización. También, otra diferencia reside en que se proporciona una unidad 4E de discriminación de datos de proceso y comando y una unidad 4F de conversión para comprimir el código de programa. Según esta realización, la unidad 4E de discriminación de datos de proceso y comando y la unidad 4F de conversión se constituyen como parte de las funciones del medio 4-1 de optimización de código, pero se pueden constituir como un medio del proceso de compresión del código de programa independiente.

La unidad 4D de optimización realiza una conversión para realizar la optimización para la parte que coincide con la condición de conversión, que es determinada por la unidad 4B de búsqueda de condiciones, para generar un nuevo código de ejecución, pero en ese momento, si el número de códigos de ejecución a compilar resulta grande, la cantidad de datos de los datos de proceso resulta grande, y existe la posibilidad de que la cantidad de datos de un nuevo código de ejecución resulte grande. Por esta razón, dependiendo del aparato incorporado, cuando el dispositivo de ejecución del programa realiza el proceso de ejecución, la cantidad de datos puede exceder la capacidad de la RAM para almacenar de manera temporal los códigos de ejecución y similares. En vista de lo anterior en base a los datos sobre la capacidad de la RAM usable que se introducen a través del medio 6 de entrada

de la condición del aparato incorporado, cuando se determina que la cantidad de datos excede la capacidad, la unidad 4D de optimización genera una pluralidad de códigos de ejecución cuya cantidad de datos es igual o menor que la capacidad, de manera separada. También, cuando la cantidad de datos del código de programa completo excede la capacidad de almacenamiento de la ROM para mantener el código de programa que tiene el aparato incorporado, sin tener en consideración la presencia o ausencia de la configuración, la unidad 4E de discriminación de datos de proceso y comando y la unidad 4F de conversión son incitadas a realizar el proceso de compresión. Las determinaciones anteriormente descritas y similares se realizan para generar un nuevo código de programa.

La Fig. 8 es un diagrama que ilustra los datos de ejecución comprimidos. A continuación, el proceso de compresión se describirá en base al código de ejecución que se describe en la Fig. 3. En base a la entrada de la compresión binaria en el medio 6 de entrada de la condición del aparato incorporado y la determinación en la unidad 4D de optimización, cuando se realiza el proceso de compresión, en la unidad 4E de discriminación de datos y comando, se realiza un proceso de discriminación para los datos de proceso y los comandos de tal manera que los respectivos códigos de ejecución del código de programa se cotejan con el diccionario almacenado en la unidad 10A de diccionario para establecer las cadenas de caracteres, números, y similares, que no coinciden con el diccionario, como los datos de proceso y para establecer las cadenas de caracteres coincidentes (textos) como los comandos.

La unidad 4F de conversión representa respectivamente los datos de proceso y los comandos en datos binarios. En este momento, se asignan 2 bits para representar el id de objeto, 8 bits para la propiedad ECHONET (epc), 3 bits para el tipo de datos de la propiedad, 3 bits para la regla de acceso, 1 bit para la disponibilidad o no disponibilidad del anunciamiento del cambio de estado, y 14 bits para el tamaño de datos. Después de ser representados como datos binarios, los datos se generan mientras que se dividen para cada cantidad de datos predeterminada. En esta realización, se establecen 7 bits como la cantidad de datos predeterminada. Por ejemplo, cuando los datos de proceso tienen una cantidad de datos que no se puede dividir entre 7, por ejemplo, se rellena con “0” para el resto de los bits. Entonces, para los datos que representan los datos de proceso (de aquí en adelante estos datos se refieren también como datos de proceso), se añade un “0” como un bit inicial a los bits divididos, y para los datos para determinar el comando (de aquí en adelante estos datos se refieren también como comando) se añade un “1”. Esto es, esto resulta un bit de determinación (bandera) para determinar si estos son los datos de proceso o el comando. Como resultado cada uno de los códigos de ejecución (el código de programa) es dividido en códigos de byte en unidades de 1 byte (8 bits). De esta manera, las piezas de los datos de proceso divididas en la cadena de caracteres de cada objeto (elemento) se compilan y procesan como los datos binarios para realizar la reducción en la cantidad de datos.

Por otro lado, el comando es representado por los datos binarios correspondientes al comando con una correspondencia 1 a 1. Como resultado, se puede determinar el comando representado por los datos binarios. Se asocia y almacena una relación entre este comando y los datos binarios como datos en el medio 10 de almacenamiento, y la unidad 4F de conversión hace referencia a los datos en el momento de la generación. En la presente memoria, si el número de comandos es igual o menor que 128, 1 byte es suficiente para determinar el comando. Por ejemplo, en caso de que el número de comandos sea igual o mayor que este o similar, son necesarios 2 o más bytes. Sin embargo, incluso en tal caso, si resulta posible determinar el comando frecuentemente usado para (que aparezca en) el proceso de ejecución (por ejemplo, relacionado a la propiedad, etc.) por medio de sólo 1 byte de datos de código, la cantidad de datos del código de programa completo se puede reducir adicionalmente. Se debería observar que los datos de código en la unidad de byte que se genera a través del método de esta realización se pueden comprimir adicionalmente a través de un método de compresión de secuencias de bits conocido tal como un método de longitud de ejecución.

También, en base a los datos del medio de salida que se introducen a través del medio 6 de entrada de la condición del aparato incorporado, el medio 5 de salida determina un método para realizar la conversión. En este momento, cuando el medio de almacenamiento electrónico se selecciona como el medio de salida, la conversión a código QR se selecciona de manera automática como ‘a no realizar’. Entonces, cuando el medio de impresión se selecciona como el medio de salida, la conversión a código QR se selecciona como ‘a realizar’.

En el dispositivo de ejecución del programa que tiene el aparato incorporado, a través del medio de almacenamiento o similar, por ejemplo, se mantiene un código de programa de ROM. El código de programa es leído en 1 byte, se determina si el código son los datos que contienen el comando o los datos que contienen los datos de proceso en base al bit de determinación, y en caso de que los datos contengan los datos de proceso, los datos se almacenan de manera temporal en la RAM. Cuando se determinan como los datos que contienen el comando, los datos se extienden al código de ejecución para realizar el proceso de ejecución.

Como se describe anteriormente, según el Ejemplo 2, con la provisión del medio 6 de entrada de la condición del aparato incorporado para realizar la configuración de entrada del código de programa del entorno del proceso de ejecución (la capacidad de la ROM, la RAM, o similar) del aparato incorporado (el dispositivo de ejecución del programa que tiene el aparato incorporado), en base a la configuración, la unidad 4D de optimización genera además el nuevo código de programa, a través del cual es posible realizar la generación detallada del código de programa de acuerdo con la característica, la condición, y similar del aparato incorporado. Entonces, con la provisión de la unidad 4E de discriminación de datos de proceso y comando y de la unidad 4F de conversión, se puede

realizar el proceso de compresión del código de programa, mediante el cual se puede lograr la reducción en la capacidad de almacenamiento de la ROM, y se pueden lograr menores costes. Entonces, en el proceso de compresión, los datos de proceso y el comando se discriminan los unos del otro en cada uno de los códigos de ejecución para dividir los respectivos datos en los 7 bits tal como se compilan las piezas de los datos de proceso en términos de cada elemento (objeto) dentro de los datos binarios y para los comandos, los números correspondientes se establecen como los datos binarios, y el bit de determinación para determinar si los datos contienen los datos de proceso o el comando se añade a cada dato para dividir los datos en datos de 1 byte cada uno, mediante lo cual tanto los datos de proceso como los comandos se pueden comprimir.

Ejemplo 3

La Fig. 9 es un diagrama que ilustra un proceso de compresión del código de programa según el Ejemplo 3 de la presente invención. La parte superior de la Fig. 9 ilustra un ejemplo de código de ejecución. Aquí, es descrito como datos de texto. El código de programa está compuesto de uno o una pluralidad de códigos de ejecución. Primero, se describirá el código de ejecución. Según esta realización también, se usa Forth como el lenguaje usado para el código de programa, que es un código para realizar el proceso de ejecución en el estándar ECHONET. Según esta realización, el medio 102 de almacenamiento como se describirá más adelante almacena un diccionario.

Como se describe en el Ejemplo 1 también, se describe el comando RGST_EPC para ejecutar y procesar un registro de la información de la propiedad del aparato incorporado. El id de objeto es un id añadido a un objeto del aparato incorporado. Se asignan 2 bits para representar el id de objeto. La propiedad ECHONET es un código de una propiedad que se puede establecer de acuerdo con el tipo de aparato incorporado (por ejemplo, un estado de establecimiento del volumen de aire en un acondicionador de aire o similar). Un valor del código se define mediante el estándar ECHONET, y se asignan 8 bits. También, en la presente memoria, se preparan 7 tipos como los tipos de datos de propiedad, y se asignan 3 bits. La regla de acceso representa escritura (Establecer), lectura (Obtener) y la autorización o desautorización de notificaciones, y se asignan 3 bits. La disponibilidad o no disponibilidad del anuncio de cambio de estado es un índice (bandera) que indica si, cuando se cambia el estado de la propiedad, se emite una notificación o no a través de la red, y se asigna 1 bit. El tamaño de los datos es para definir el tamaño de datos de la propiedad, el valor máximo del tamaño se establece como 8640 bytes en el estándar ECHONET, y se asignan 14 bits para representar el valor numérico.

A continuación, se dará una descripción de una configuración y un proceso del dispositivo del proceso de compresión del código de programa. El dispositivo de compresión del código de programa según esta realización está compuesto por el medio 101 del proceso de control, el medio 102 de almacenamiento, y el medio 103 de comunicación. El medio 101 del proceso de control incluye una unidad 101A de discriminación de datos de proceso y comando y una unidad 101B de conversión. En el medio 101 del proceso de control, con la unidad 101A de discriminación de datos de proceso y comando, cada uno de los códigos de ejecución del código de programa coteja con el diccionario almacenado en el medio 102 de almacenamiento para establecer las cadenas de caracteres, los números, y similares, que no coinciden con el diccionario, como los datos de proceso y para establecer las cadenas de caracteres (texto) coincidentes como el comando, realizando así el proceso de discriminación entre los datos de proceso y el comando. En este momento, por ejemplo, en base a una instrucción, dato, y similar, que es introducido por el operador desde el medio de entrada, el código de ejecución puede ser generado por la unidad de generación de código de programa (no mostrada en el dibujo) proporcionada en una etapa anterior de la unidad 101A de discriminación de datos de proceso y comando del medio 101 del proceso de control o se puede generar en otro dispositivo y se puede procesar el código de ejecución contenido en la señal que se envía a través de un circuito de comunicación (no necesariamente relacionada con ECHONET).

La unidad 101B de conversión expresa los datos de proceso y el comando en datos de formato binario (de aquí en adelante referidos como datos binarios) y genera los datos que se dividen en una cantidad de datos predeterminada. En este ejemplo, se establecen 7 bits como la cantidad de datos predeterminada. Por ejemplo, cuando los datos de proceso tienen una cantidad de datos que no se puede dividir entre 7, por ejemplo, se rellena con “0” para el resto de bits. Entonces, para los datos que representan los datos de proceso (de aquí en adelante estos datos son referidos también como datos de proceso), se añade un “0” como un bit inicial a los bits divididos, y para los datos para determinar el comando (de aquí en adelante estos datos se refieren también como comando) se añade un “1”. Esto es, esto resulta un bit de determinación (bandera) para determinar si estos son los datos de proceso o el comando. Como resultado, como se muestra en la parte inferior de la Fig. 9, cada uno de los códigos de ejecución (el código de programa) es dividido en códigos de byte en unidades de 1 byte (8 bits). De esta manera, las piezas de los datos de proceso divididas en las cadenas de caracteres de cada objeto (elemento) se compilan como los datos binarios y se realiza un proceso de codificación del mismo para realizar la reducción en la cantidad de datos.

Por otro lado, el comando es representado por los datos binarios correspondientes al comando con una correspondencia 1 a 1. Como resultado, se puede determinar el comando representado por los datos binarios. Se asocia una relación entre este comando y los datos binarios y se almacena como datos en el medio 102 de almacenamiento, y la unidad 101B de conversión se refiere a los datos en el momento de la generación. En la presente memoria, si el número de comandos es igual o menor que 128, 1 byte es suficiente para determinar el comando. Por ejemplo, en caso de que el número de comandos sea igual o mayor que este o similar, son

necesarios 2 o más bytes. Sin embargo, incluso en tal caso, si resulta posible determinar el comando frecuentemente usado para (que aparezca en) el proceso de ejecución (por ejemplo, relacionado a la propiedad, etc.) por medio de sólo datos de código de 1 byte, la cantidad de datos del código de programa completo se puede reducir adicionalmente. Se debería observar que los datos de código en la unidad de byte que se genera a través del método de esta realización se pueden comprimir adicionalmente a través de un método de compresión de secuencias de bits conocidas tal como un método de longitud de ejecución.

El código de programa comprimido que se genera de la manera anteriormente descrita se almacena en el medio 124 de almacenamiento del código de programa del aparato incorporado según la Realización 4 como se describirá a través del medio 103 de comunicación o un circuito de comunicación por cable o inalámbrico, por ejemplo, en respuesta a la solicitud de descarga enviada desde el lado del aparato incorporado a través de la red o por medio de una distribución activa desde el lado del dispositivo del proceso de compresión del código de programa (el medio 101 del proceso de control).

Como se describe anteriormente, según el Ejemplo 3, los datos de proceso y el comando se discriminan los unos del otro en cada uno de los códigos de ejecución para dividir los respectivos datos en una unidad predeterminada (7 bits, según esta realización), dichas piezas de los datos de proceso se compilan en términos de cada elemento (objeto) en datos binarios y para los comandos, se establecen los números correspondientes como los datos binarios, y el bit de determinación para determinar si los datos contienen datos de proceso o el comando se añade a cada dato para dividir los datos en datos de 1 byte, mediante lo cual se pueden comprimir tanto los datos de proceso como los comandos. Entonces, en el momento de realizar el proceso de ejecución, se realiza la lectura para cada byte, y es posible determinar de manera fácil si los datos incluyen datos de proceso o el comando. También, según se establece el código de programa con el uso de un lenguaje basado en la notación Polaca inversa tal como el Forth, el código de ejecución está compuesto en el orden de datos de proceso y comando, mediante lo cual es posible discriminar entre los datos de proceso y el comando fácilmente. Además, estableciendo la unidad para dividir el código de programa como de 1 byte, por ejemplo, dicho dispositivo de procesamiento tal como un micro ordenador de un chip puede hacer frente a la realización del proceso de ejecución. Entonces, por ejemplo, mediante la realización de un ajuste de acuerdo con la frecuencia tal como el descenso del número de bytes que contienen un comando frecuentemente usado (que aparezca) en el proceso de ejecución, es posible mejorar más el efecto de la compresión.

Realización 1

La Fig. 10 es un dibujo que ilustra un dispositivo de ejecución del programa según la realización 1 de la presente invención. En esta realización, el dispositivo de ejecución del programa está compuesto de un medio 121 de determinación del tipo de código, un medio 122 de extensión del código, un medio 123 de ejecución, un medio 124 de almacenamiento del código de programa, y un medio 125 de almacenamiento de memoria intermedia. Entonces, se realiza el proceso de ejecución basado en el código de programa que se genera en la realización 3.

El medio 121 de determinación del tipo de código lee (introduce) el código de programa comprimido (codificado) almacenado en el medio 124 de almacenamiento del código de programa en una cantidad de datos predeterminada. Según esta realización, en conformidad con la realización 3, la cantidad de datos predeterminada se establece como de 1 byte (8 bits). Entonces, se determina si los datos leídos son datos que contienen datos de proceso (bytes de datos. Datos que no contienen ningún comando) o datos que contienen un comando (byte de comando). El medio 121 de determinación del tipo de código realiza la determinación en base al bit de determinación (bandera) descrito en el Ejemplo 3. Entonces, cuando se determinan como datos de proceso, los datos de proceso se almacenan en el medio 125 de almacenamiento de memoria intermedia, y cuando se determina como el comando, el control del proceso se pasa al medio 122 de extensión de código.

Cuando el medio 121 de determinación del tipo de código determina los datos como el comando, el medio 122 de extensión de código genera (extiende, decodifica) el código de ejecución en base a los datos de proceso almacenados en el medio 125 de memoria intermedia y el comando. Cuando se genera el código de ejecución, el control del proceso se pasa al medio 123 de ejecución.

El medio 123 de ejecución realiza el proceso de ejecución en base al código de ejecución generado por el medio 122 de extensión de código. En este momento, según esta realización, el medio 123 de ejecución es un intérprete para realizar el proceso de ejecución del comando con respecto a los datos de proceso mientras analiza (traduce) el código de ejecución. Por lo tanto, el código de programa se ejecuta y procesa de manera secuencial para cada código de ejecución. En este momento, el medio 123 de ejecución tiene el diccionario descrito en el ejemplo 3 en el medio de almacenamiento (no mostrado en el dibujo) para analizar el código de ejecución. Cuando se finaliza el proceso de ejecución, el control del proceso se pasa al medio 121 de determinación del tipo de código.

El medio 124 de almacenamiento del código de programa y el medio 125 de almacenamiento de memoria intermedia son ambos medios de almacenamiento. En este momento, según esta realización, el medio 124 de almacenamiento del código de programa es una memoria no volátil, pero no es simplemente una ROM, y es un medio de almacenamiento (por ejemplo, una EPROM, una EEPROM, o similar) en el que se puede realizar la reescritura del código de programa mediante un añadido, un cambio parcial o completo, una eliminación, etc. En el caso de la

reescritura del código de programa, como se ha descrito en la realización 1, por ejemplo, mediante la descarga a través de la red, la señal que contiene el código de programa es recibida por el medio de comunicación (no mostrado en el dibujo) que tiene el aparato incorporado, y se almacena y mantiene en el medio 124 de almacenamiento del código de programa.

Este medio 125 de almacenamiento de memoria intermedia es, por ejemplo, una memoria volátil. El medio 125 de almacenamiento de memoria intermedia es un medio de almacenamiento temporal para almacenar al menos los datos de proceso hasta que el medio 122 de extensión del código genere el código de ejecución, y no deja nada si se introducen los siguientes datos de proceso. Por lo tanto, la capacidad de almacenamiento que el medio 125 de almacenamiento de memoria intermedia tiene es arbitraria, pero preferiblemente, se establece como la capacidad de almacenamiento máxima que es necesaria para almacenar la parte de los datos de proceso (o el código de ejecución) del código de ejecución ejecutados y procesados por el aparato incorporado (el dispositivo de ejecución del programa) para un momento.

Según esta realización, los datos del código de programa comprimido a ser ejecutados y procesados se leen en un byte cada uno, y se realiza la determinación de si los datos contenidos en el mismo son los datos de proceso o el comando. En el caso de los datos de proceso, los datos se almacenan en el medio 125 de almacenamiento de memoria intermedia, y cuando se determina como el comando, el medio 122 de extensión pone los datos de proceso almacenados en el medio 125 de almacenamiento de memoria intermedia y el comando en el código de ejecución. El medio 123 de ejecución ejecuta y procesa el código de ejecución. Como resultado, en base al código de programa mantenido en el medio 124 de almacenamiento del código de programa por el dispositivo de ejecución del programa, se realiza el proceso de ejecución para cada código de ejecución. En este momento, el código de ejecución (el código de programa) que se ha sometido al proceso de ejecución no se deja de manera descuidada en un medio 125 de almacenamiento de memoria interna tal como la RAM, y la capacidad de almacenamiento del medio 125 de almacenamiento de memoria intermedia se suprime para no exceder la cantidad necesaria para almacenar de manera temporal los datos de proceso del código de ejecución (o el código de ejecución) necesarios para el proceso de ejecución por una vez. También, el medio 121 de determinación del tipo de código, el medio 122 de extensión de código, y el medio 123 de ejecución pueden ser respectivamente medios físicamente independientes, pero en los casos habituales, el dispositivo de ejecución del programa está compuesto de un ordenador tal como un así llamado micro ordenador o similar, y estos medios se constituyen, por ejemplo, mediante el dispositivo de procesamiento de control basado en una CPU. Entonces, según el dispositivo de procesamiento del control ejecuta los procesos del medio respectivo, se realizan los procesos de ejecución del dispositivo de ejecución del programa.

La Fig. 11 es un diagrama de flujo que describe las operaciones en el momento en que se realiza el proceso de ejecución en el dispositivo de ejecución del programa. En este momento, en concreto, se describirán principalmente las operaciones realizadas por el medio 121 de determinación del tipo de código, el medio 122 de extensión del código, y el medio 123 de ejecución del dispositivo de ejecución del programa. El medio 121 de determinación del tipo de código lee los datos del código de programa comprimido del medio 124 de almacenamiento del código de programa en una unidad de 1 byte (S1). Entonces, el medio 121 de determinación del tipo de código realiza la determinación de si los datos leídos son o no el comando o los datos de proceso (no el comando) en base al bit de determinación descrito en la Realización 1 que es “0” o “1” (S2). Cuando se determinan como datos de proceso (no el comando), los datos se almacenan en el medio 125 de almacenamiento de memoria intermedia mientras el bit de determinación se elimina del mismo (S3), y los datos comprimidos adicionales del código de programa se leen en una unidad de 1 byte (S1).

Por otro lado, cuando se determina que los datos leídos son el comando, el control se pasa al medio 122 de extensión del código. El medio 122 de extensión del código discrimina el comando representado por los datos en base a los datos leídos (los datos binarios) para convertir los datos en un comando en formato de texto. La relación entre los datos binarios y el comando descrita en la Realización 1 se usa para esta discriminación. Después de que el comando se discrimine, se descubre el elemento (objeto) de los datos de proceso ejecutados y procesados por el comando y la cantidad de datos asignados (la cantidad de bits), mediante lo cual en base a los datos binarios de los datos de proceso, se realiza la división de los datos de proceso para cada elemento (objeto). Se realiza el proceso de extensión anteriormente descrito, mediante el cual se genera la ejecución del código (S4).

Entonces, cuando se pasa el control desde el medio 122 de extensión del código al medio 123 de ejecución, se realiza el proceso de ejecución en base al código de ejecución generado por el medio 122 de extensión del código (S5). En el lenguaje basado en la notación Polaca inversa, cada elemento (objeto) se apila en una pila y se procesa en base al comando. Cuando se finaliza el proceso de ejecución basado en el comando, se pasa el control al medio 121 de determinación del tipo de código, y se realiza un proceso para el siguiente proceso de ejecución en base al código de ejecución. Como se describe anteriormente, el proceso de ejecución para cada código de ejecución se realiza de manera secuencial para ejecutar y procesar el código de programa, y se hace funcionar el aparato incorporado. También, la señal que contiene los datos basados en el proceso de ejecución se envía a un aparato distinto del aparato incorporado a través de la red para su operación.

Como se describe anteriormente, según la realización 1, los datos comprimidos del código de programa son leídos en una cantidad de datos predeterminada (1 byte según esta realización) desde el medio 124 de almacenamiento del código de programa. Cuando el medio 121 de determinación del tipo de código determina los datos leídos como los datos que contienen los datos de proceso, los datos de proceso se almacenan de manera temporal en el medio 125 de almacenamiento de memoria intermedia. Cuando se determina como los datos que contienen el comando, el medio 122 de extensión del código genera posteriormente el código de ejecución, y el medio 123 de ejecución realiza el proceso de ejecución en base al código de ejecución. Por tanto, basta que el medio 125 de almacenamiento de memoria intermedia simplemente almacene los datos de proceso hasta que el código de ejecución se genere, y como resultado, se puede reducir la capacidad de almacenamiento proporcionada como el medio 125 de almacenamiento de memoria intermedia. Además, como el proceso completo en el dispositivo de ejecución del programa desde la lectura de los datos comprimidos del código de programa hasta el proceso de ejecución del código de ejecución es simple, es posible evitar la disminución en la velocidad incluso no se mantiene el así generado código de ejecución (el código de programa). Como se describe anteriormente, el dispositivo de ejecución del programa según esta realización realiza un buen balance entre la reducción en la capacidad de almacenamiento relacionada con el proceso y el mantenimiento de la velocidad del proceso de ejecución, y se logra concretamente un efecto en el caso de que el dispositivo de ejecución del programa se incorpore en un electrodoméstico que requiere altamente la supresión de costes para el control del funcionamiento. Además, la señal que contiene los datos comprimidos del código de programa se envían a través del circuito de comunicación al aparato incorporado o similar de modo que los datos del medio 124 de almacenamiento del código de programa se pueden volver a escribir, mediante lo cual es posible mantener el código de programa del medio 124 de almacenamiento del código de programa actualizado de manera más eficiente en un estado desarrollado.

Ejemplo 4

La Fig. 12 es un dibujo que ilustra una configuración de datos del código de ejecución comprimido según la realización 5 de la presente invención. Según el Ejemplo 1 descrito anteriormente, los datos de proceso puestos en los datos binarios se dividen en 7 bits cada uno desde la cabecera (la así llamada alineación izquierda en la secuencia de datos representada), y si los datos de proceso no se dividen entre 7, el bit restante se rellena con un “0” o similar.

En este momento, cuando se realiza el proceso de ejecución en base al código de ejecución, en el caso de adoptar una configuración de datos tal como una pila, los datos apilados en el final resultan la cabecera, y el proceso se realiza desde los datos (esto es, el proceso de los datos en el lado derecho en la secuencia de datos representada). Por ejemplo, cuando se puede cambiar el número de elementos (objetos), los datos de este número se indican en la cabecera de la pila. Desde dicho punto, puede ser ventajosa la unificación en el mismo sistema de proceso. En vista de lo anterior, como se muestra en la Fig. 12(a), los datos binarios se establecen según la así llamada alineación derecha, y en el proceso de extensión en el medio 122 de extensión del código también, la cabecera resulta una parte de los datos de proceso (no los datos de relleno).

Además, según la Fig. 12(b), como se incluyen los datos que indican el número de bytes de datos que no contienen comandos en los datos de 1 byte (los datos que no contienen comandos) a ser leídos primero, es posible determinar si los datos contienen el comando o no en el medio 121 de determinación del tipo de código. En la Fig. 12(b), se asignan 5 bits (valores numéricos desde 0 a 31) a los datos que indican el número de bytes de los datos de proceso. Como resultado, no es necesario proporcionar el bit de determinación para cada byte de datos. Por esta razón, en concreto, como el número de bytes (el número de bits) de los datos que contienen los datos de proceso es mayor, se puede realizar una compresión más eficiente. Además, se descubre el número de bytes de los datos de proceso y el comando sigue posteriormente a los datos de proceso, y por lo tanto es innecesario proporciona el bit de determinación para el comando. Como resultado, comparado con el caso de proporcionar el bit de determinación, por ejemplo, el número de comandos que pueden ser discriminados en 1 byte es doble. Por esta razón, es efectivo también en el caso en que el número de comandos sea grande.

Ejemplo 5

Según las realizaciones anteriormente descritas, el código de programa se describe en el lenguaje basado en la notación Polaca inversa, y el medio 123 de ejecución es el intérprete para analizar la descripción para realizar el proceso de ejecución. La presente invención no se limita a esto. Por ejemplo, el intérprete puede estar compuesto de JavaVM (o JVM (Java y Java Virtual Machine JVM son nombres de marcas o marcas de Sun Microsystems, Inc. en los Estados Unidas de América y otros países) o similar, para realizar el proceso de ejecución en base al código de ejecución en el lenguaje de tipo pila, como un sistema de proceso. Como es aplicable a un lenguaje más general, existe la posibilidad de que se pueda realizar una generación de código más desarrollada.

REIVINDICACIONES

1. Un dispositivo de procesamiento de compresión de código de programa, para compresión de código de programa que está constituido por un código de ejecución descrito por una notación polaca inversa, que está compuesto de datos de proceso usados para un proceso de ejecución y un comando que representan un contenido de proceso de ejecución, que están dispuestos en orden, caracterizado por que comprende:

medios de discriminación (101A) para discriminar entre un dato de proceso y un comando en cada código de ejecución; y

medios de conversión (101B) para convertir piezas de los datos de proceso divididos por cada objeto en el código de ejecución en un formato binario, convertir el comando discriminado mediante los medios de discriminación (101A) en un valor numérico correspondiente al comando, dividir los datos de proceso convertidos en formato binario y el comando convertido en un valor numérico entre 7 bits para formar una pluralidad de grupos de 7 bits con cada grupo que corresponde a bien los datos de proceso o el comando, y, para cada grupo, añadir un bit que es una bandera para determinar si el grupo contiene el comando o no, donde

los medios de discriminación están configurados para cotejar el código de ejecución con un diccionario almacenado para discriminar entre los datos de proceso y el comando discriminando cadenas de caracteres y números que no están asociados con una coincidencia en el diccionario como los datos de proceso y discriminando cadenas de caracteres que están asociados con una coincidencia en el diccionario como el comando.

2. Un método de procesamiento de compresión de código de programa caracterizado por que comprende las etapas de:

leer un código de ejecución descrito por una notación polaca inversa, en el que los datos de procesos usados para un proceso de ejecución y un comando que representa un contenido de proceso de ejecución se disponen en orden, y discriminar entre los datos de proceso y el comando del código de ejecución mediante medios de discriminación,

convertir, mediante medios de conversión, los datos de proceso divididos por cada objeto en el código de ejecución en un formato binario,

convertir, mediante medios de conversión, el comando discriminado por los medios de discriminación, en un valor numérico correspondiente,

dividir, mediante medios de conversión, los datos de proceso convertidos en un formato binario y el comando convertido en un valor numérico entre 7 bits para formar una pluralidad de grupos de 7 bits con cada grupo que corresponde a bien los datos de proceso o el comando; y

añadir, para cada grupo, mediante métodos de conversión, un bit que es una bandera para determinar si el grupo contiene el comando o no, siendo llevadas a cabo las etapas para cada uno de los códigos de ejecución que constituyen un código de programa, donde

la etapa de discriminación incluye la etapa de

cotejar el código de ejecución con un diccionario almacenado para discriminar entre los datos de proceso y el comando, discriminando cadenas de caracteres y números que no están asociados con una coincidencia en el diccionario como los datos de proceso y discriminando cadenas de caracteres que están asociados con una coincidencia en el diccionario como el comando.

3. Un programa de ordenador que comprende instrucciones que, cuando el programa se ejecuta por un ordenador, hace que el ordenador lleve a cabo las etapas del método de la reivindicación 2.

F I G. 1

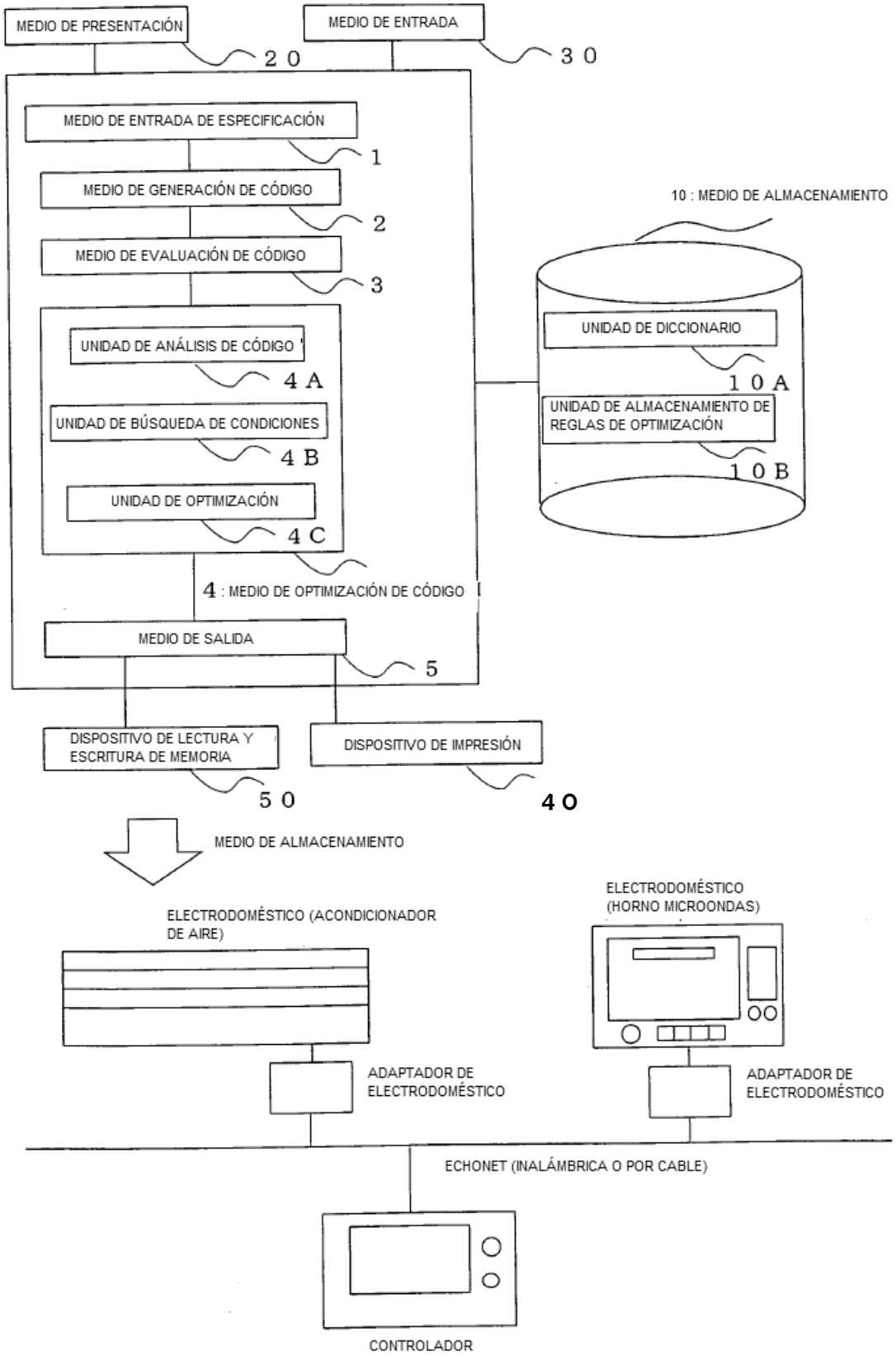

MEDIO DE PRESENTACIÓN
20
MEDIO DE ENTRADA
30
MEDIO DE ENTRADA DE ESPECIFICACIÓN
1
MEDIO DE GENERACIÓN DE CÓDIGO
2
MEDIO DE EVALUACIÓN DE CÓDIGO
3
UNIDAD DE ANÁLISIS DE CÓDIGO
4A
UNIDAD DE BÚSQUEDA DE CONDICIONES
4B
UNIDAD DE OPTIMIZACIÓN
4C
4 : MEDIO DE OPTIMIZACIÓN DE CÓDIGO
MEDIO DE SALIDA
5
10 : MEDIO DE ALMACENAMIENTO
UNIDAD DE DICCIONARIO
10A
UNIDAD DE ALMACENAMIENTO DE REGLAS DE OPTIMIZACIÓN
10B
DISPOSITIVO DE LECTURA Y ESCRITURA DE MEMORIA
50
DISPOSITIVO DE IMPRESIÓN
40
MEDIO DE ALMACENAMIENTO
ELECTRODOMÉSTICO (ACONDICIONADOR DE AIRE)
ELECTRODOMÉSTICO (HORNO MICROONDAS)
ADAPTADOR DE ELECTRODOMÉSTICO
ADAPTADOR DE ELECTRODOMÉSTICO
ECHONET (INALÁMBRICA O POR CABLE)
CONTROLADOR

FIG. 2

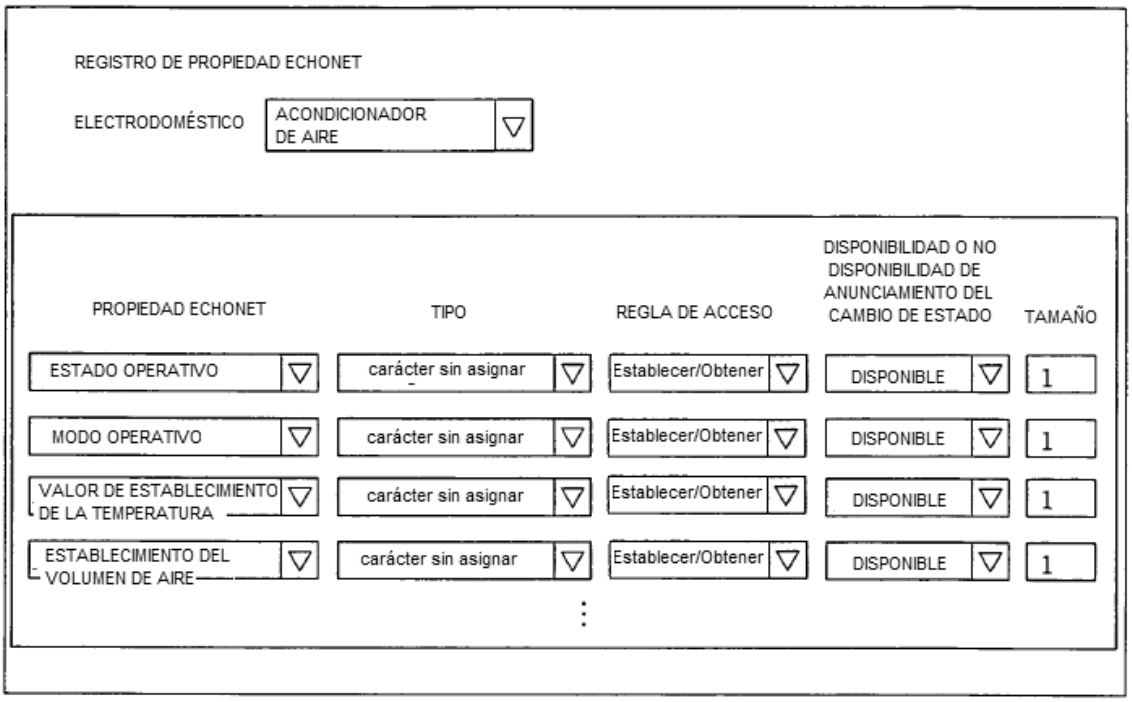
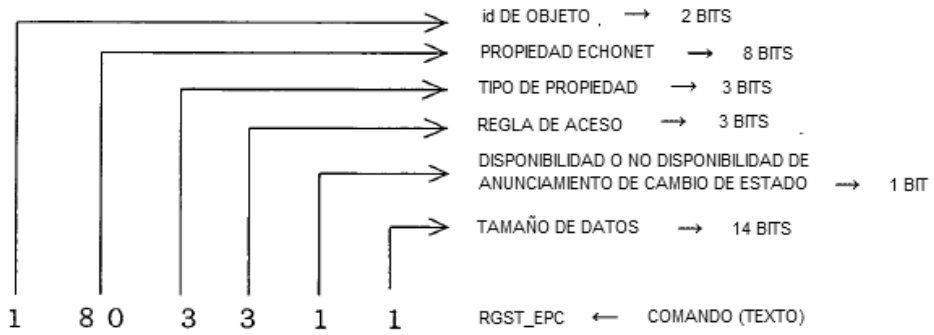
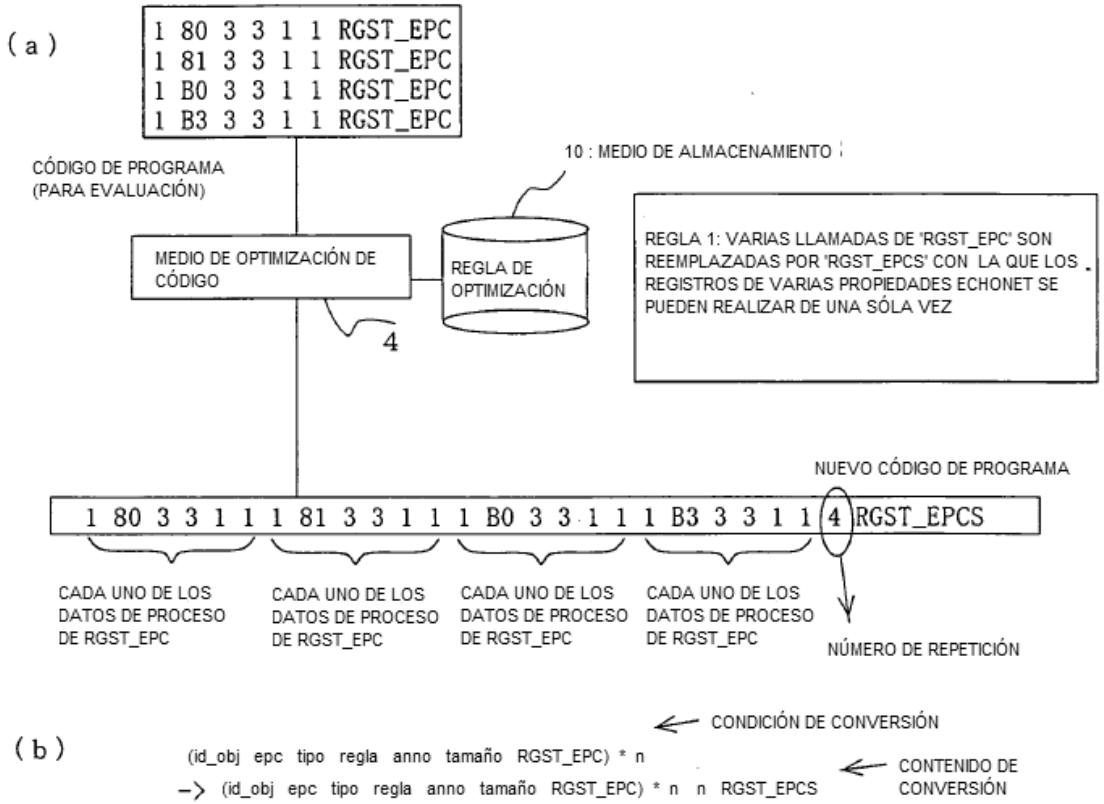

REGISTRO DE PROPIEDAD ECHONET
ELECTRODOMÉSTICO
ACONDICIONADOR DE AIRE
PROPIEDAD ECHONET
TIPO
REGLA DE ACCESO
DISPONIBILIDAD O NO DISPONIBILIDAD DE ANUNCIAMIENTO DEL CAMBIO DE ESTADO
TAMAÑO
ESTADO OPERATIVO
carácter sin asignar
Establecer/Obtener
DISPONIBLE
1
MODO OPERATIVO
carácter sin asignar
Establecer/Obtener
DISPONIBLE
1
VALOR DE ESTABLECIMIENTO DE LA TEMPERATURA
carácter sin asignar
Establecer/Obtener
DISPONIBLE
1
ESTABLECIMIENTO DEL VOLUMEN DE AIRE
carácter sin asignar
Establecer/Obtener
DISPONIBLE
1

FIG. 3


id DE OBJETO → 2 BITS
PROPIEDAD ECHONET → 8 BITS
TIPO DE PROPIEDAD → 3 BITS
REGLA DE ACESO → 3 BITS
DISPONIBILIDAD O NO DISPONIBILIDAD DE ANUNCIAMIENTO DE CAMBIO DE ESTADO → 1 BIT
TAMAÑO DE DATOS → 14 BITS
1 8 0 3 3 1 1
RGST_EPC ← COMANDO (TEXTO)

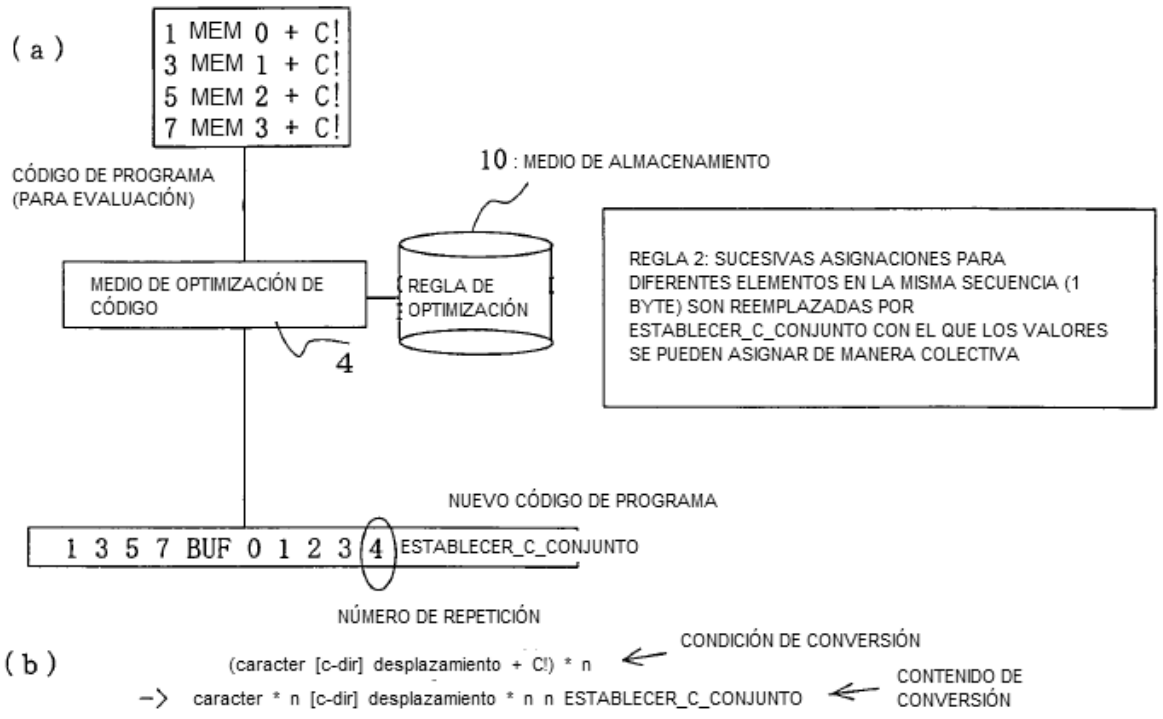
F I G. 4


(a)
1 80 3 3 1 1 RGST_EPC
1 81 3 3 1 1 RGST_EPC
1 B0 3 3 1 1 RGST_EPC
1 B3 3 3 1 1 RGST_EPC
CÓDIGO DE PROGRAMA (PARA EVALUACIÓN)
10 : MEDIO DE ALMACENAMIENTO
MEDIO DE OPTIMIZACIÓN DE CÓDIGO
REGLA DE OPTIMIZACIÓN
4
REGLA 1: VARIAS LLAMADAS DE 'RGST_EPC' SON REEMPLAZADAS POR 'RGST_EPCS' CON LA QUE LOS REGISTROS DE VARIAS PROPIEDADES ECHONET SE PUEDEN REALIZAR DE UNA SÓLA VEZ
NUEVO CÓDIGO DE PROGRAMA
1 80 3 3 1 1 1 81 3 3 1 1 1 B0 3 3 1 1 1 B3 3 3 1 1 4 RGST_EPCS
CADA UNO DE LOS DATOS DE PROCESO DE RGST_EPC
CADA UNO DE LOS DATOS DE PROCESO DE RGST_EPC
CADA UNO DE LOS DATOS DE PROCESO DE RGST_EPC
CADA UNO DE LOS DATOS DE PROCESO DE RGST_EPC
NÚMERO DE REPETICIÓN

(b)

CONDICIÓN DE CONVERSIÓN
(id_obj epc tipo regla anno tamaño RGST_EPC) * n
-> (id_obj epc tipo regla anno tamaño RGST_EPC) * n n RGST_EPCS
CONTENIDO DE CONVERSIÓN

F I G. 5


(a)
1 MEM 0 + C!
3 MEM 1 + C!
5 MEM 2 + C!
7 MEM 3 + C!
CÓDIGO DE PROGRAMA (PARA EVALUACIÓN)
10 : MEDIO DE ALMACENAMIENTO
MEDIO DE OPTIMIZACIÓN DE CÓDIGO
REGLA DE OPTIMIZACIÓN
4
REGLA 2: SUCESIVAS ASIGNACIONES PARA DIFERENTES ELEMENTOS EN LA MISMA SECUENCIA (1 BYTE) SON REEMPLAZADAS POR ESTABLECER_C_CONJUNTO CON EL QUE LOS VALORES SE PUEDEN ASIGNAR DE MANERA COLECTIVA
NUEVO CÓDIGO DE PROGRAMA
1 3 5 7 BUF 0 1 2 3 4 ESTABLECER_C_CONJUNTO
NÚMERO DE REPETICIÓN
(b)
(caracter [c-dir] desplazamiento + C!) * n
CONDICIÓN DE CONVERSIÓN
-> caracter * n [c-dir] desplazamiento * n n ESTABLECER_C_CONJUNTO
CONTENIDO DE CONVERSIÓN

F I G. 6

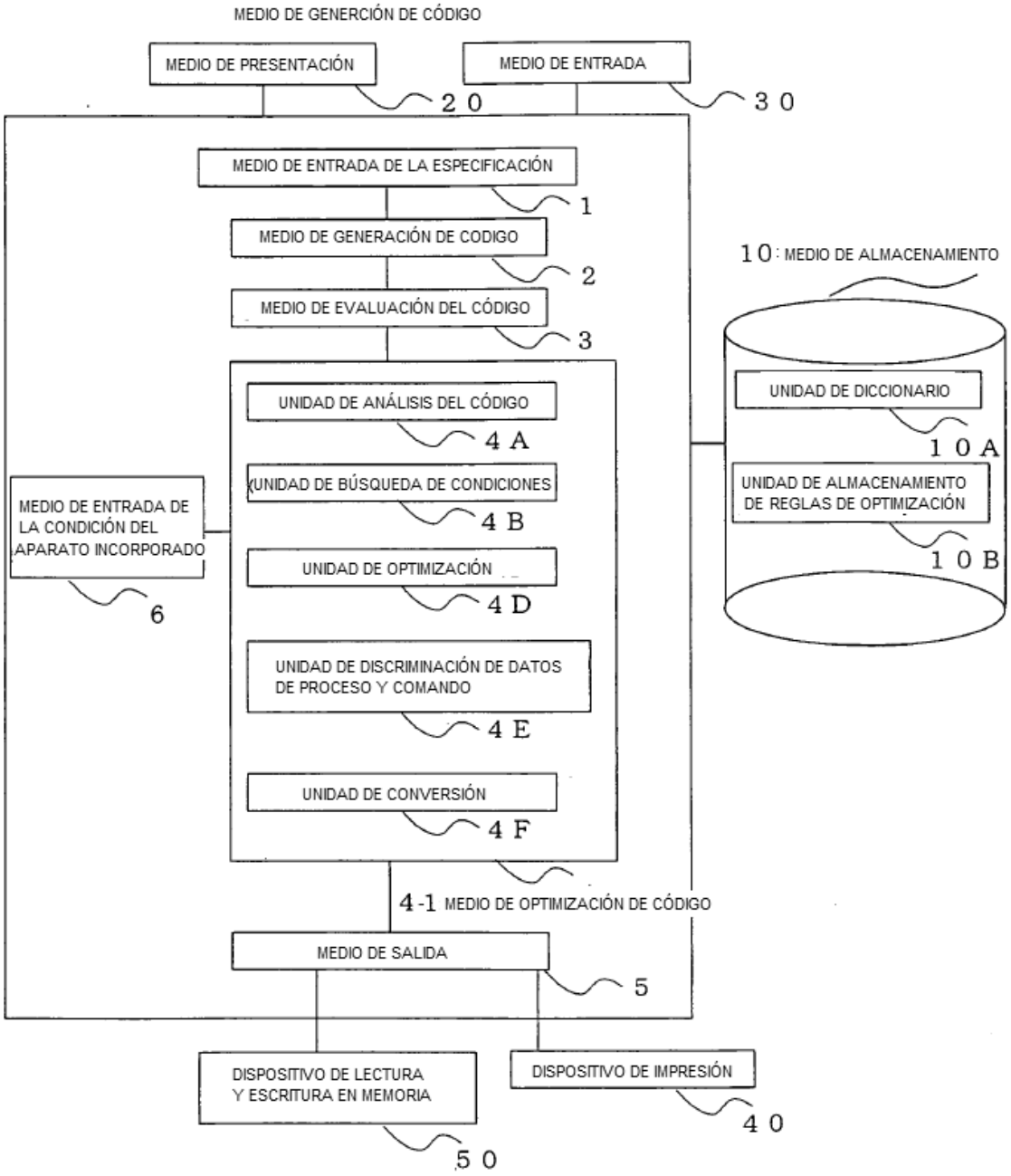

MEDIO DE GENERCIÓN DE CÓDIGO
MEDIO DE PRESENTACIÓN
20
MEDIO DE ENTRADA
30
MEDIO DE ENTRADA DE LA ESPECIFICACIÓN
1
MEDIO DE GENERACIÓN DE CODIGO
2
MEDIO DE EVALUACIÓN DEL CÓDIGO
3
UNIDAD DE ANÁLISIS DEL CÓDIGO
4A
UNIDAD DE BÚSQUEDA DE CONDICIONES
4B
UNIDAD DE OPTIMIZACIÓN
4D
UNIDAD DE DISCRIMINACIÓN DE DATOS DE PROCESO Y COMANDO
4E
UNIDAD DE CONVERSIÓN
4F
4-1 MEDIO DE OPTIMIZACIÓN DE CÓDIGO
MEDIO DE SALIDA
5
MEDIO DE ENTRADA DE LA CONDICIÓN DEL APARATO INCORPORADO
6
10 MEDIO DE ALMACENAMIENTO
UNIDAD DE DICCIONARIO
10A
UNIDAD DE ALMACENAMIENTO DE REGLAS DE OPTIMIZACIÓN
10B
DISPOSITIVO DE LECTURA Y ESCRITURA EN MEMORIA
50
DISPOSITIVO DE IMPRESIÓN
40

FIG. 7

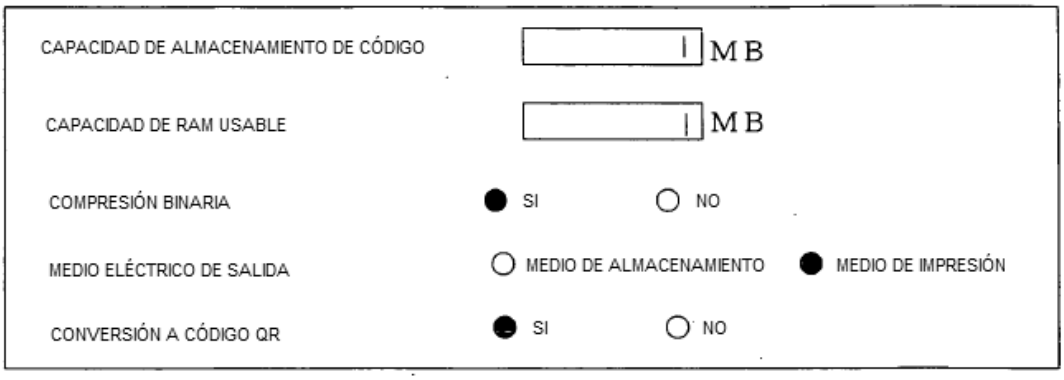
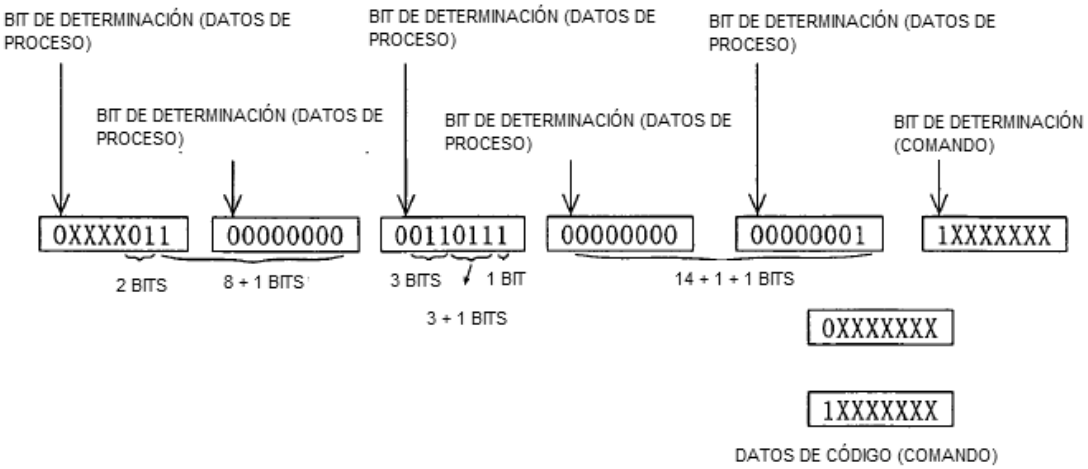
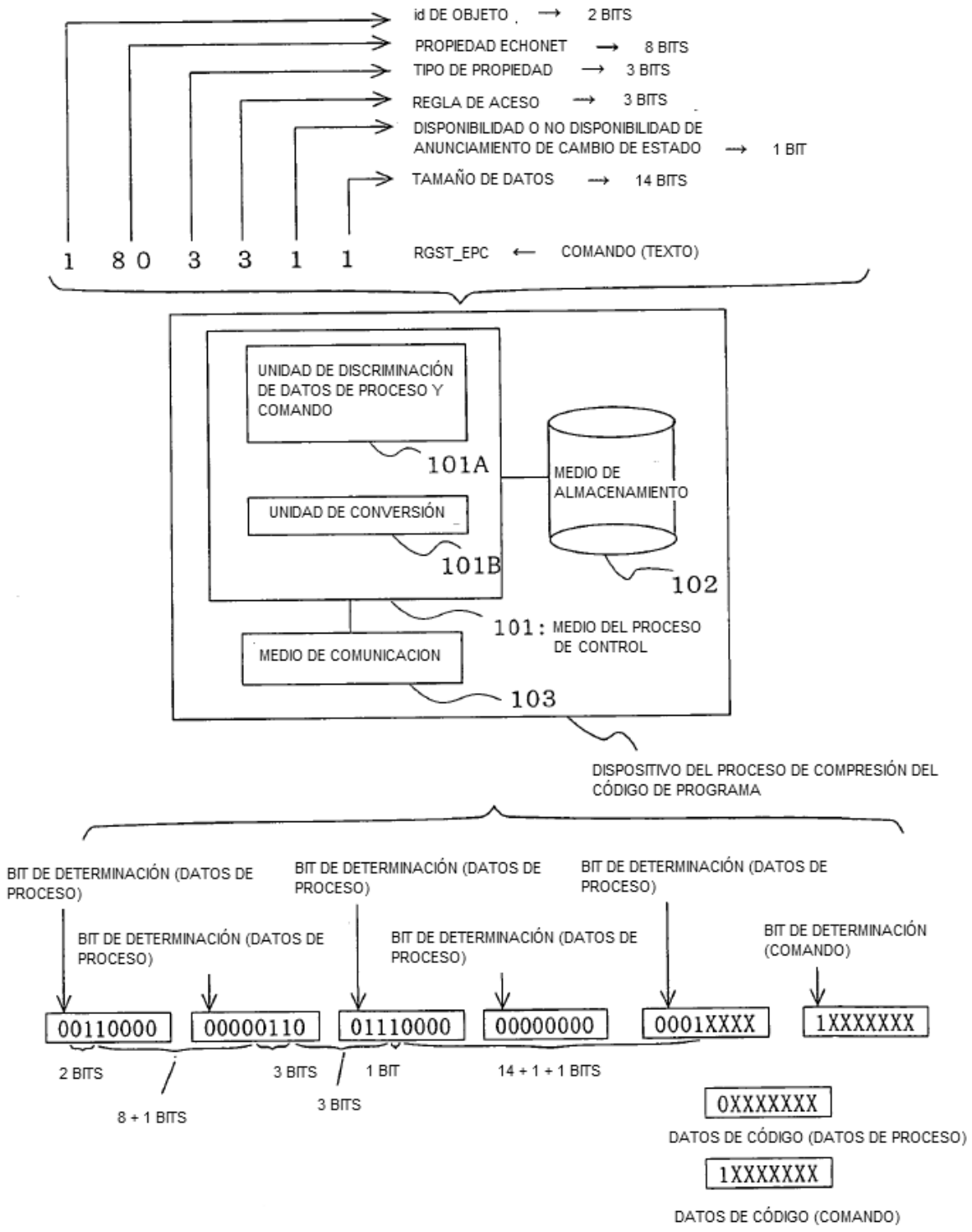

CAPACIDAD DE ALMACENAMIENTO DE CÓDIGO
MB
CAPACIDAD DE RAM USABLE
MB
COMPRESIÓN BINARIA
SI
NO
MEDIO ELÉCTRICO DE SALIDA
MEDIO DE ALMACENAMIENTO
MEDIO DE IMPRESIÓN
CONVERSIÓN A CÓDIGO QR
SI
NO

FIG. 8


BIT DE DETERMINACIÓN (DATOS DE PROCESO)
BIT DE DETERMINACIÓN (DATOS DE PROCESO)
BIT DE DETERMINACIÓN (DATOS DE PROCESO)
BIT DE DETERMINACIÓN (DATOS DE PROCESO)
BIT DE DETERMINACIÓN (DATOS DE PROCESO)
BIT DE DETERMINACIÓN (COMANDO)
0XXXX011
00000000
00110111
00000000
00000001
1XXXXXXX
2 BITS
8 + 1 BITS
3 BITS
1 BIT
3 + 1 BITS
14 + 1 + 1 BITS
0XXXXXXX
1XXXXXXX
DATOS DE CÓDIGO (COMANDO)

F I G. 9


id DE OBJETO → 2 BITS
PROPIEDAD ECHONET → 8 BITS
TIPO DE PROPIEDAD → 3 BITS
REGLA DE ACESO → 3 BITS
DISPONIBILIDAD O NO DISPONIBILIDAD DE ANUNCIAMIENTO DE CAMBIO DE ESTADO → 1 BIT
TAMAÑO DE DATOS → 14 BITS
1 8 0 3 3 1 1
RGST_EPC ← COMANDO (TEXTO)
UNIDAD DE DISCRIMINACIÓN DE DATOS DE PROCESO Y COMANDO
101A
UNIDAD DE CONVERSIÓN
101B
MEDIO DE ALMACENAMIENTO
102
101: MEDIO DEL PROCESO DE CONTROL
MEDIO DE COMUNICACION
103
DISPOSITIVO DEL PROCESO DE COMPRESIÓN DEL CÓDIGO DE PROGRAMA
BIT DE DETERMINACIÓN (DATOS DE PROCESO)
BIT DE DETERMINACIÓN (DATOS DE PROCESO)
BIT DE DETERMINACIÓN (DATOS DE PROCESO)
BIT DE DETERMINACIÓN (DATOS DE PROCESO)
BIT DE DETERMINACIÓN (DATOS DE PROCESO)
BIT DE DETERMINACIÓN (COMANDO)
00110000
00000110
01110000
00000000
0001XXXX
1XXXXXXX
2 BITS
8 + 1 BITS
3 BITS
3 BITS
1 BIT
14 + 1 + 1 BITS
0XXXXXXX
DATOS DE CÓDIGO (DATOS DE PROCESO)
1XXXXXXX
DATOS DE CÓDIGO (COMANDO)

F I G. 1 0

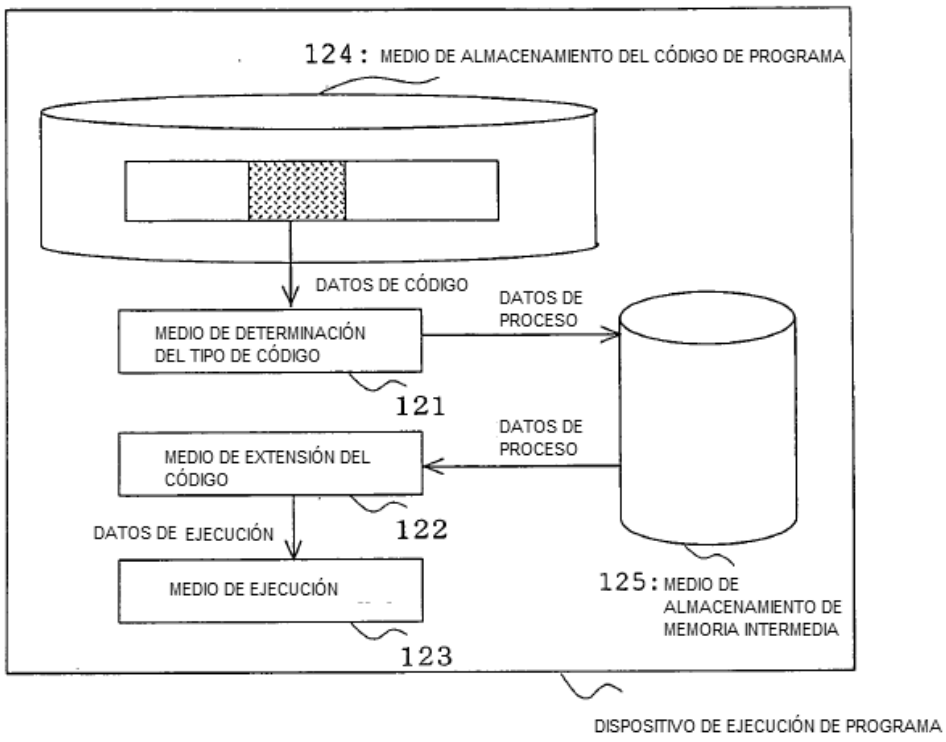

124: MEDIO DE ALMACENAMIENTO DEL CÓDIGO DE PROGRAMA
DATOS DE CÓDIGO
MEDIO DE DETERMINACIÓN DEL TIPO DE CÓDIGO
121
DATOS DE PROCESO
DATOS DE PROCESO
MEDIO DE EXTENSIÓN DEL CÓDIGO
122
DATOS DE EJECUCIÓN
MEDIO DE EJECUCIÓN
123
125: MEDIO DE ALMACENAMIENTO DE MEMORIA INTERMEDIA
DISPOSITIVO DE EJECUCIÓN DE PROGRAMA

FIG. 11

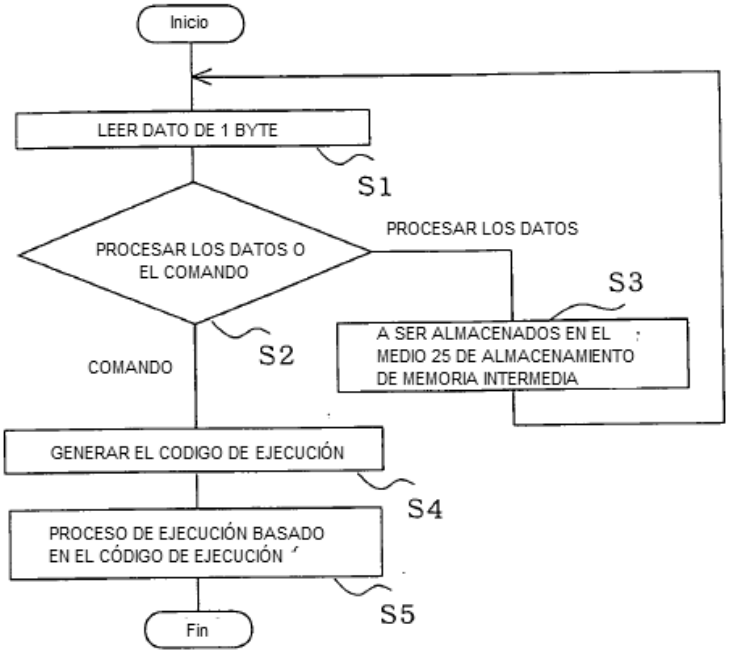
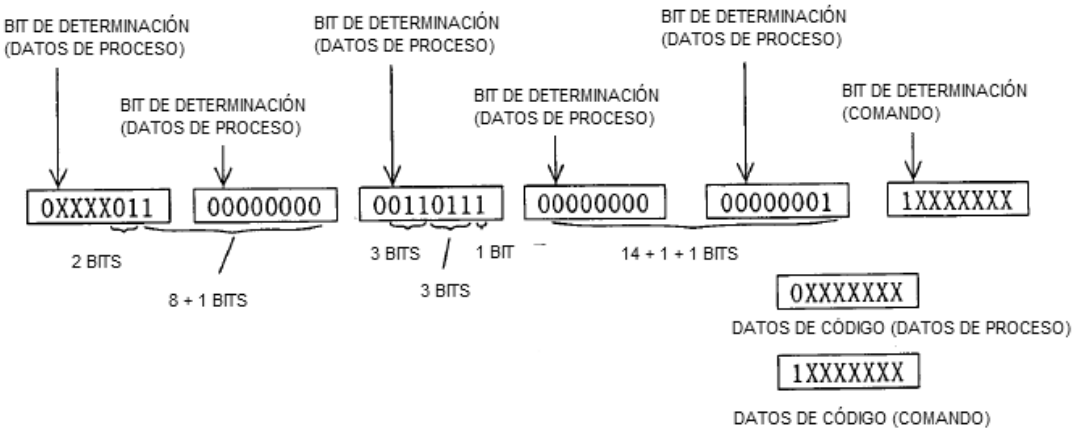


FIG. 12

(a)



(b)