Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2741595 A1 2010/05/06

(21) 2 741 595

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(86) Date de dépôt PCT/PCT Filing Date: 2008/10/27
(87) Date publication PCT/PCT Publication Date: 2010/05/06
(85) Entrée phase nationale/National Entry: 2011/04/26
(86) N° demande PCT/PCT Application No.: JP 2008/069839
(87) N° publication PCT/PCT Publication No.: 2010/050037

(51) Cl.Int./Int.Cl. *B41J 29/38* (2006.01)

(71) Demandeurs/Applicants:
KABUSHIKI KAISHA SATO, JP;
KABUSHIKI KAISHA SATO CHISHIKI ZAISAN KENKYUSYO, JP

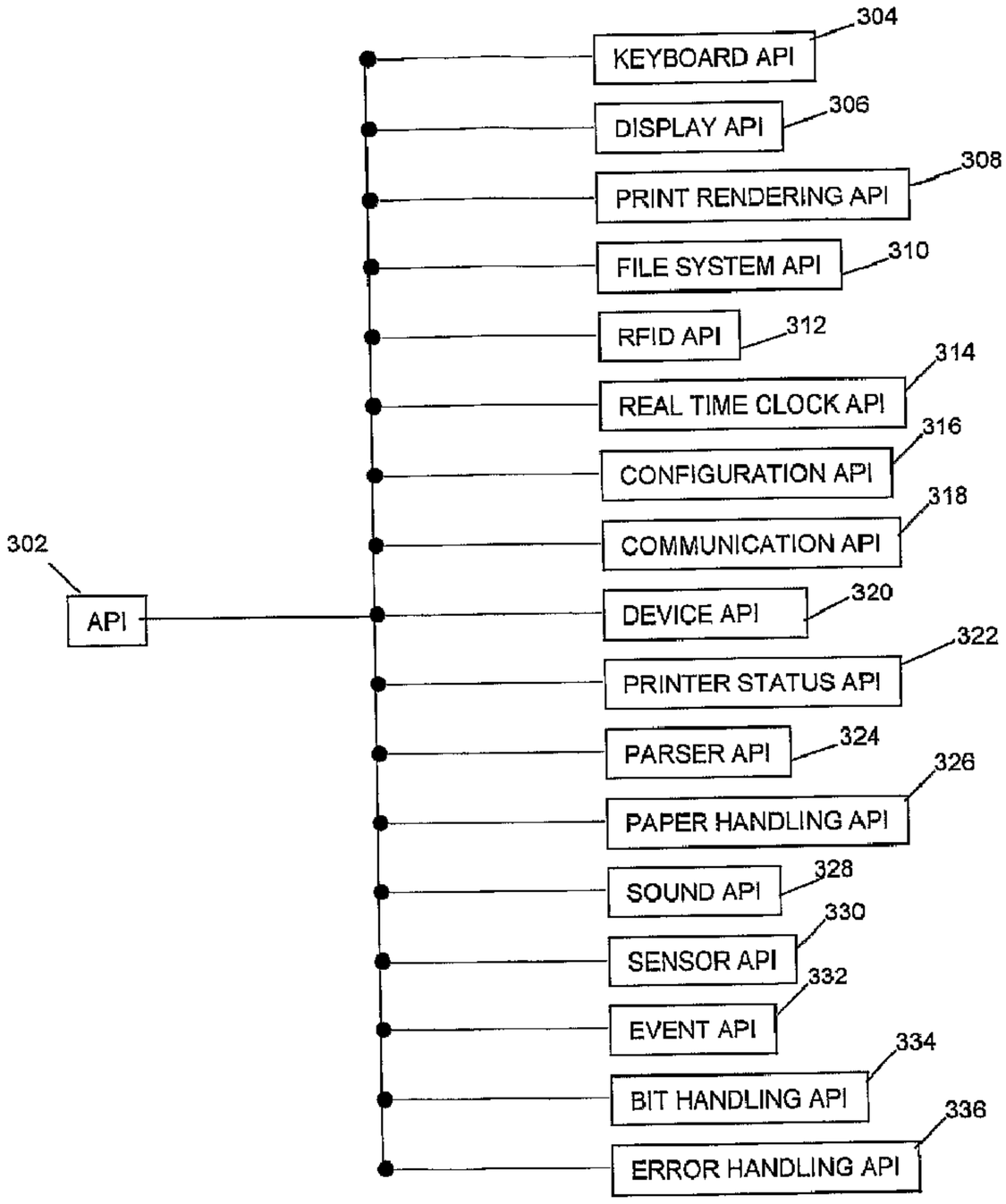
(72) Inventeurs/Inventors:
BERG, LARS-AKE, SE;
HEDBERG, MATS, SE

(74) Agent: BERESKIN & PARR LLP/S.E.N.C.R.L.,S.R.L.

(54) Titre : INTERFACE DE PROGRAMMATION D'APPLICATION D'IMPRIMANTE D'ETIQUETTE UTILISANT UN LANGAGE DE SCRIPT DE PROGRAMME

(54) Title: LABEL PRINTER API USING PROGRAM SCRIPTING LANGUAGE

FIG. 3



(57) Abrégé/Abstract:
A system and method are defined for modifying functionality of a printer that is provided with firmware for controlling printing operations. First programming code is developed that, when executed on a computer readable medium, interfaces with the

CA 2741595 A1 2010/05/06

(21) 2 741 595

(13) A1

(57) Abrégé(suite)/Abstract(continued):
printer's firmware and modifies the functionality of the printer. The first programming code is written in a first programming language, such as the LUA programming language, and the firmware is written in a second programming language other than the first programming language. Thereafter, the first programming code is executed on the computer readable, and the functionality of the printer is modified as a function of the executed first programming code interfacing with the firmware. The firmware is not modified by the interfacing.

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 May 2010 (06.05.2010)

PCT

(10) International Publication Number
WO 2010/050037 A1

(51) International Patent Classification:
B41J 29/38 (2006.01)

(21) International Application Number:
PCT/JP2008/069839

(22) International Filing Date:
27 October 2008 (27.10.2008)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicants *(for all designated States except US)*: **KABUSHIKI KAISHA SATO CHISHIKI ZAISAN KENKYUSYO** [JP/JP]; 9-10, Ebisu 4-chome, Shibuya-ku, Tokyo, 1500013 (JP). **KABUSHIKI KAISHA SATO** [JP/JP]; 9-10, Ebisu 4-chome, Shibuya-ku, Tokyo, 1500013 (JP).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **BERG, Lars-Ake** [SE/SE]; c/o SATO TECHNOLOGY & BUSINESS DEVELOPMENT CENTRE AB, 91, Molndalsvagen, Goteborg, S-41263 (SE). **HEDBERG, Mats** [SE/SE]; c/o SATO TECHNOLOGY & BUSINESS DEVELOPMENT CENTRE AB, 91, Molndalsvagen, Goteborg, S-41263 (SE).

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

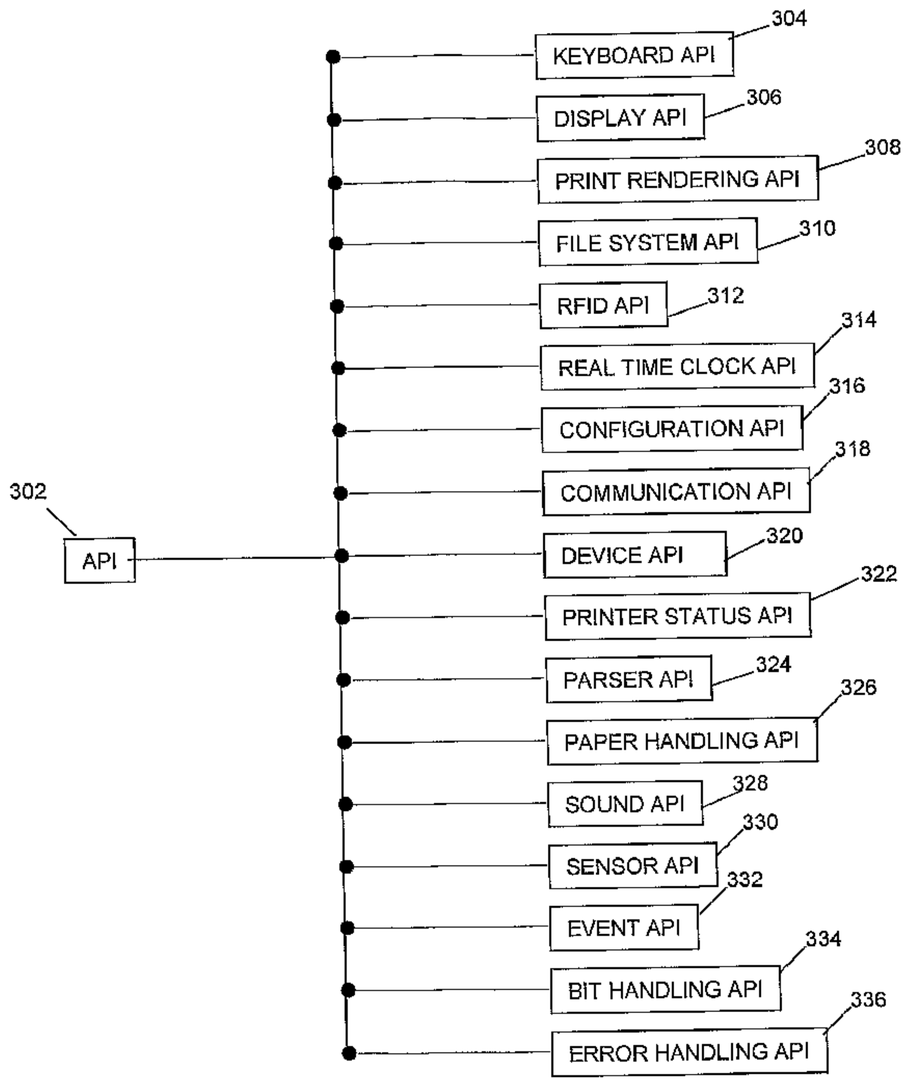
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI

[Continued on next page]

(54) Title: LABEL PRINTER API USING PROGRAM SCRIPTING LANGUAGE

(57) Abstract: A system and method are defined for modifying functionality of a printer that is provided with firmware for controlling printing operations. First programming code is developed that, when executed on a computer readable medium, interfaces with the printer's firmware and modifies the functionality of the printer. The first programming code is written in a first programming language, such as the LUA programming language, and the firmware is written in a second programming language other than the first programming language. Thereafter, the first programming code is executed on the computer readable, and the functionality of the printer is modified as a function of the executed first programming code interfacing with the firmware. The firmware is not modified by the interfacing.

FIG. 3



WO 2010/050037 A1

WO 2010/050037 A1

(BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *of inventorship (Rule 4.17(iv))*

Published:

— *with international search report (Art. 21(3))*

DESCRIPTION

LABEL PRINTER API USING LUA PROGRAM SCRIPTING LANGUAGE

BACKGROUND

Field of the Invention

[0001] The present invention relates generally to printers and, more particularly, to enhancing printer functionality by interfacing with label and tag printer firmware.

Description of the Related Art

[0002] Most label and tag printers operate via a conventional predefined proprietary control language. For example, SATO Barcode Printer Language ("SBPL"), ZEBRA programming language ("ZPL"), DATAMAX Printer Language ("DPL"), INTERMEC Printer Language ("IPL") or the like all include proprietary functionality that requires a fairly significant minimum skill level in order to enable a user to effect changes in the printers' output functionality. Various printer models, such as provided by vendors of printers supporting the printer languages listed above, can perform printing tasks defined by the predefined individual commands. Specific changes to a printer's functionality are typically made by changing the firmware of the printer.

[0003] Since each printer's respective control language is predefined, exiting printers can only perform predefined tasks. In case various printer functionality needs to be added, then the firmware needs to be accordingly changed. This often involves making complex source code revisions and embedded programming development tools to make the revisions. This is not a flexible approach and customization cannot be done locally by the customer.

SUMMARY

[0004] In the prior art, there is no ability to customize or otherwise modify label/tag printer software without changing the printer's firmware source code, and without expensive development tools.

[0005] Accordingly, a system and method are defined for modifying functionality of a printer that is provided with firmware for controlling printing operations. First programming code is developed that, when executed on a computer readable medium, interfaces with the printer's firmware and modifies the functionality of the printer. The first programming code is written in a first programming language, such as the LUA programming language, and the firmware is written in a second programming language other than the first programming language. Thereafter, the first programming code is executed on the computer readable medium, and the functionality of the printer is modified as a function of the executed first programming code interfacing with the firmware. The firmware is not modified by the interfacing.

[0006] In a preferred embodiment, an interpreted script language is provided in connection with a label and tag printer and used to develop printer software applications (i.e., printer application programs) that interface and interact with the printer's firmware through an application programming interface (API).

[0007] Preferably, a label and tag printer is equipped with the LUA virtual machine. The LUA script language is preferably used to develop printer applications adapted for label and tag printers that interface with the firmware of the printer through the application programming interface (API), and allows access to all firmware and hardware functions of the printer.

[0008] Other features and advantages of the present invention will become apparent from the following description of the invention that refers to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] For the purpose of illustrating the invention, there is shown in the drawings a form which is presently preferred, it being understood, however, that the invention is not limited to the precise arrangements and instrumentalities shown. The features and advantages of the present invention will become apparent from the following description of the invention that refers to the accompanying drawings, in which:

[0010] Fig. 1 illustrates an example hardware arrangement, in accordance with a preferred embodiment;

[0011] Fig. 2 illustrates the functional elements of an example information processor shown in Fig. 1;

[0012] Fig. 3 is a block diagram that illustrates printer APIs provided in accordance with a preferred embodiment;

[0013] Fig. 4 illustrates an example label printed by a label and tag printer and defined using APIs provided in accordance with a preferred embodiment;

[0014] Fig. 5 illustrates an example data field size indicating respective horizontal and vertical positions and defined for data field to be provided on a label;

[0015] Fig. 6 illustrates another example label printed by a label and tag printer and defined using APIs and representing color options provided in accordance with a preferred embodiment; and

[0016] Figs. 7-10 illustrate an implementation of a preferred embodiment that includes a plurality of electronic memory formats.

DESCRIPTION OF EMBODIMENTS

[0017] In accordance with the teachings herein, one or more interfaces are provided enabling individuals to customize label and tag printer settings without a need to update, change or otherwise alter firmware source code provided with the label and tag printer, and without a need for expensive development tools. Thus, a label and tag printer is effectively enabled for application development without a need for having others to write applications using the printer's proprietary interpreted language.

[0018] In a preferred embodiment, an interpreted script language is provided in combination with a label and tag printer. A set of printer extensions formatted as printer application programming interfaces ("APIs") are provided to enable interaction and manipulation with printer firmware for custom and extended functionality. The APIs are preferably provided for label and tag printers. In a preferred embodiment, the interpreted scripting language is LUA.

[0019] By interfacing the printer firmware with the set of printer APIs, for example, developed in the scripting language, users of label and tag printers are afforded improved flexibility and accessibility via software commands used by the label and tag printer to perform various tasks. Using APIs in combination with the printer's firmware, users are no longer required to change a label and tag printer's firmware source code or access to development tools typically required for changing or modifying the label and tag printer's firmware source code. As described in greater detail below, the APIs provided in connection with the teachings herein support various functionality, such as interfacing different keyboards, displays, providing variable fonts and formats of print rendering, accessing printer file systems, printer configurations and various other functions related to label and tag printing.

[0020] Thus, the present invention provides a plurality of printing solutions without a need for firmware source code modifications or expensive embedded systems including,

for example, software development tools for modifying predefined proprietary control and printer firmware.

[0021] Referring now to the drawing figures, in which like reference numerals represent like elements, Fig. 1 illustrates an example hardware arrangement, in accordance with an embodiment of the present invention, for providing and installing printer application programs either via a direct cable connection or over a communication network, and referred herein, generally, as system 100. In the example shown in Fig. 1, information processor(s) 102 are provided with an integrated development programming environment ("IDE"), such as to develop applications in the LUA programming language, as known to those skilled in the art. Information processor 102 preferably includes all databases necessary to support the present invention. However, it is contemplated that information processor 102 can access any required database via communication network 106 or any other communication network to which information processor 102 may be coupled. Communication network 106 is preferably a global public communication network such as the Internet, but can also be a wide area network (WAN), local area network (LAN), an intranet or other network that enables computing devices and peripheral devices to communicate.

[0022] In a preferred embodiment, information processor 102 are any computer readable medium devices that are capable of sending and receiving data across communication network 106, e.g., mainframe computers, mini computers, personal computers, laptop computers, a personal digital assistants (PDA), cellular telephones and Internet access devices such as Web TV. In addition, information processors 102 are preferably equipped with web browser software, such as MICROSOFT INTERNET EXPLORER, MOZILLA FIREFOX, or the like. Information processors 102 are coupled to communication network 106 using any known data communication networking technology.

[0023] Also shown in Fig. 1 is printer 108 that is preferably a label and tag printer and operable to print labels and tags of data received from information processors 102. Label

and tag printer 108 may be provided with keyboard 110 and display 112 to enable input and output functionality with label and tag printer 108 in the absence of or in conjunction with information processor 102.

[0024] Fig. 2 illustrates the functional elements of an example information processor 102, and includes one or more central processing units (CPU) 202 used to execute software code and control the operation of information processor 102. Other elements include read-only memory (ROM) 204, random access memory (RAM) 206, one or more network interfaces 208 to transmit and receive data to and from other computing devices across a communication network, storage devices 210 such as a hard disk drive, floppy disk drive, tape drive, CD ROM or DVD for storing program code databases and application data, one or more input devices 212 such as a keyboard, mouse, track ball, microphone and the like, and a display 214. Further, one or more of functional elements 202-214 may be suitably configured or provided with label and tag printer 108, as well.

[0025] The various components of information processor 102 need not be physically contained within the same chassis or even located in a single location. For example, storage device 210 may be located at a site which is remote from the remaining elements of information processor 102, and may even be connected to CPU 202 across communication network 106 via network interface 208. Information processor 102 preferably includes a memory equipped with sufficient storage to provide the necessary databases, forums, and other community services as well as acting as a web server for communicating hypertext markup language (HTML), Java applets, Active-X control programs. Information processors 102 are arranged with components, for example, those shown in Fig. 2, suitable for the expected operating environment of information processor 102. The CPU(s) 202, network interface(s) 208 and memory and storage devices are selected to ensure that capacities are arranged to accommodate expected demand.

[0026] The nature of the invention is such that one skilled in the art of writing computer executable code (i.e., software) can implement the functions described herein using one or more of a combination of popular computer programming languages and

developing environments including, but not limited to, LUA, C, C++, Visual Basic, JAVA, HTML, XML, ACTIVE SERVER PAGES, JAVA server pages, servlets, MYSQL and PHP.

[0027] Although the present invention is described by way of example herein and in terms of a web-based system using web browsers and a web site server (e.g., information processor 102), system 100 is not limited to such a configuration. It is contemplated that system 100 is arranged such that label and tag printer 108 communicates with and outputs data received from information processor 102 using any known communication method, for example, using a non-Internet browser WINDOWS viewer coupled with a local area network protocol such as the Internet Packet Exchange (IPX), dial-up, third-party, private network or a value added network (VAN).

[0028] It is further contemplated that any suitable operating system can be used on information processor 102, for example, DOS, WINDOWS 3.x, WINDOWS 95, WINDOWS 98, WINDOWS NT, WINDOWS 2000, WINDOWS ME, WINDOWS CE, WINDOWS POCKET PC, WINDOWS XP, MAC OS, UNIX, LINUX, PALM OS, POCKET PC and any other suitable operating system.

[0029] In a preferred embodiment, label and tag printer 108 applications are developed in the LUA programming language. In accordance with a preferred embodiment, a LUA interpreter is included that is operable to process LUA programming statements provided in the applications. Unlike typical high-level programming languages, LUA is a scripting language and not a basic programming language. Unlike typical label and tag printers that receive SBPL commands or other predefined commands to cause the label and tag printer to execute various functionality, the present invention implements printer APIs that are extensions to the LUA language that interfaces with the printer firmware and that is interpreted by the LUA interpreter. This enables more advanced and complex applications to be developed such as via function calls. Furthermore, by implementing APIs that are extensions to the LUA programming language, a portion of label and tag printer 108 functionality can be modified, as opposed

to prior art label and tag printers that require a complete overwrite of their respective firmware applications in order to implement a single change.

[0030] Another benefit of the present invention is that applications developed in the LUA programming language and implemented in label and tag printer 108 APIs are more compact and execute significantly faster. This is because prior art label and tag printers require more software instructions in order to implement various functions that otherwise are accomplished by APIs of the present invention. Further, the LUA interpreter interfaces with the APIs of the present invention efficiently, such as by taking advantage of function calls, variable declarations or the like, and code utilization is significantly improved over prior art label and tag printers, as a function of fewer memory requirements.

[0031] Another advantage of the LUA printer APIs of the present invention over prior art label and tag printers is an ability to access print functions without requiring an interpreter to access the print functions via an existing control language, such as SBPL or ZPL. In other words, the APIs directly interface with the label and tag printer 108 firmware that controls the printer's 108 printing functionality. This is an improvement over prior art systems that require, for example, a generation of SBPL commands to interface with an existing control language, thereby increasing steps and overhead associated with modifying printer functionality.

[0032] Moreover, since the APIs of the present invention interface via a LUA interpreter, the size limitations associated with the application are virtually eliminated with the exception of internal memory requirements based on a particular label and tag printer 108. This is an improvement over prior art label and tag printers that have, for example, limitations on the number of lines of code that can be installed and do not support dynamic memory allocation. APIs of the present invention support, for example, function calls and other high level programming language functionality, such as releasing memory ("garbage collection") that increases the amount of programming code implemented for a particular functionality.

[0033] Another benefit of the present invention is that development tools, such as MICROSOFT VISUAL STUDIO or other integrated development environments ("IDEs") are usable as a plug-in component for LUA. Application development for respective label and tag printers 108 is more attractive for developers who are familiar with or comfortable with developing software applications in visual integrated development environments.

[0034] Another benefit of the present invention is that LUA applications can be run on computers having a LUA virtual machine that simulates printer operations. The user can test developed LUA applications on computers prior to downloading and installing the applications on printers, which precludes a need to actually operate the printers to test the LUA applications.

[0035] In a preferred embodiment, a plurality of independent LUA virtual machines can be operable in a sort of chain and implemented on a single label and tag printer 108 or a network thereof. In this way, configurable dynamic local settings can be implemented, for example, for bar code printers that sort data according to various custom settings such as, for example, regional settings and language settings. The various local settings may be stored on a particular computer system's operating system, and changes to the behavior may depend upon, for example, the local settings.

[0036] By implementing APIs via a LUA interpreter, additional printer functionality can be provided beyond that previously available in the prior art. A discussion regarding additional printer functionality and improved implementations thereof afforded in accordance with a preferred embodiment is now provided.

[0037] Preferably, programming function calls and an ability to declare, address and pass values to and from programming functions via variables is supported via the LUA implementation in an example embodiment. Preferably, programming functions return a value, such as an error code, that represents whether a function executed successfully or not. In case, for example, a function does not execute as intended, an error code is

returned that represents the cause of the error, the symptom of the error, the result of the error, or other suitable information. Moreover, function names and variable names, such as related to table names, string values, dates, numeric values or the like, are preferably not displayed readily to users and may be displayed, at least partially hidden or completely hidden from view.

[0038] Preferably, security implementations are supported by the teachings herein, including, for example, requiring users to have provided sufficient authorization and granted access rights to perform various tasks, such as to access particular data, one or more data directories, or to create, remove or other modify data directories, data files or the like.

[0039] Another feature supported by APIs is an interface rendering. As used herein, rendering refers, generally, to creation of interactive programming objects. For example, rendered objects may be formed as data fields (e.g., text-based fields), barcodes, graphic fields. The rendered objects include one or more properties that can be manipulated, such as by methods. Preferably, objects (e.g., text field, barcode and graphic objects) rendered via APIs are provided (e.g., added or otherwise embedded) with a label object and printed via label and tag printer 108.

[0040] Moreover, APIs support providing objects, such as barcodes, positioned on a label at least partially outside of the printable region of label and tag printer 108 without causing an error while developing the label, while printing the label or both. This feature makes it possible to develop a label having, for example, a barcode in which only half of the bar code is printed by label and tag printer 108. This feature provides an improvement over prior art methods of modifying printer functionality in case a user desires a partial or otherwise incomplete object, such as a barcode, to be printed on a label via label and tag printer 108.

[0041] In one embodiment and during operation, a respective position of a rendered object, such as a text field, barcode, graphic box, image or the like, is defined by a horizontal and a vertical position parameter (e.g., “hPos” and “vPos”). The parameter

values preferably define an anchor point position for an object. In case hPos and vPos are properly set, an error parameter, (e.g., "E_SUCCESS") is defined. Alternatively, if hPos or vPos is not properly set, a different error parameter (e.g., "EPARAM") is defined.

[0042] In addition to parameters defined for positioning, the present invention supports magnification of objects, such as a barcode or an image. For example, horizontal and vertical magnification parameters (e.g., "hMag" and "vMag") are defined for horizontal and vertical pixel magnification to set (e.g., from values 1-12), which represents respective degrees of horizontal and vertical magnification of an object.

[0043] Fig. 3 is a block diagram that illustrates associated APIs 302 provided in accordance with a preferred embodiment to interface with label and tag printer 108 firmware. As shown in Fig. 3, keyboard API 304 is operable to receive and interpret (i.e., read) signals from keyboard 110 integrated with label and tag printer 108. Alternatively, keyboard API 304 operates to read and interpret signals from an external keyboard or other input device 212 not directly coupled to label and tag printer 108. Display API 306 operates to write a wide variety of textual and graphical content to display 112 integrated with printer 108. Alternatively, display API 306 operates to write textual and graphical content to an external display 214 attached to label and tag printer 108. Preferably, display API 306 supports a wide selection of fonts and coding types, for example, for many different written languages.

[0044] Continuing with reference to Fig. 3, print rendering API 308 supports user-defined data field elements to be output on a label printed by label and tag printer 108. Examples of such data field elements include textual data fields, 1-D and 2-D barcodes, lines and boxes, trigonometric functions, images, individual pixels, graphics, formatted dates, and counter values. Printer rendering API 308 enables a user to define a type of data field and a respective position for the data field to be output on a particular label/tag. Preferably, selectable options for fonts and coding types that support various languages are provided by printer rendering API 308. Fixed length or variable length data formats are preferably supported, and includable in a field definition or input from a file or

communication interface via printer rendering API 308. Other features include support for selectable color output options for defining a color of one or more of the above-described field objects. Moreover, one or more printouts or feed commands are issuable as a function of print rending API 308 that to output blank or printed labels and tags. Preferably, a user-selectable print quality control feature is further provided by print rendering API 308.

[0045] Further, file system API 310 is preferably provided to enable a user to store, delete, read and write to files that are located in one or more of label and tag printer's 108 read only memory file system, random access memory file system, flash memory file system, or external memory sources, such as compact flash memory cards, secure digital memory cards, USB memory devices, or the like. Providing a user with access to data, files or the like that are stored in various internal and external sources associated with label and tag printer 108 significantly increases flexibility for users to control and manipulate label and tag printer 108 operation. Examples of various functionality preferably provided in connection with file system API 310 include formatting label and tag printer's 108 file system, determine entries in iterations of a directory, navigating to a particular directory, create a new directory, copy files, remove a directory or file, determine used and available bytes in the file system, and change access rights to a file or directory,

[0046] Continuing with reference to Fig. 3, radio frequency identification ("RFID") API 312 that supports read and write access to a RFID chip/inlay provided with label and tag printer 108. Additionally, real time clock API 314 enables a user to define and read date and time data to and from label and tag printer 108. Configuration API 316 supports user-defined printer-specific parameters. For example, configuration API 316 enables a user to define communication parameters associated with print speed, quality, date and time functions, local languages, menu control or the like. Configuration API 316 is particularly useful to enable a user to define or modify operating controls for label and tag printer 108 that are typically exclusive to firmware provided with label and tag printer 108 and modifiable only by technically proficient users having specialized software and

skills. Additionally, communication API 318 preferably controls communication with external I/O interface devices. A plurality of communication protocols and standards are supported, including, for example, RS232, RS485, RS422, USB, LAN, WLAN and external digital of relay interface.

[0047] Other APIs 302 shown in Fig. 3 include device API 320 that is operable to control devices in the printer, printer status API 322 that is operable to report the status of the printer at any given time, and parser API 324 that is operable to parse commands and/or files, such as XML commands and/or XML files that have been sent to label and tag printer 108. Once parsed, the XML commands can be interpreted and used to control output provided by label and tag printer 108.

[0048] In addition to device API 320, printer status API 322, and parser API 324, APIs 302 preferably include paper handling API 326 is provided to support a variety of paper functions, including for example, print feed, form feed, line feed, test feed or the like, for one or a plurality of label and tag sizes. Additionally, sound API 328 is shown that provides audio controls, such as for a beeper, buzzer or other sound device in label and tag printer 108. Moreover, sensor API 330 is shown that is operable to receive information from sensor devices, such as a label gap sensor and label mark sensor, that are provided with label and tag printer 108, and operable to determine various conditions, such as when an end of a label is reached, when an end of a ribbon is reached, and when an end of a label or ribbon is almost reached. In one embodiment, sensor API 330 operates to emit a warning when a determination of one or more of these conditions occurs. Other APIs 302 shown in Fig. 3 include event API 332 that receives and handles various events that occur in label and tag printer 108, bit handling API 334 that is operable to perform bit manipulation of data, as necessary, and error handling API 336 that is operable to handle errors that may occur with label and tag printer 108, such as a power outage, a memory error, a paper jam or the like.

[0049] Thus and in accordance with a preferred embodiment, a plurality of APIs 302 are developed, for example, in the LUA programming language or in the C programming

language, and implemented in label and tag printer 108 without a need for interpreter accessing print functions via an existing control language, such as SBPL or ZPL. In addition to the APIs 302 illustrated in Fig. 3, various other miscellaneous functions are envisioned herein to be implemented in accordance with one or more embodiments. For example, functionality is supported for cloning a table and/or meta tables for rapid and ease of development. Other examples include functionality for determining an error code value as a function of a turned error string value (e.g., "err2str()"), functionality for saving tables in a one or more of a plurality of formats (e.g., XML, LUA or other), functionality for loading a table provided a plurality of formats, and support for multiple written and spoken languages for menus and prompts.

[0050] Fig. 4 illustrates an example label 400 printed by label and tag printer 108, defined using APIs 302 and provided in accordance with a preferred embodiment. As shown in Fig. 4, anchor points 402 are defined in the upper leftmost (e.g., defined via hMag and vMag variable values) and position of textual data printed on label 400, notwithstanding the respective orientation or position of the printed textual data. Also shown in Fig. 4 is paper feed direction 404 of label 400 as it is printed via label and tag printer 108. In addition to textual data printed on label 400, graphical image 406 is provided, such as rendered via print rendering API 308.

[0051] Fig. 5 illustrates an example data field size defined for data field 500 to be provided on label 400 and indicates respective horizontal and vertical positions ("hPos" and "vPos") 502 for the upper leftmost corner of data field 500 and horizontal and vertical positions for the lower rightmost position 504. Moreover, data orientation function 506 (e.g., dir(0,.359)) indicates the relative orientation of data field 500 as it is output on label 400.

[0052] Fig. 6 illustrates another example label 600 printed by label and tag printer 108, defined using APIs 302 and provided in accordance with a preferred embodiment. In the example label 600 shown in Fig. 6, a plurality of colors 602, 604 and 606 are shown that are provided on label 600 at respective positions 608. Thus, as shown and described

herein, the LUA API is an interface that operates in conjunction with a label and tag printer's 108 firmware and the LUA interpreter by implementing APIs via the LUA interpreter, customers can create LUA application programs that implement preferred printing operations.

[0053] Referring now to Fig. 7, in a preferred embodiment label and tag printer 108 is provided with two types of memory: flash memory 702 and synchronous dynamic random access memory (SDRAM) 704. Flash memory 702 (preferably used in lieu of ROM) stores, among other suitable data, label and tag printer's 108 boot code, base firmware (e.g., drivers, barcode formats or the like), LUA API and the LUA virtual machine. As known in the art, the boot code stored in flash memory 702 operates during the printer's 108 boot-up-process. In a preferred embodiment, the label and tag printer's 108 base firmware and LUA API is stored in flash memory 702 in a compressed format, thereby conserving memory space in flash memory 702.

[0054] Prior to the label and tag printer 108 boot-up process, SDRAM 704 is preferably largely empty, and ready to receive data. As illustrated in Fig. 8, when label and tag printer 108 boots, label and tag printer's 108 firmware, LUA API and LUA virtual machine are preferably stored in flash memory 702 in compressed format. After the boot process, the firmware, LUA API, and LUA virtual machine are uncompressed and provided to SDRAM 704 in the uncompressed format. Thereafter, label and tag printer's 108 base firmware, drivers, bar codes data, label formats, parser, image buffer and other data, as appropriate, are temporarily stored for operation in SDRAM 704.

[0055] With reference to Fig. 9, the LUA virtual machine preferably operates via SDRAM 704 in connection with the printer's 108 firmware, LUA API and parser. In this way and in accordance with a preferred embodiment, label and tag printer 108 does not rely upon flash memory 702 exclusively during printing operations but, instead, operates via uncompressed data and instructions stored in SDRAM 704 for operation. This embodiment is preferable over typical prior art systems and method because SDRAM

704 operates faster and more efficiently than flash memory 702, and SDRAM 704 can purge uncompressed data and instructions after printer operations are complete.

[0056] With reference to Figs. 9 and 10, during printing operations, the LUA APIs provided in connection with the teachings herein operate via the LUA virtual machine and interface with label and tag printer 108 firmware for outputting a label image or other object that is stored in the buffer of the label and tag printer 108. Preferably, label and tag printer 108 can receive instructions (e.g., via a LUA API) and/or data from a plurality of input sources, preferably as defined via the LUA API. For example, data could be scanned, typed or otherwise provided and received via any suitable input port, including input devices 212, communication ports, network interface 208 or the like.

[0057] Thus as described and claimed herein, and shown in the accompanying drawings, functionality provided by label and tag printer 108 is preferably enhanced, modified or added as a function of APIs 302 that interface with printer's 108 firmware. The applications are preferably defined using a high level programming language, such as the LUA programming language, thereby precluding a need for individuals to be proficient in a particular printer model firmware proprietary programming language, or to have access to a proprietary development tool to modify a printer's firmware. Accordingly, a printer is enabled for application development without a need to write applications using a printer's proprietary interpreted language.

[0058] Although the present invention is described and shown in relation to particular embodiments thereof, many other variations and modifications and other uses will become apparent to those skilled in the art. It is preferred, therefore, that the present invention be limited not by the specific disclosure herein.

1. A method for modifying functionality of a printer that is provided with firmware for controlling printing operations, the method comprising:

providing first programming code that, when executed on a computer readable medium, interfaces with the printer firmware and modifies the functionality of the printer;

executing the first programming code on the computer readable medium and interfacing the first programming code with the firmware; and

modifying the functionality of the printer as a function of the executing of the first programming code and the interfacing with the firmware, wherein the firmware is not modified by the interfacing.

2. The method of claim 1, wherein the printer is a label and tag printer.

3. The method of claim 1, wherein the first programming language is a scripting language.

4. The method of claim 3, wherein the scripting language is LUA.

5. The method of claim 4, wherein the executing of the first programming code further comprises interpreting the first programming code in a LUA virtual machine.

6. The method of claim 1, further comprising providing at least one application programming interface.

7. The method of claim 6, wherein the at least one application programming interface includes one or more of a keyboard application programming interface, a display application programming interface, a print rendering application programming interface, a file system application programming interface, a radio frequency identification application programming interface, a real time clock application programming interface, a configuration application programming interface, a communication application programming interface, a device application programming interface, a printer status application programming interface, an

XML parsing application programming interface, a paper handling programming interface, a sound application programming interface, a sensor application programming interface, an event application programming interface, an event application programming interface, a bit handling application programming interface and an error handling application programming interface.

8. The method of claim 1, wherein the first programming code is executed on a computing device other than the printer.

9. The method of claim 1, wherein the first programming code is developed in an integrated development environment.

10. A system for modifying printer functionality, the system comprising:

a printer having the printer functionality;

firmware that is provided with the printer and that, when executed on the printer, controls printing operations;

first programming code that is executable on a computer readable medium, wherein the first programming code includes commands to modify the printer functionality; and

a computer readable medium on which the first programming code is executed;

wherein the first programming code and the firmware interface to modify the functionality of the printer when the first programming code executes on the computer readable medium, and further wherein the firmware is not modified by the interfacing of the first programming code with the firmware.

11. The system of claim 10, wherein the printer is a label and tag printer.

12. The system of claim 10, wherein the second programming language is a scripting language.

13. The system of claim 12, wherein the scripting language is LUA.

14. The system of claim 13, wherein the executing of the first programming code further comprises interpreting the first programming code in a LUA virtual machine.

15. The system of claim 10, further comprising at least one application programming interface.

16. The system of claim 15, wherein the at least one application programming interface includes one or more of a keyboard application programming interface, a display application programming interface, a print rendering application programming interface, a file system application programming interface, a radio frequency identification application programming interface, a real time clock application programming interface, a configuration application programming interface, a communication application programming interface, a device application programming interface, a printer status application programming interface, an XML parsing application programming interface, a paper handling programming interface, a sound application programming interface, a sensor application programming interface, an event application programming interface, an event application programming interface, a bit handling application programming interface and an error handling application programming interface.

17. The system of claim 10, wherein the first programming code is written without the firmware.

18. The system of claim 10, further comprising an integrated development environment for developing the first programming code.

19. A method for modifying functionality of a printer that is provided with firmware for controlling printing operations, the method comprising:

developing first programming code that, when executed on a computer readable medium, interfaces with the printer firmware and modifies the functionality of the printer;

storing the first programming code and the firmware in a compressed format on a first memory provided in the printer;

uncompressing the compressed first programming code and the firmware into a second memory provided with the printer when booting up the printer;

executing the first programming code and the firmware on the second memory and interfacing the first programming code with the firmware; and

modifying the functionality of the printer as a function of the executing of the first programming code and the interfacing with the firmware, wherein the firmware is not modified by the interfacing.

20. The method of claim 19, wherein the first memory is a flash memory.

21. The method of claim 19, wherein the second memory is a SDRAM.

22. A system for modifying printer functionality, the system comprising:

a printer having the printer functionality;

firmware that is provided with the printer that, when executed on the printer, controls printing operations;

first programming code that is executable on a computer readable medium, wherein the first programming code includes commands to modify the printer functionality;

a first memory on which the first programming code and the firmware is stored as compressed format; and

a second memory on which the first programming code and the firmware is executed;

wherein when the printer boots up, the compressed first programming code and the firmware in the first memory is uncompressed into the second memory;

wherein when the first programming code and the firmware executes on the second memory, the first programming code and the firmware interface to modify the functionality of

the printer, and further wherein the firmware is not modified by the interfacing of the first programming code with the firmware.

23. The system of claim 22, wherein the first memory is a flash memory.

24. The system of claim 22, wherein the second memory is a SDRAM.

25. A method for modifying functionality of a printer that is provided with firmware for controlling printing operations, the method comprising:

developing first programming code that, when executed on a computer readable medium, interfaces with the printer firmware and modifies the functionality of the printer;

storing the first programming code on a first memory provided in the printer;

executing the first programming code and the firmware on the second memory and interfacing the first programming code with the firmware; and

modifying the functionality of the printer as a function of the executing of the first programming code and the interfacing with the firmware, wherein the firmware is not modified by the interfacing.

26. A system for modifying printer functionality, the system comprising:

a printer having the printer functionality;

firmware written in a first programming language and provided with the printer that, when executed on the printer, controls printing operations;

first programming code that is executable on a computer readable medium, wherein the first programming code includes commands to modify the printer functionality;

a first memory on which the first programming code is stored; and

a second memory on which the first programming code and the firmware is executed;

wherein when the first programming code and the firmware executes on the second memory, the first programming code and the firmware interface to modify the functionality of the printer, and further wherein the firmware is not modified by the interfacing of the first programming code with the firmware.

27. The method of claim 6, wherein the application programming interface is written in the same programming language as the first programming code.

28. The method of claim 27, wherein the firmware is written in a different programming language than the first programming code.

29. The system of claim 15, wherein the application programming interface is written in the same programming language as the first programming code.

30. The system of claim 29, wherein the firmware is written in a different programming language than the first programming code.

FIG. 1

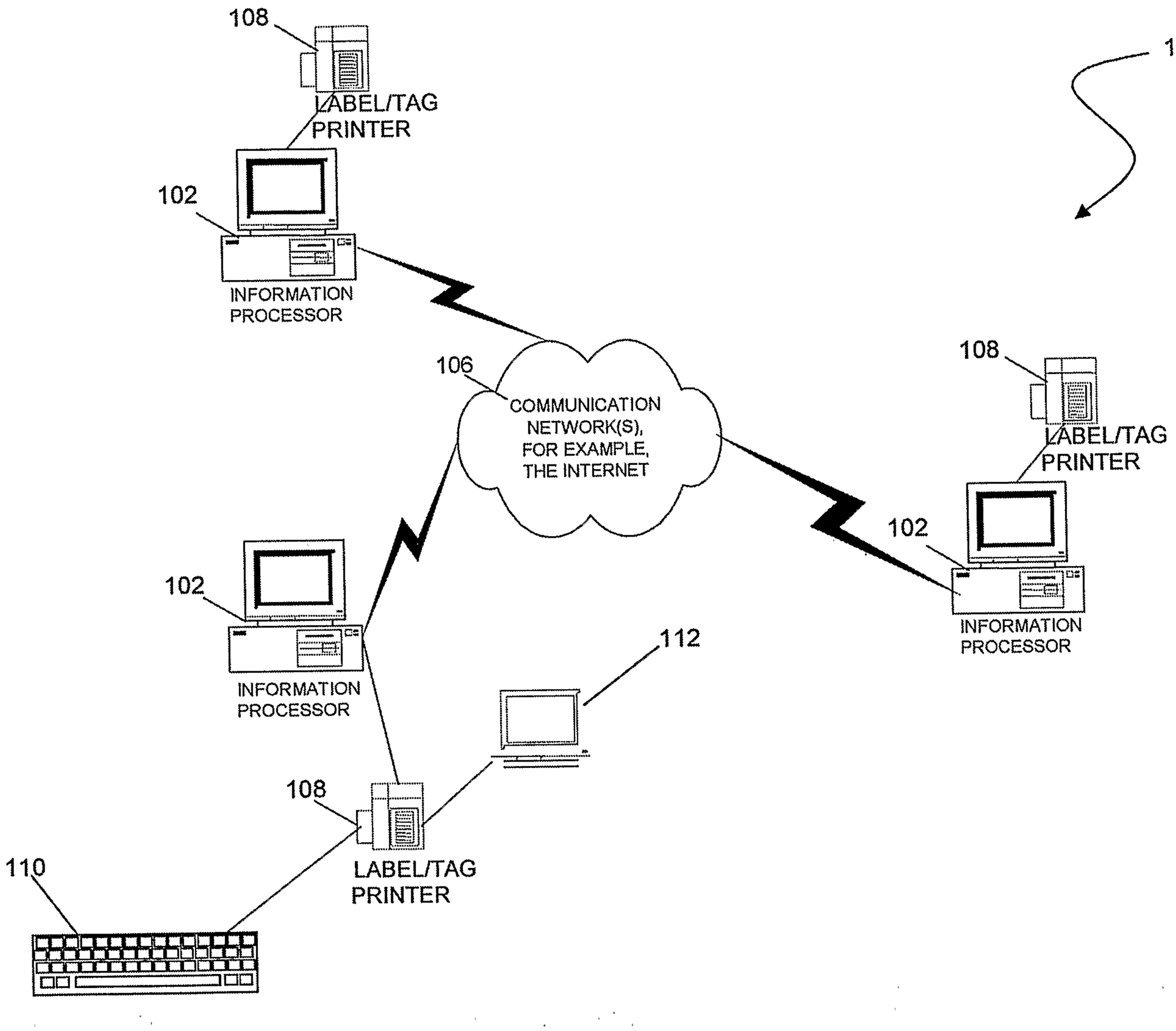

100
108
LABEL/TAG PRINTER
102
INFORMATION PROCESSOR
106
COMMUNICATION NETWORK(S), FOR EXAMPLE, THE INTERNET
108
LABEL/TAG PRINTER
102
INFORMATION PROCESSOR
102
INFORMATION PROCESSOR
112
108
LABEL/TAG PRINTER
110

FIG. 2

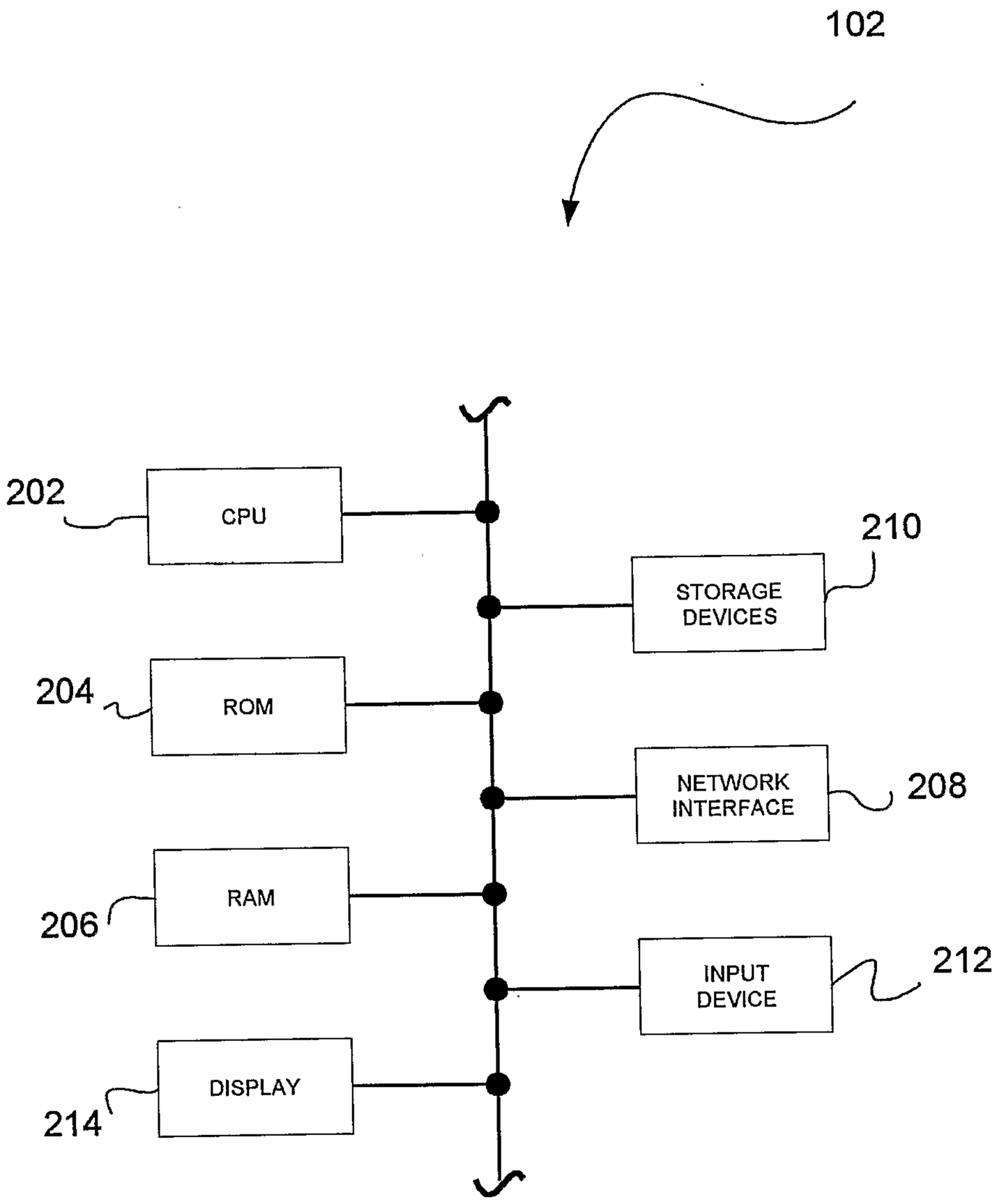

102
202
CPU
210
STORAGE DEVICES
204
ROM
NETWORK INTERFACE
208
206
RAM
INPUT DEVICE
212
214
DISPLAY

FIG. 3

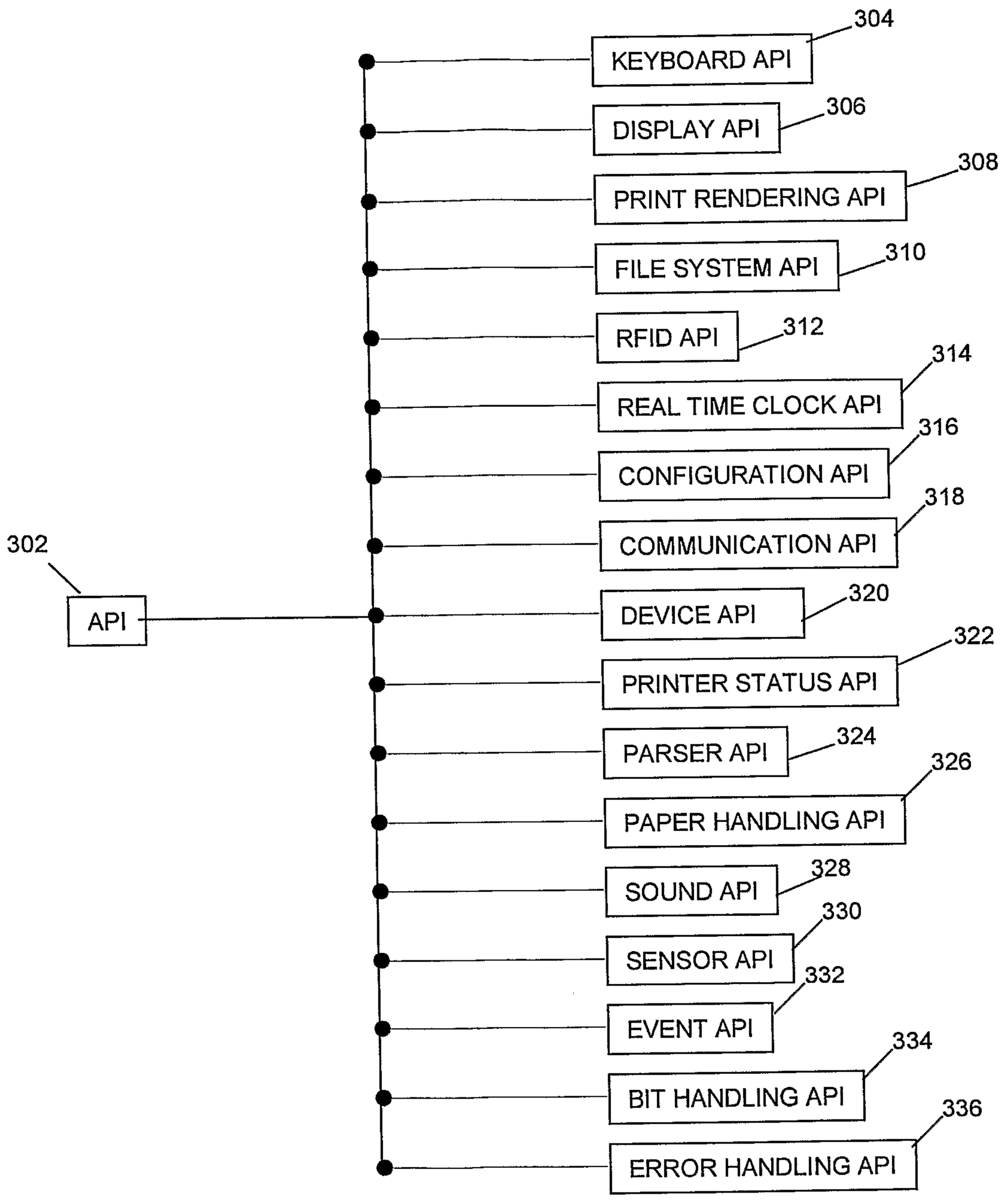

302
API
304
KEYBOARD API
306
DISPLAY API
308
PRINT RENDERING API
310
FILE SYSTEM API
312
RFID API
314
REAL TIME CLOCK API
316
CONFIGURATION API
318
COMMUNICATION API
320
DEVICE API
322
PRINTER STATUS API
324
PARSER API
326
PAPER HANDLING API
328
SOUND API
330
SENSOR API
332
EVENT API
334
BIT HANDLING API
336
ERROR HANDLING API

FIG. 4

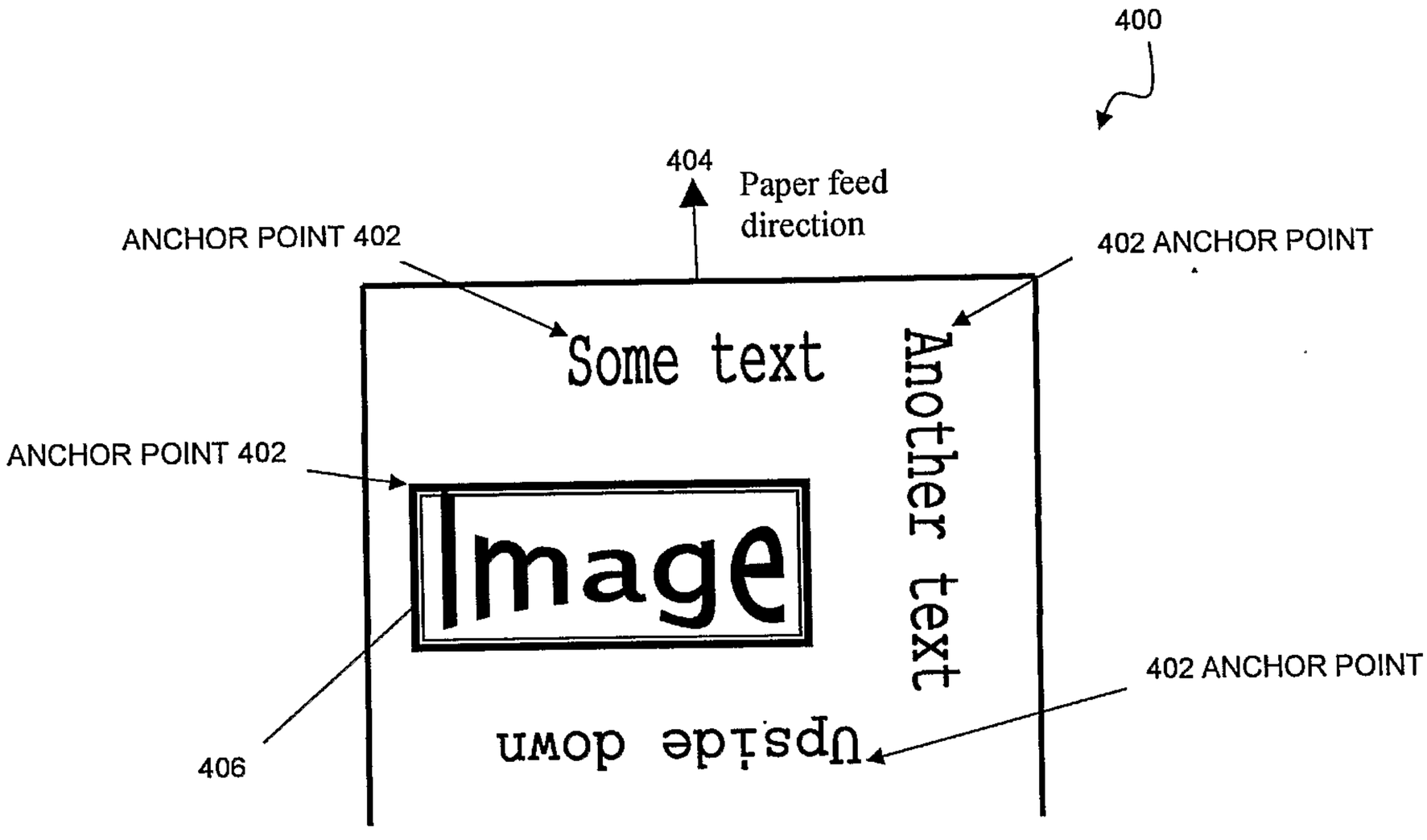

400
404
Paper feed direction
ANCHOR POINT 402
402 ANCHOR POINT
Some text
Another text
ANCHOR POINT 402
Image
402 ANCHOR POINT
406
Upside down

FIG. 5

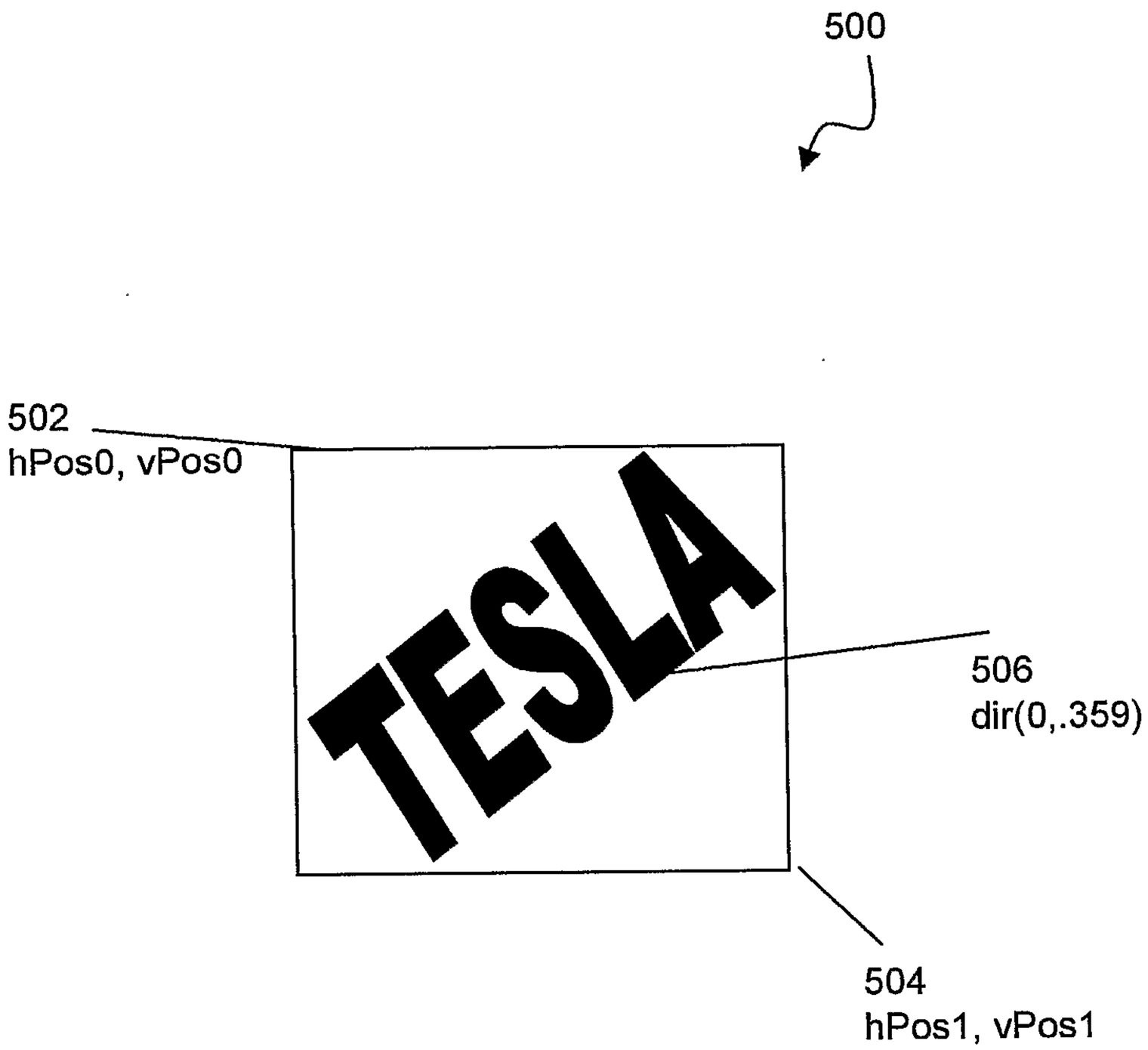


FIG. 6

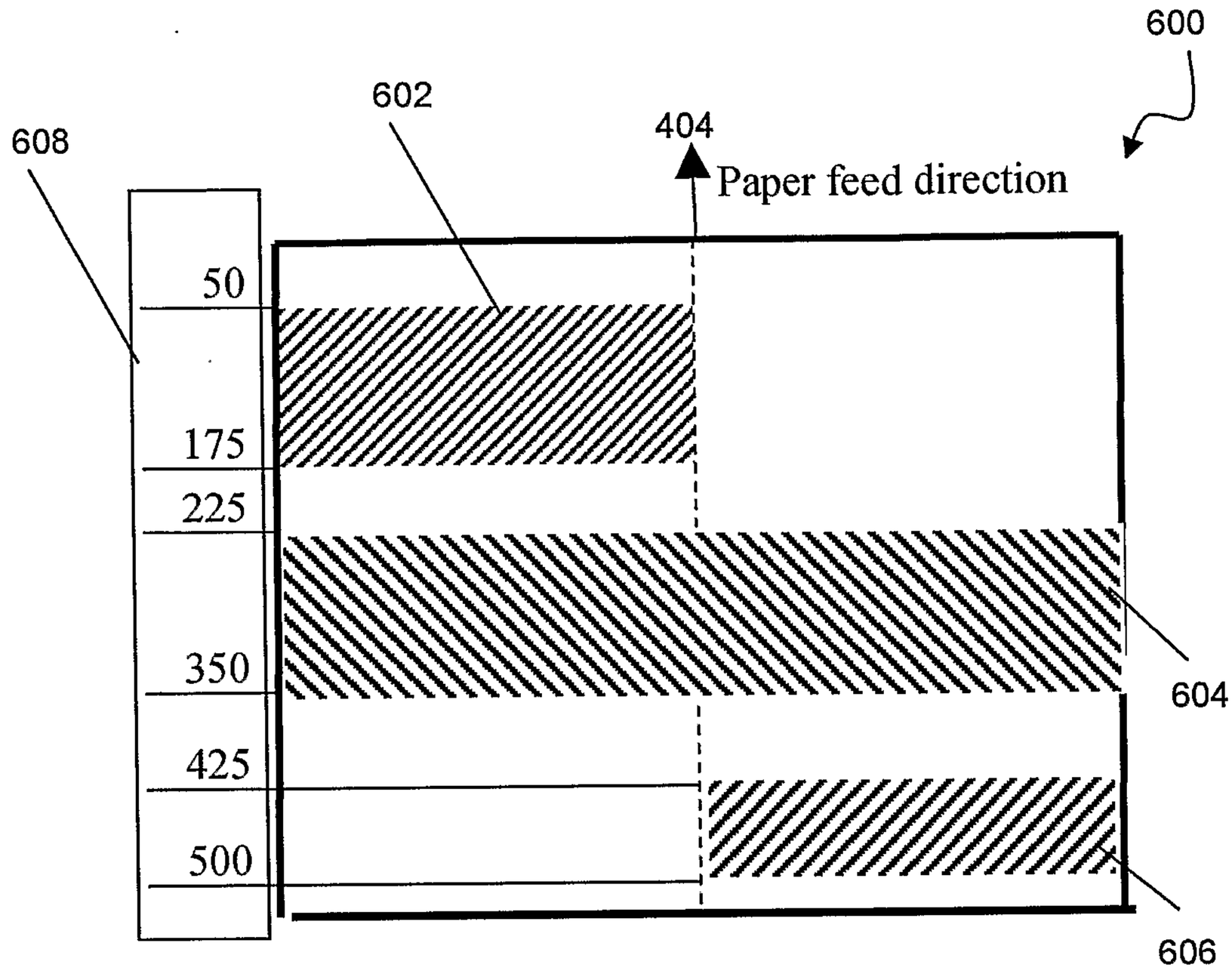

600
602
404
608
Paper feed direction
50
175
225
350
425
500
604
606

FIG. 7

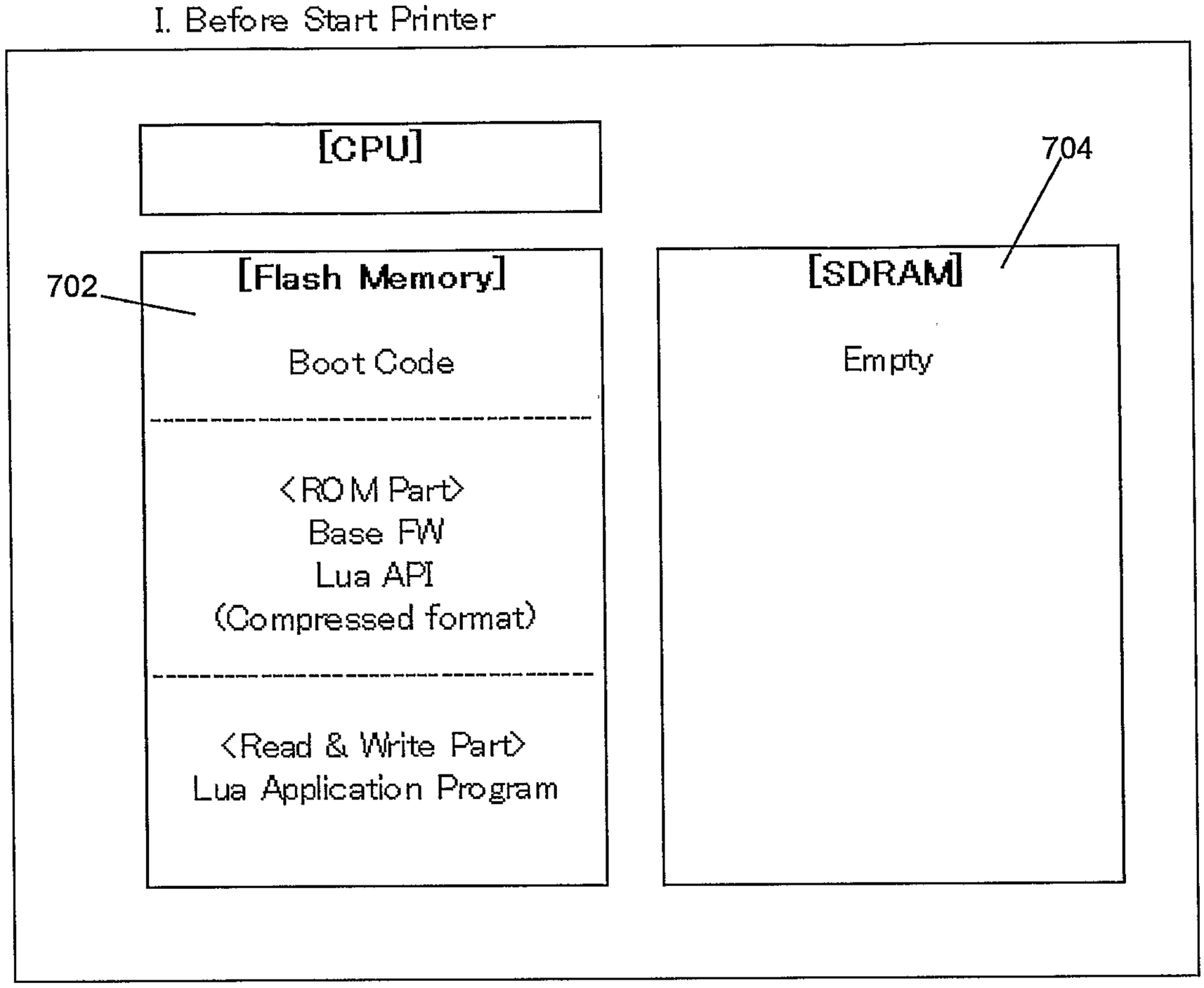

I. Before Start Printer
[CPU]
704
702
[Flash Memory]
Boot Code
<ROM Part>
Base FW
Lua API
(Compressed format)
<Read & Write Part>
Lua Application Program
[SDRAM]
Empty

FIG. 8

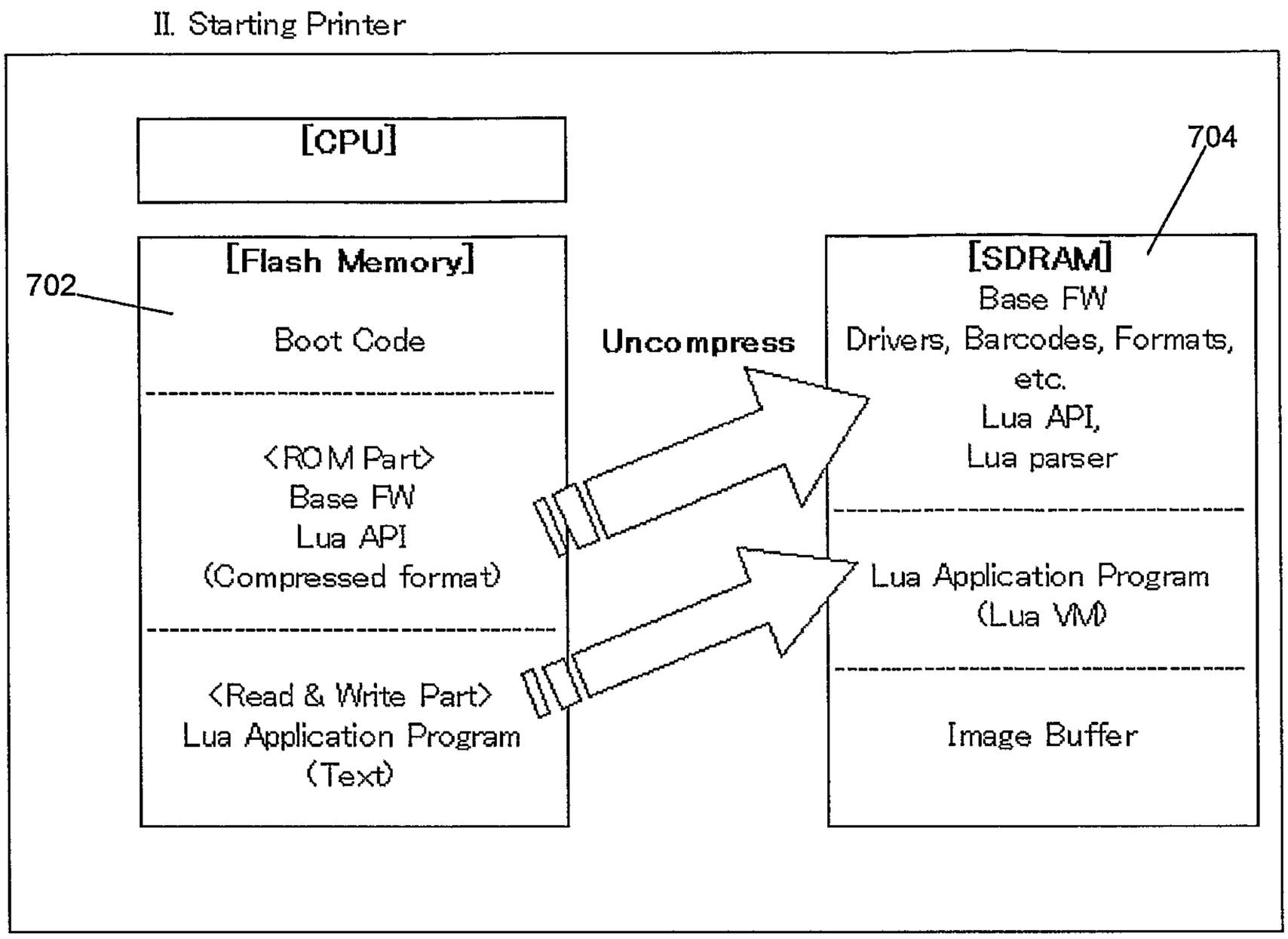

II. Starting Printer
[CPU]
704
[Flash Memory]
702
Boot Code
<ROM Part>
Base FW
Lua API
(Compressed format)
<Read & Write Part>
Lua Application Program
(Text)
Uncompress
[SDRAM]
Base FW
Drivers, Barcodes, Formats, etc.
Lua API,
Lua parser
Lua Application Program
(Lua VM)
Image Buffer

FIG. 9

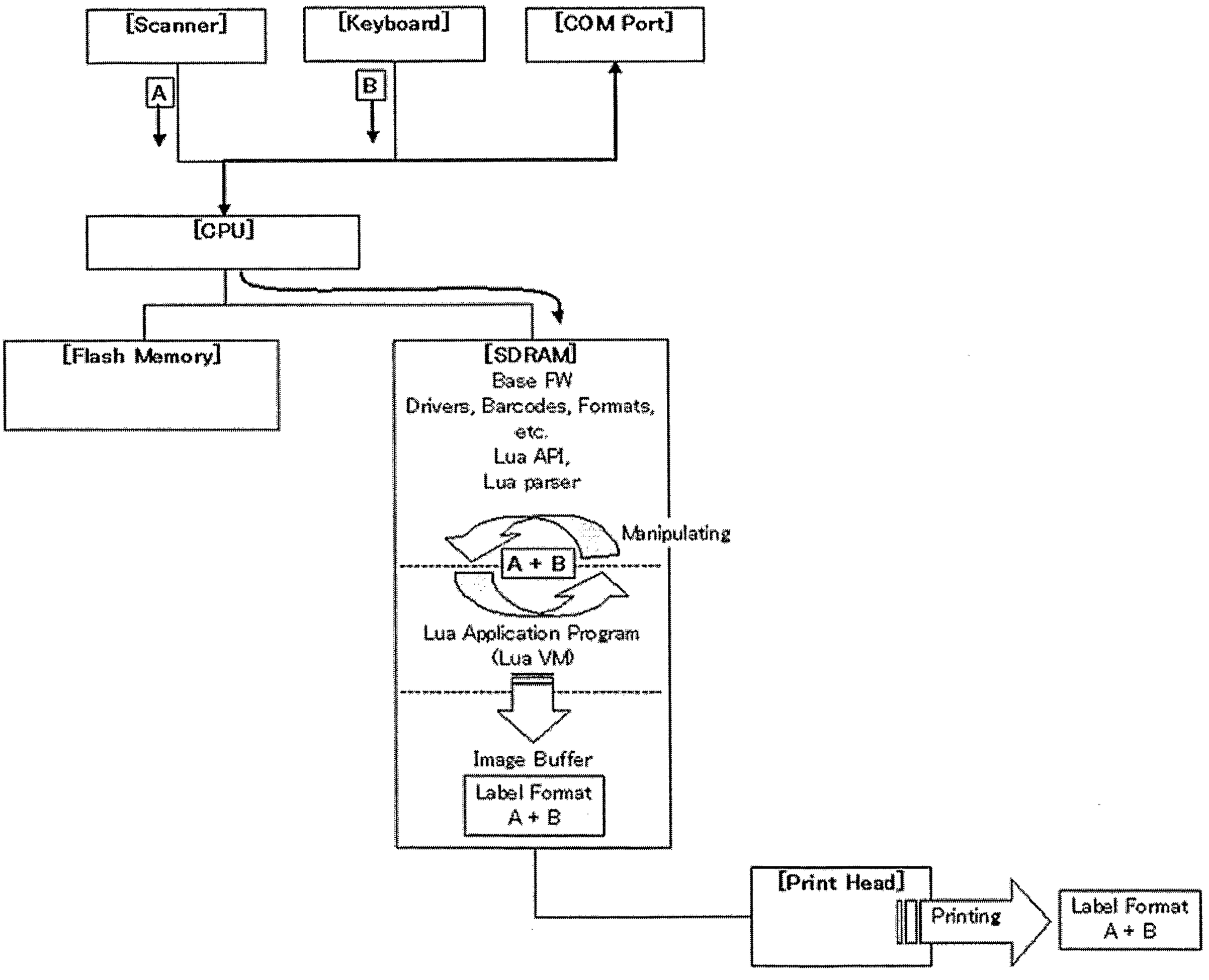

III. Printing Operation
[Scanner]
[Keyboard]
[COM Port]
A
B
[CPU]
[Flash Memory]
[SDRAM]
Base FW
Drivers, Barcodes, Formats, etc.
Lua API,
Lua parser
Manipulating
A + B
Lua Application Program (Lua VM)
Image Buffer
Label Format A + B
[Print Head]
Printing
Label Format A + B

FIG. 10

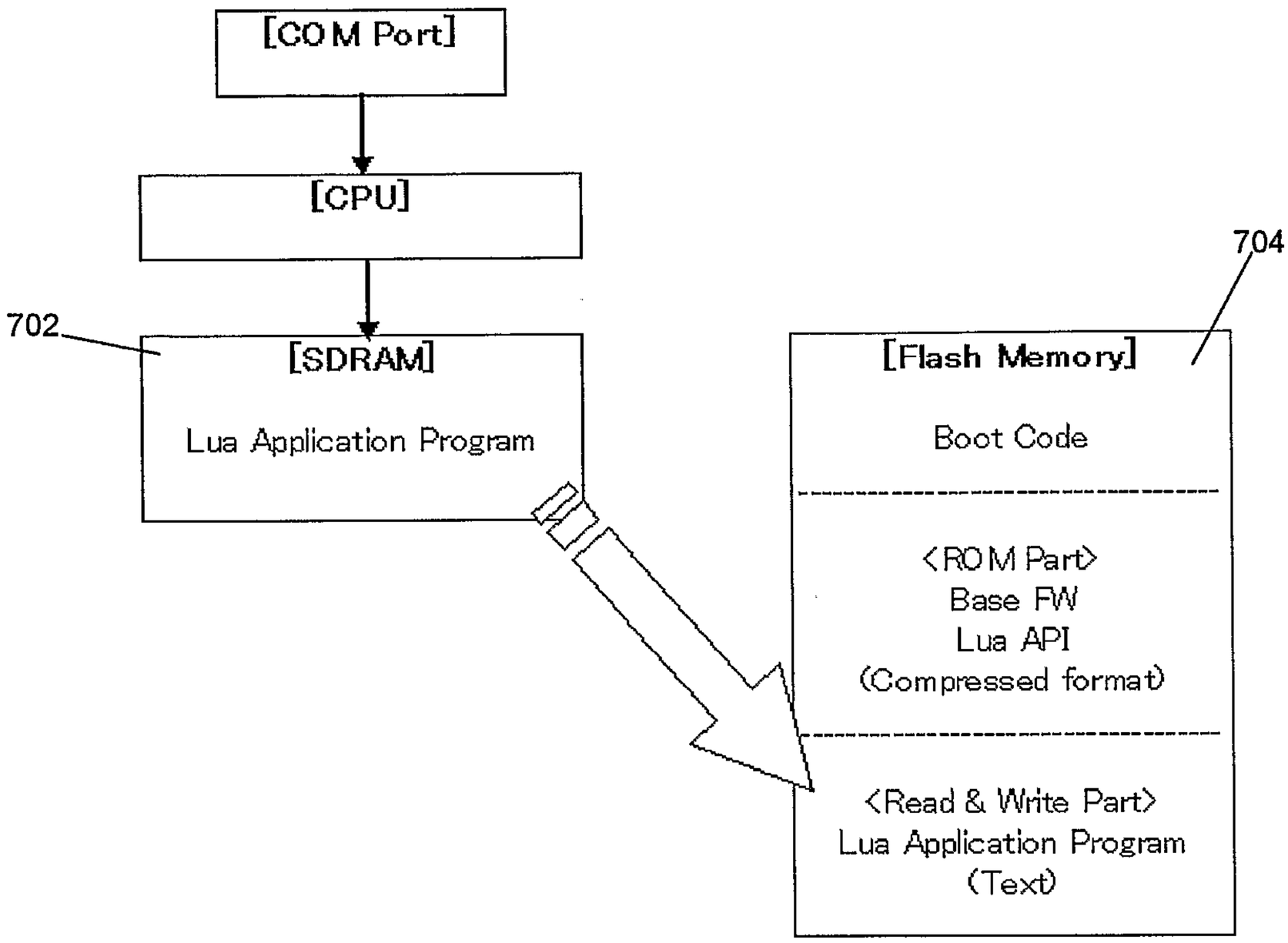
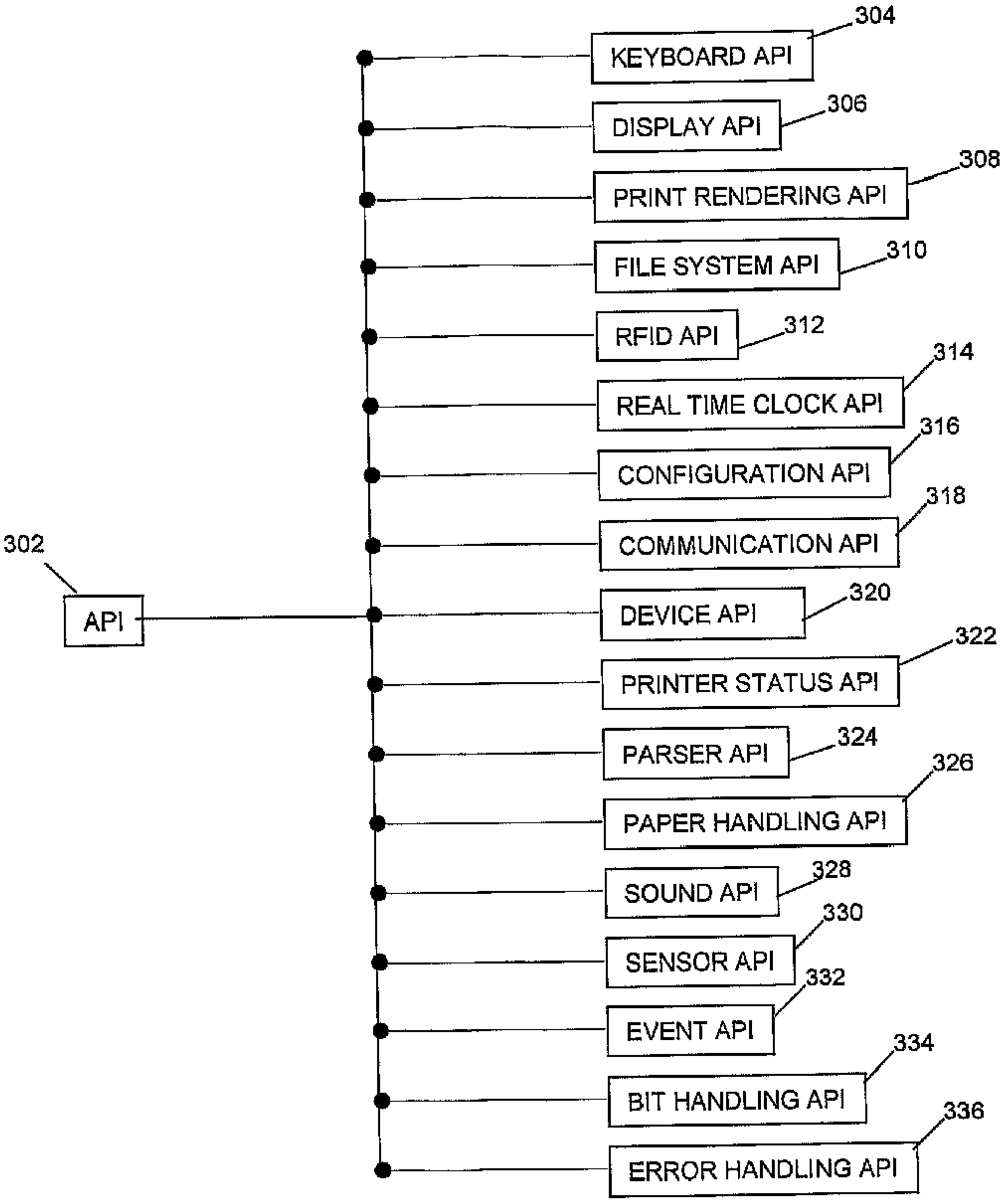

Ⅳ. Download Lua Application Program
[COM Port]
[CPU]
702
[SDRAM]
Lua Application Program
704
[Flash Memory]
Boot Code
<ROM Part>
Base FW
Lua API
(Compressed format)
<Read & Write Part>
Lua Application Program
(Text)

FIG. 3


302
API
304
KEYBOARD API
306
DISPLAY API
308
PRINT RENDERING API
310
FILE SYSTEM API
312
RFID API
314
REAL TIME CLOCK API
316
CONFIGURATION API
318
COMMUNICATION API
320
DEVICE API
322
PRINTER STATUS API
324
PARSER API
326
PAPER HANDLING API
328
SOUND API
330
SENSOR API
332
EVENT API
334
BIT HANDLING API
336
ERROR HANDLING API