(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 January 2007 (11.01.2007)

PCT

(10) International Publication Number
WO 2007/005790 A2

(51) International Patent Classification:
G06F 11/00 (2006.01) *G06F 9/44* (2006.01)

(21) International Application Number:
PCT/US2006/025912

(22) International Filing Date: 30 June 2006 (30.06.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/696,096 30 June 2005 (30.06.2005) US

(71) Applicant *(for all designated States except US)*: **SLING MEDIA, INC.** [US/US]; 901 Mariners Island Blvd., Suite 300, San Mateo, CA 94404 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **TARRA, Raghuveer** [IN/US]; 901 Mariners Island Blvd., Suite 300, San Mateo, CA 94404 (US).

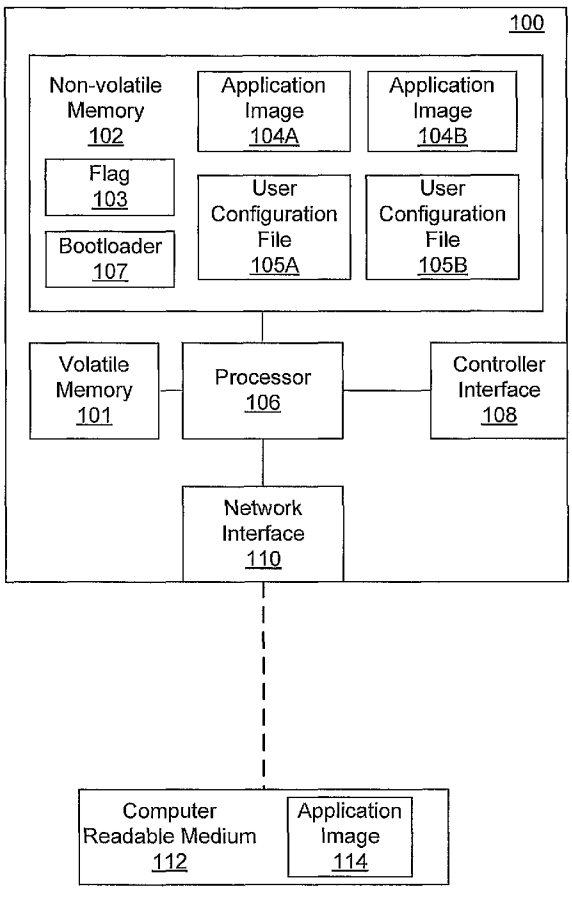
(74) Agents: **HULSE, Robert, A.** et al.; FENWICK & WEST LLP, SILICON VALLEY CENTER, 801 California Street, Mountain View, CA 94041 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI,

[Continued on next page]

(54) Title: FIRMWARE UPDATE FOR CONSUMER ELECTRONIC DEVICE



(57) Abstract: To update firmware on a consumer device intelligently, two or more application images are stored as firmware on the consumer device. If the primary application image is corrupt, the back-up application image is executed on the consumer device. The back-up application image can be updated based on the primary application image. User configuration files can be preserved during the update of an application image, or they can be overwritten. This firmware updating scheme can be advantageously implemented in a personal media broadcasting system.

WO 2007/005790 A2

FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

FIRMWARE UPDATE FOR CONSUMER ELECTRONIC DEVICE

Inventor:
Raghuveer Tarra

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application No. 60/696,096, filed June 30, 2005, which is incorporated by reference in its entirety.

BACKGROUND

[0002] This invention relates generally to consumer devices having one or more processors, and, more specifically, to updating and loading the firmware stored on such devices.

[0003] A number of consumer grade devices that run embedded software have the ability to accept firmware updates that allow the end user to use feature enhancements and to address problem fixes. These updates are often initiated by the user and are typically critical in nature. Failures in the update process are often catastrophic.

[0004] A failure could be due to power failure or glitches on the embedded device and/or the host initiating the update. Failures may also result from network interruptions and any number of other problems. When a failure occurs, the firmware on the consumer device is often erased or otherwise corrupted. Because firmware is typically integral to the successful operation of the device, the corruption or loss of the firmware can cause the consumer device to perform improperly, and in some cases, to be completely inoperable. Perhaps worst of all, once the device has been rendered inoperable, it may be impossible or impractical to restore the firmware. Thus, a failure during a firmware upgrade can result in the loss of the consumer device.

[0005] Therefore, what is needed is a way to intelligently update the firmware of a consumer device.

SUMMARY OF THE INVENTION

[0006] To avoid the problems in previous consumer devices, embodiments of the invention intelligently update the firmware of a consumer device. A back-up application image is stored as firmware on the consumer device. If the primary application image is

corrupted, the back-up application image can be executed instead, maintaining at least some of the functionality of the consumer device.

[0007] In one embodiment, two or more application images are stored as firmware on the consumer device. A processor executes the first application image. A determination is made as to whether the first application image is corrupt. If the first application image is determined to be corrupt, the processor executes the second application image. By executing the second application image as a back-up, the consumer device is able operate as normal even if the first application image is corrupt. The consumer device may also initiate an update to repair the corrupted application image. Because user configuration files are stored independently of application images, the update can repair the corrupted application image without overwriting user configuration files.

[0008] Furthermore, so that the back-up firmware can provide greater functionality, embodiments of the invention update the back-up application image. A processor executes a primary application image stored as firmware on the consumer device. The first application image is updated with an updated application image. If the updated application image is determined to be not corrupt, the updated application image is used to update the back-up application image, which is stored as firmware on the consumer device. The back-up application image is available in case the primary image is subsequently corrupted. In this way, the back-up application image can be safely updated while reducing the likelihood of corrupting the back-up image. By updating the back-up application image, a back-up copy of firmware can be kept up to date with non-corrupt application images.

[0009] The first application image can be determined to be corrupt using a variety of methods. According to one embodiment, the first application image can be determined to be corrupt by reading one or more flags to determine if an update was initiated but not completed. An incompletely updated application image is determined to be corrupt. According to another embodiment, a checksum is used to determine if an application image is corrupt.

[0010] The features and advantages described in this summary and the following detailed description are not all-inclusive. Many additional features and advantages will be apparent to one of ordinary skill in the art in view of the drawings, specification, and claims hereof.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] FIG. 1 is a block diagram illustrating a consumer device, according to one embodiment of the present invention.

[0012] FIG. 2 is a state diagram illustrating the execution of a plurality of application images, according to one embodiment of the present invention.

[0013] FIG. 3 is a flow chart illustrating a method for updating an application image, according to one embodiment of the present invention.

[0014] FIG. 4 is a flow chart illustrating a method for loading an application image, according to one embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0015] In one embodiment of the present invention, a back-up application image is stored locally in a consumer device. If the primary application image is corrupted (for example, due to an update failure or a data loss), the back-up application image can be executed instead, maintaining at least some of the functionality of the consumer device. During the execution of the back-up application image, the primary application image can be restored. Storing two application images in the consumer device beneficially increases the failure resistance and reliability of the consumer device.

[0016] In one embodiment, the consumer device is a personal media broadcaster, such as the one described in U.S. Application No. 11/147,664, entitled "Personal Media Broadcasting System," filed June 7, 2005, which is incorporated by reference in its entirety. An application image for a personal media broadcaster may include processor instructions causing the media broadcaster to encode a video source and stream it over a local or remote network to a media player. Storing two or more application images as firmware on a personal media broadcaster is particularly beneficial given the typically challenging operating conditions of personal media broadcasters. Personal media broadcasters are often power-cycled without warning, have unreliable connections to the Internet, and are frequently moved, resulting in a high risk of firmware corruption. Furthermore, users often expect that, with minimal user intervention, their personal media broadcasters be up-to-date with continuously improved firmware. The intelligent firmware update described herein advantageously provides reliable firmware updates for a personal media broadcaster operating under even the most challenging of conditions.

[0017] FIG. 1 is a block diagram illustrating a consumer device, according to one embodiment of the present invention. The consumer device 100 contains a non-volatile memory 102, a volatile memory 101, a processor 106, and at least one interface for receiving firmware updates. For example, the consumer device 100 illustrated in the figure includes a network interface 110 and a controller interface 108. The network interface 110 can be implemented, for example, as an Ethernet card, an 801.11-compliant wireless device, a modem, a cellular modem, or another interface capable of sending and receiving data on a network. Other components not illustrated can also be included in the consumer device 100.

[0018] The volatile memory 101 can be implemented as any kind of computer-readable medium. The volatile memory 101 can be implemented, for example, by dynamic or static random access memory. The volatile memory 101 is capable of storing data and processor instructions for the processor 106. Typically, the volatile memory 101 is designed so that it will lose its contents if power to the volatile memory 101 is disconnected, but it need not be.

[0019] The non-volatile memory 102 can be implemented as any kind of computer-readable medium that retains its contents when power to the non-volatile memory 102 is disconnected. The non-volatile memory 102 can be implemented, for example, by a disk drive, flash memory, Electrically Erasable Programmable Read-Only Memory (EEPROM), or magnetic, Ovonic Unified, or ferroelectric random access memory. Although the non-volatile memory 102 is illustrated as a single component, the non-volatile memory 102 can be implemented as any number of computer-readable media. When reference is made herein to distinct non-volatile memories, it should be understood that in one embodiment these distinct non-volatile memories are implemented as distinct locations within the same non-volatile memory. In another embodiment, distinct non-volatile memories are implemented as physically separate components of the consumer device 100.

[0020] The non-volatile memory 102 stores processor instructions executable by the processor 106. A group of computer instructions is typically organized into a functional program called an application image 104. An application image 104 (sometimes called a "binary image") is a set of processor instructions that, when executed by the processor 106, will cause the consumer device 100 to have its intended functionality. Because the application image 104 is stored embedded in the consumer device 100, the application image 104 is often referred to as "firmware."

[0021] The non-volatile memory 102 includes two or more application images, such as application image 104A and application image 104B. Application image 104A and

application image 104B are independent and distinct groups of processor instructions. When executed on the processor 106, either application image 104 will allow the consumer device 100 to perform its intended functionality without relying on the other (provided the application image is not corrupt).

[0022] The application images 104A and 104B can be identical instruction-for-instruction, or they can be different. For example, application image 104A can be an updated version of the application image 104B, implementing additional functionality of which the application image 104B is not capable. As another example, application image 104A can be corrupt, and application image 104B can be not corrupt.

[0023] According to one embodiment of the present invention, the application image 104A is stored on a first non-volatile memory and the application image 104B is stored on a second non-volatile memory. The first non-volatile memory and the second non-volatile memory can be implemented, for example, as separate locations on a single component of non-volatile memory 102. As another example, the first non-volatile memory and the second non-volatile memory can be implemented as distinct components of non-volatile memory. In FIG 1, the first and second non-volatile memories are implemented in a single physical non-volatile memory 102.

[0024] The non-volatile memory 102 may also contain one or more user configuration files 105. A user configuration file 105 comprises data stored on a computer-readable medium that indicates user preferences for the operation of the consumer device 100. According to one embodiment of the present invention, the memory 102 contains two or more user configuration files 105A and 105B, each configuration file 105 associated with an application image 104. While an application image 104 is executing on the processor, the user configuration file 105 associated with that application image 104 is considered the active user configuration file for the consumer device 100.

[0025] According to one embodiment of the present invention, the non-volatile memory 102 contains one or more flags 103 indicating whether an update of the application image 104A has been initiated and whether the update of the application image 104A was completed. Additional flags can also be implemented to indicate the update or corruption status of other application images, configuration files, and so on.

[0026] According to one embodiment of the present invention, the non-volatile memory 102 contains a bootloader 107. The bootloader 107 is a set of processor instructions for

initializing the consumer device 100 and beginning the execution of one of the application images 104.

[0027] The volatile memory 101 and the non-volatile memory 102 have been illustrated as separate components for the purpose of illustration, but various embodiments of the invention are not limited to such a configuration. In one embodiment, the data structures described as being stored on the volatile memory 101 and the data structures described as being stored on the non-volatile memory 102 are stored on a common memory. Furthermore, the examples of volatile and non-volatile memories described herein are given for the purposes of illustration and are not limiting. Other examples of volatile and non-volatile memory will be apparent to one of skill in the art without departing from the scope of the present invention.

[0028] FIG. 2 is a state diagram illustrating the execution of a plurality of application images, according to one embodiment of the present invention. The consumer device 100 begins in a powered down state 202. When the consumer device 100 is in the powered down state 202, the processor 106 does not execute processor instructions. In some embodiments, data stored on the volatile memory 101 is lost when the consumer device 100 is in the powered down state 202.

[0029] Power on brings the consumer device 100 to an executing bootloader state 204. In the executing bootloader state 204, the consumer device 100 initializes and determines which application image 104 to load. In one embodiment, the application image 104A serves as a primary application image, and the application image 104B serves as a back-up application image, available for execution if the application image 104A is determined to be corrupt. Under normal conditions, the application images 104A is selected. If the application image 104A is determined 204 to be corrupt, the application image 104B is selected instead.

[0030] According to one embodiment of the present invention, in the executing bootloader state 204 the consumer device 100 copies the selected application image 104 from the non-volatile memory 102 to the volatile memory 101.

[0031] If the application image 104A is determined to be not corrupt, the consumer device 100 enters the executing application image 104A state 206. The processor instructions of application image 104A are executed by the processor. In state 206, the user configuration file associated with either the application image 104A is used as the active user configuration file.

[0032] If the application image 104A is determined to be corrupt, the consumer device 100 enters the executing application image 104B state 208. The processor instructions of application image 104B are executed by the processor. In state 208, the user configuration file associated with either the application image 104A or the application image 104B is used as the active user configuration file, according to various embodiments. According to one embodiment of the present invention, in state 208 the consumer device 100 determines if the user configuration file associated with application image 104A is corrupt. If the user configuration file associated with application image 104A is not corrupt, it is the active user configuration file. If the user configuration file associated with application image 104A is corrupt, another user configuration file is the active user configuration file.

[0033] A command to update the firmware causes the consumer device 100 to enter the running update state 210. In state 210, the consumer device 100 receives an updated application image 114 and stores the updated application image in the non-volatile memory 102. The successful completion of the update returns the consumer device 100 to the executing application image state. According to one embodiment of the present invention, at the successful completion of the update, the consumer device 100 restarts, causing the consumer device 100 to return to the powered down state 202.

[0034] Power-cycling the consumer device 100 during any state restarts the consumer device 100 and causes the consumer device 100 to return the powered down state 202. For example, if power is inadvertently removed in the running update state 208B, the consumer device 100 returns to the powered down state 202, even though the update of the application image 104 might not be complete.

[0035] FIG. 3 is a flow chart illustrating a method for updating an application image, according to one embodiment of the present invention. An updated application image is used to update the primary application image.

[0036] Updated application images can be obtained from a variety of sources, either local or remote. Typically, an application image 114 for update is stored on a computer readable medium 112 external to the consumer device 100. The consumer device 100 can access the computer readable medium 112 through the network interface 110 (for example, to access an application image stored on a local or remote server), the controller interface 108 (for example, to access a disk drive or USB port) or any other suitable mechanism. Various methods for obtaining an application image 114 for update will be apparent to one of skill in the art without departing from the scope of the present invention. Typically, updating the

application image 104A includes copying the application image 114 to the non-volatile memory 102 storing the application image 104A or 104B.

[0037] An update of an application image can be initiated, for example, by a local or remote user command, or it can be initiated automatically in response to determining that the application image is corrupt, or in response to determining that an updated application image is available. For the purposes of illustration, the method is described for updating the application image 104A, but according to various embodiments, the method can also be used to update other application images 104 stored as firmware on the consumer device 100. . According to one embodiment of the present invention, the update is performed by the consumer device 100.

[0038] The consumer device 100 optionally sets 302 a flag indicating that an update of the application image has been initiated. According to one embodiment of the present invention, a flag is used to indicate that an application image update has been initiated and/or completed. If the update process is interrupted or for any reason does not complete, the flag will reflect the interruption and facilitate determining that the application image 104A is corrupt.

[0039] The consumer device 100 updates 304 the application image 104A. According to one embodiment of the present invention, updating 304 the application image 104A includes modifying or replacing the processor instructions of the application image 104A with processor instructions of an application image 114 stored on a computer readable medium 112 external to the consumer device 100. For example, using a local or network connection, the consumer device 100 can copy the application image 114 to the non-volatile memory 102 that previously stored the application image 104A. The non-volatile memory 102 that previously stored the application image 104A now stores an updated application image. According to one embodiment of the present invention, the consumer device 100 copies the application image 114 to the non-volatile memory 102 in 64--kilobyte segments and verifies that each segment was copied correctly into the non-volatile memory 102 before proceeding to the next segment.

[0040] According to one embodiment of the present invention, updating 304 the application image 104A includes copying the application image 104B to the non-volatile memory 102 that previously stored the application image 104A. The non-volatile memory 102 that previously stored the application image 104A now stores a recovered application image. Such a recovery can be useful, for example, if the user would like to return the consumer device 100 to a previous firmware version, or to standard factory settings.

According to one embodiment of the present invention, the user can initiate a "hard reset" update that will copy the application image 104B to the non-volatile memory 102 that previously stored the application image 104A. Hard resets can be initiated, for example, by a software command, by a hardware button on the consumer device 100, or both.

[0041] Updating 304 the application image 104A can include overwriting the user configuration file associated with the application image 104A, or it can preserve the user configuration file. Preserving the user configuration file beneficially allows a user to enjoy the functionality of the consumer device 100 without reconfiguration.

[0042] The consumer device 100 optionally sets 306 a flag indicating that an update of the application image is complete. According to one embodiment of the present invention, a flag is used to indicate that an application image update has been initiated and/or completed. Setting 306 the flag indicating that the update is complete beneficially facilitates determining that the updated application image is not corrupt.

[0043] The consumer device 100 loads 308 an application image 104 from the non-volatile memory 102. For example, the consumer device 100 can restart and load an application image 104 as in a typical start-up event. A method for loading an application image 104, according to one embodiment of the present invention, is described herein with reference to FIG. 4. In one embodiment, loading 308 an application image includes determining if the application image 104A is corrupt. If the application image 104A is determined to be corrupt, the consumer device 100 can automatically restart the update of the application image 104A.

[0044] The consumer device 100 optionally updates 310 application image 104B. According to one embodiment of the present invention, certain application image updates can be designated as capstone updates which update both a primary application image and a back-up application image. Updating 310 application image 104B can include setting a flag indicating that an update of application image 104B has been initiated, copying the updated application image to the non-volatile memory 102 previously storing application image 104B, and setting a flag indicating that an update of the application image 104B is complete. Updating application image 104B can further include associating application image 104B with an updated user configuration file, such as the user configuration file previously associated with application image 104A.

[0045] Updating 310 application image 104B beneficially adds features and improvements to the back-up application image, so that the functionality of the consumer

device 100 when the back-up application image is loaded will be similar to the functionality when the updated application image is loaded. Furthermore, by updating 310 application image 104B after application image 104A has been successfully updated and loaded, corruption of the application image 104B can be avoided, increasing the reliability of the consumer device 100.

[0046] FIG. 4 is a flow chart illustrating a method for loading an application image, according to one embodiment of the present invention. The consumer device 100 loads an application image as part of the start-up process.

[0047] Upon a start-up event, such as a power cycle or a restart, the processor 106 executes the processor instructions of the bootloader 107. The processor instructions of the bootloader 107 cause the consumer device 100 to load an application image 104 and begin executing the processor instructions of the loaded application image 104.

[0048] The consumer device 100 determines 402 if the application image 104A is corrupt. According to one embodiment of the present invention, the consumer device 100 determines 402 if the application image 104A is corrupt by determining a checksum of the first application image 104A. If the checksum does not match an expected value, the application image 104A is determined 402 to be corrupt. According to another embodiment of the present invention, the consumer device 100 determines if the application image 104A is corrupt by reading a flag to determine if an update of the application image 104A has been initiated. If reading the flag indicates that the application image 104A has been initiated, but not completed, the application image 104A is determined 402 to be corrupt. According to yet another embodiment of the present invention, the determination 402 can include a combination of reading a flag to determine if an update of the application image 104A has been initiated and determining a checksum. These examples of methods for determining if the application image 104A is corrupt are given for the purposes of illustration only and are not limiting. Other examples of method for determining 402 if the application image 104A is corrupt will be apparent to one of skill in the art without departing from the scope of the present invention.

[0049] If the consumer device 100 determines 402 that the application image 104A is not corrupt, the consumer device 100 loads 404 application image 104A. According to one embodiment of the present invention, loading 404 application image 104A includes copying application image 104A from a first non-volatile memory 102 to a volatile memory 101. The processor 106 begins executing the processor instructions of application image 104A. Loading and executing a non-corrupt application image 104A beneficially allows the

consumer device 100 to operate with a primary firmware image, providing the functionality and reliability to which the user is accustomed.

[0050] If the consumer device 100 determines 402 that the application image 104A is corrupt, the consumer device 100 loads 406 the application image 104B. For example, the consumer device 100 can load 406 the application image 104B by copying the image 104B to the volatile memory 101. As another example, the consumer device 100 can load 406 the application image 104B by copying the image 104B to the volatile memory 101 and to the non-volatile memory 102 that previously stored application image 104A. The processor 106 begins executing the processor instructions of application image 104B.

[0051] Loading and executing application image 104B beneficially allows the consumer device 100 to operate with its intended functionality, even when application image 104A is corrupt. In some cases such as when the application image 104A is a more recent firmware version than the application image 104B, certain advanced functionality may be reduced or modified during the execution of the application image 104B.

[0052] According to one embodiment of the present invention, the consumer device 100 updates 408 application image 104A. If the application image 104A is determined 402 to be corrupt, the consumer device 100 initiates an update to repair the application image 104A. The update of the application image 104A can start automatically, or a message can be presented to a user indicating that the primary application image is corrupt and prompting the user for a command to initiate an update.

[0053] According to one embodiment of the present invention, an update of an application image can operate in at least two modes, one that will preserve the user configuration file associated with application image 104A, and another that will overwrite the user configuration file. According to one embodiment of the present invention, the consumer device 100 determines whether the user configuration file should be overwritten. For example, if the consumer device 100 is updating the application image 104A in response to determining that the application image 104A is corrupt, the user configuration file would not be overwritten. As another example, if the consumer device 100 is updating the application image 104A in response to a user-initiated hard-reset, the user configuration file would be overwritten, for example, with the user configuration file associated with application image 104B. Updating the user configuration file advantageously allows user preferences to be reset to previous or default settings, enabling the consumer device 100 to recover from potentially inoperable user settings.

[0054] In the figure and accompanying description, reference is made to performing various actions, such as determining corruption, loading, and updating, on application images. For the purpose of illustration, such actions are discussed with reference to application images 104A and 104B. However, according to various embodiments, the steps described herein can also be performed on other application images. For example, the method described herein with reference to FIG. 4 can also be performed on updated and/or recovered application images.

[0055] Where reference is made herein to a primary embodiment containing two distinct application images 104, this discussion has been provided for the purposes of illustration and is not limiting. One of skill in the art will appreciate that further benefits are achieved by storing more than two distinct application images 104 in the consumer device 100. Such benefits include, but are not limited to, further increased failure resistance, user selection of a plurality of firmware versions, and redundant application image back-up. The methods and systems described herein can also be modified to implement embodiments with more than two application images, as will be will be apparent to one of skill in the art without departing from the scope of the present invention.

[0056] The foregoing description of the embodiments of the invention has been presented for the purpose of illustration; it is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Persons skilled in the relevant art can appreciate that many modifications and variations are possible in light of the above teachings. It is therefore intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto.

What is claimed is:

1. A method for executing firmware in a consumer device, the method comprising:

executing on a processor of the consumer device a first application image stored as firmware on the consumer device;

restarting the consumer device;

determining if the first application image is corrupt; and

responsive to determining that the first application image is corrupt, executing on the processor a second application image stored as firmware on the consumer device.

2. The method of claim1, further comprising:

initializing an update of the first application image.

3. The method of claim 2, wherein initializing the update of the first application image comprises:

setting a flag indicating that an update of the first application image has been initiated; and

storing the flag in a memory of the consumer device.

4. The method of claim 3, wherein determining if the first application image is corrupt comprises:

reading the flag to determine if an update of the first application image has been initiated.

5. The method of claim 1, wherein determining if the first application image is corrupt comprises:

determining a checksum of the first application image.

6. The method of claim 1, wherein executing the first application image comprises:

copying the first application image from a first non-volatile memory to a volatile memory; and

executing on the processor the first application image from the volatile memory.

7. The method of claim 6, wherein executing the second application image comprises:

copying the second application image from a second non-volatile memory to the volatile memory; and

executing on the processor the second application image from the volatile memory.

8. The method of claim 6, wherein executing the second application image comprises:

copying the second application image from a second non-volatile memory to the first non-volatile memory;

copying the second application image from the first non-volatile memory to the volatile memory; and

executing on the processor the second application image from the volatile memory.

9. The method of claim 1, further comprising:

responsive to determining that the first application image is corrupt, initializing an update of the first application image.

10. The method of claim 9, wherein initializing an update of the first application image comprises:

updating a user configuration file stored in a memory of the consumer device.

11. The method of claim 1, wherein restarting the consumer device comprises power-cycling the consumer device.

12. The method of claim 1, wherein the first application image is different than the second application image.

13. The method of claim 1, wherein the consumer device comprises a personal media broadcasting system.

14. A method for updating firmware on a consumer device, the method comprising:

executing on a processor of the consumer device a primary application image stored as firmware on the consumer device;

updating the primary application image with an updated application image;

determining if the updated application image is corrupt;

responsive to determining that the updated application image is not corrupt, updating a back-up application image stored as firmware on the consumer device with the updated application image.

15. The method of claim 14, wherein updating the primary application image with an updated application image comprises:

setting a flag indicating that an update of the primary application image is complete; and

storing the flag in a memory of the consumer device.

16. The method of claim 15, wherein determining if the updated application image is corrupt comprises:

reading the flag to determine if the update of the primary application image is complete.

17. The method of claim 14, wherein determining if the updated application image is corrupt comprises:

determining a checksum of the updated application image.

18. The method of claim 14, wherein executing the primary application image comprises:

copying the primary application image from a first non-volatile memory to a volatile memory; and

executing on the processor the primary application image from the volatile memory.

19. The method of claim 18, wherein updating the primary application image with an updated application image comprises:

copying the updated application image to the first non-volatile memory.

20. The method of claim 18, wherein the back-up application image is stored in a second non-volatile memory, and wherein updating the back-up application image with the second application image comprises:

copying the updated application image to the second non-volatile memory.

21. The method of claim 14, wherein the updated application image is stored on a computer-readable medium external to the consumer device.

22. The method of claim 14, wherein the primary application image is different than the back-up application image.

23. The method of claim 14, wherein the consumer device comprises a personal media broadcasting system.

24. A consumer device comprising:

a processor;

a memory, said memory comprising a first application image, a second application image, and a bootloader, the bootloader comprising processor code for performing the method comprising:

determining if the first application image is corrupt;

responsive to determining that the first application image is not corrupt, executing the first application image on the processor; and

responsive to determining that the first application image is corrupt, executing the second application image on the processor.

25. The consumer device of claim 24, wherein the memory further comprises a flag, and wherein determining if the first application image is corrupt comprises reading the flag.

26. The consumer device of claim 24, wherein determining if the first application image is corrupt comprises determining a checksum of the first application image.

27. The consumer device of claim 24, wherein the method further comprises:

responsive to determining that the first application image is corrupt, updating the first application image.

28. The consumer device of claim 24, wherein the first application image is different than the second application image.

29. The consumer device of claim 24, wherein the consumer device comprises a personal media broadcasting system.

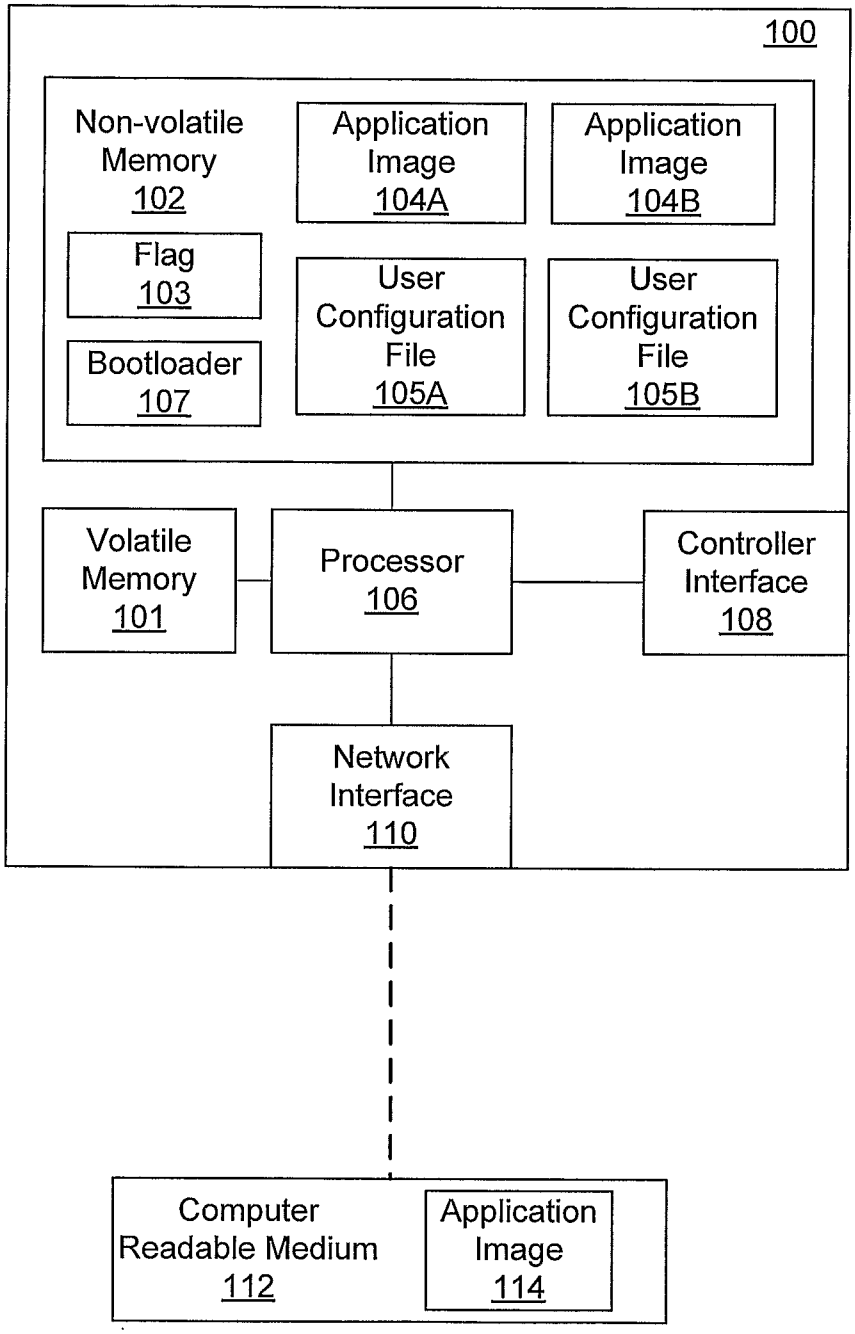

100
Non-volatile Memory 102
Application Image 104A
Application Image 104B
Flag 103
User Configuration File 105A
User Configuration File 105B
Bootloader 107
Volatile Memory 101
Processor 106
Controller Interface 108
Network Interface 110
Computer Readable Medium 112
Application Image 114

FIG. 1

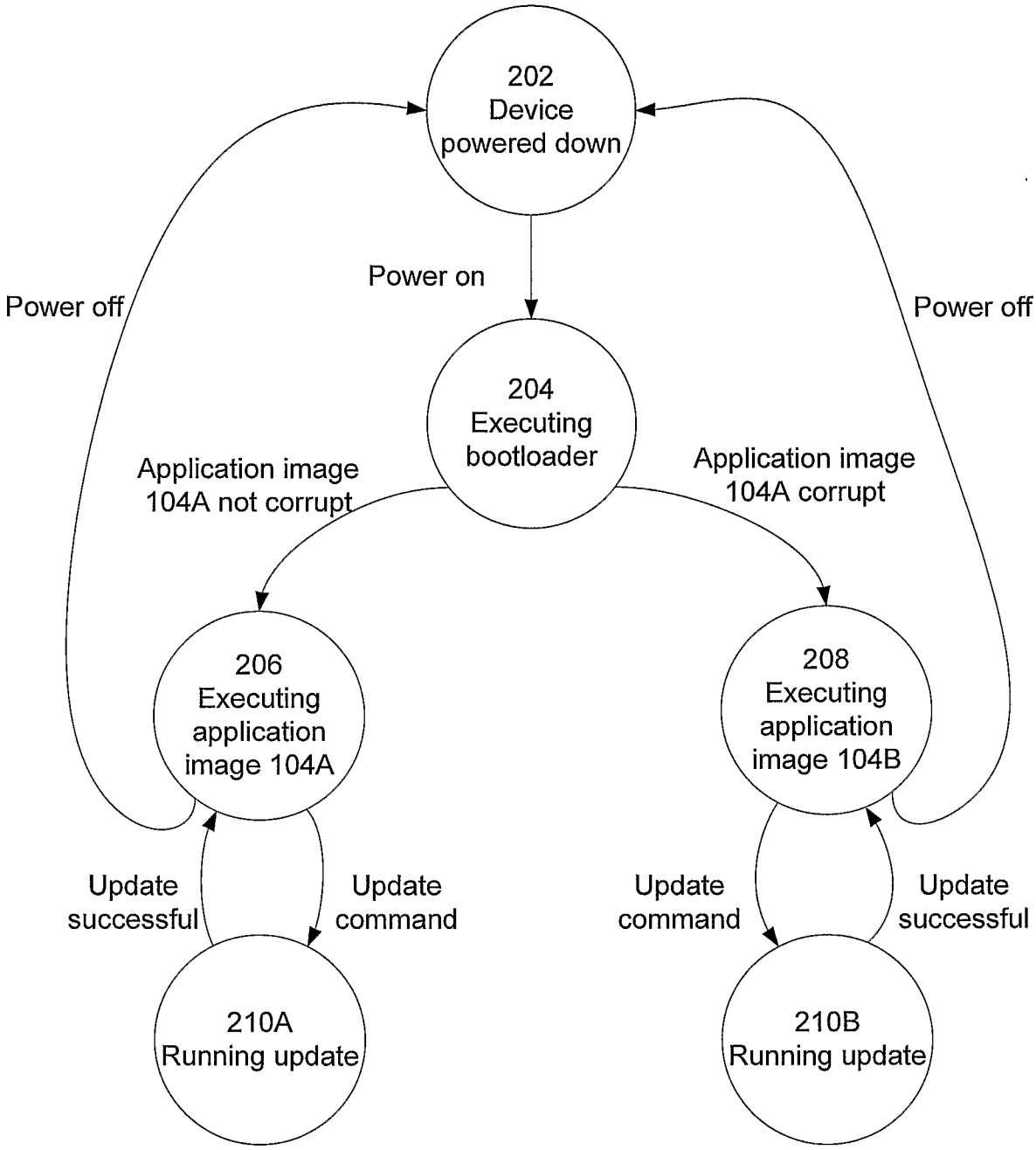

202
Device
powered down
Power on
Power off
Power off
204
Executing
bootloader
Application image
104A not corrupt
Application image
104A corrupt
206
Executing
application
image 104A
208
Executing
application
image 104B
Update
successful
Update
command
Update
command
Update
successful
210A
Running update
210B
Running update

FIG. 2

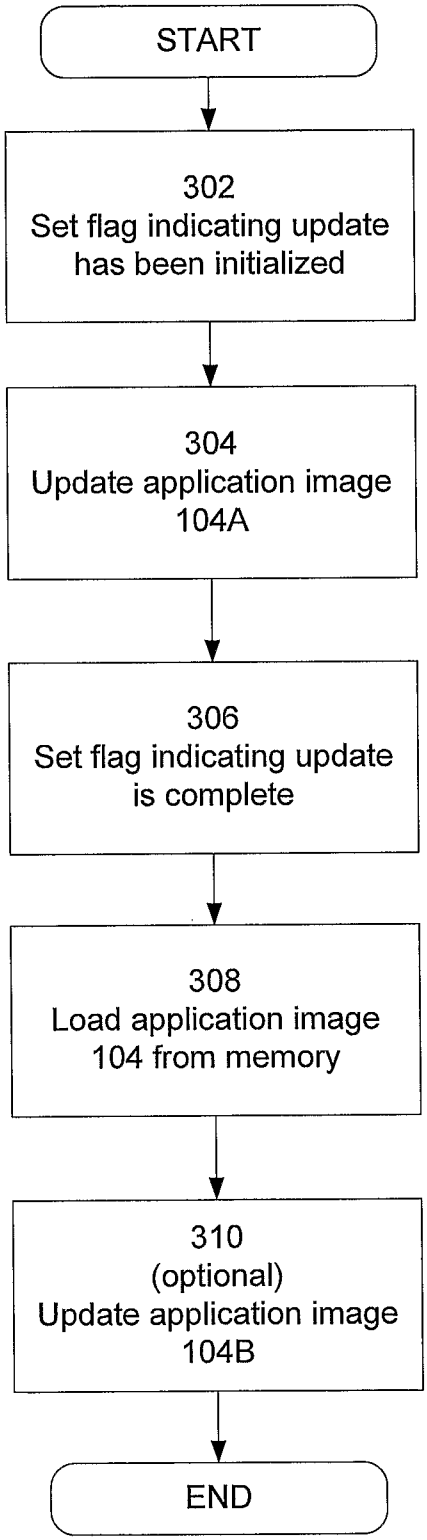

START
302
Set flag indicating update has been initialized
304
Update application image 104A
306
Set flag indicating update is complete
308
Load application image 104 from memory
310
(optional)
Update application image 104B
END

FIG. 3

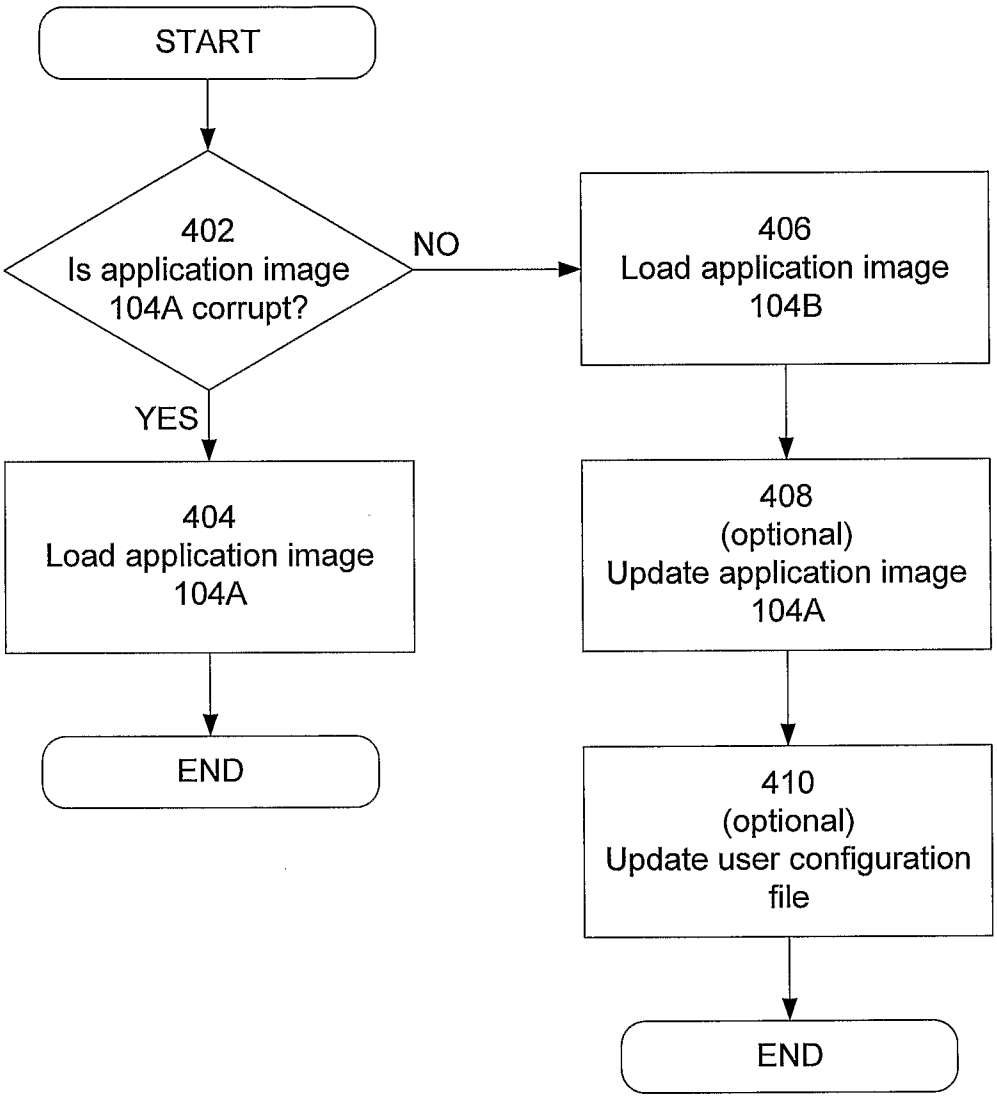

START
402
Is application image 104A corrupt?
NO
YES
406
Load application image 104B
404
Load application image 104A
408
(optional)
Update application image 104A
END
410
(optional)
Update user configuration file
END

FIG. 4