



(51) International Patent Classification:
G06F 9/445 (2006.01)

(21) International Application Number:
PCT/IB2010/001178

(22) International Filing Date:
19 May 2010 (19.05.2010)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0911236.8 29 June 2009 (29.06.2009) GB

(71) Applicant (for all designated States except US): **NOKIA CORPORATION** [FI/FI]; Keilalahdentie 4, FIN-02150 Espoo (FI).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **REYNOLDS, Matthew** [GB/GB]; 2-6 Boundary Row, Southwark, London SE1 8HP (GB). **VARGHESE, Arun** [IN/IN]; Jupited Block, Prestige Technology Park, Sarjapur-Marathalli Outer Ring Road, 560103 Bangalore (IN).

(74) Common Representative: **NOKIA CORPORATION**; 6021 Connection Drive MS 2-5-520, Irving, TX 75039 (US).

(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, CL, 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, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, 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, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

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

(54) Title: COMPUTING DEVICE AND METHOD TO REDUCE THE TIME NEEDED FOR THE DEVICE TO START

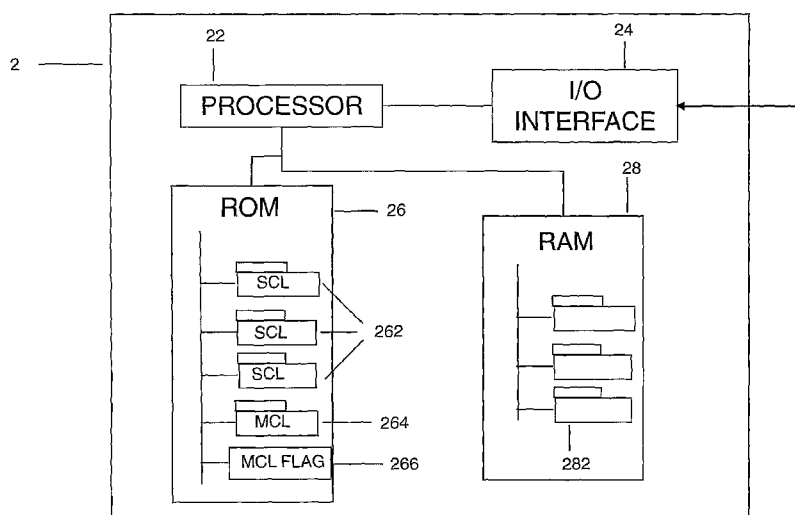


Figure 1

(57) Abstract: One example of the invention operates during the device shut-down procedure, to generate a merged command list containing the components from the various individual static command lists. The generated merged command list is then stored in non-volatile memory, and subsequently accessed on the next device start-up to control the launching of required software components. In this way, only one command list, being the merged command list, need be processed on device start-up, and hence device start-up time is reduced.

Computing device and method to reduce the time needed for the device to start

Technical Field

Examples of the present invention relate to a computing device and associated method.

5

Background to Examples of the Invention

When a computing device starts up it will usually need to launch various software components or run various commands. Several lists of such components or commands, referred to herein as Static Command Lists (SCLs), may be stored. On start-up these lists are processed individually, and the necessary software components launched or commands run. Processing several such lists on device start-up increases the time needed for the device to start.

10

Summary of Examples of the Invention

15

One example of the invention operates during the device shut-down procedure, to generate a merged command list containing the required operations from the various individual static command lists. The generated merged command list is then stored in non-volatile memory, and subsequently accessed on the next device start-up to control the start-up operations performed. In this way, only one command list, being the merged command list, need be processed on device start-up, and hence device start-up time is reduced.

20

In one example of the invention there is provided a method, comprising: generating a merged command list defining device start-up operations to be performed on the start-up of a computing device, the generating including merging two or more existing command lists, each defining one or more start-up operations; storing the generated merged command list in non-volatile storage in the computing device; and then on device start-up, using the stored merged command list to define the device start-up operations to be performed.

25

30

In one example of the invention the generating of the merged command list is performed during a device shut-down procedure. In another example of the invention the generating of the merged command list is performed when an update of software

stored on the device occurs. In a further example of the invention the generating of the merged command list is performed on command. In one example the command may be received from a user. In another example the command may be received from a network to which the device is logically connected.

5

In one example an indicator is stored that indicates whether the stored merged command list is valid. In this example the indicator is examined on device start-up before the stored merged command list is used to define the device start-up operations to be performed, the stored merged command list then being used if the indicator indicates that it is valid. Moreover, in this example, the indicator is re-set when a new merged command list is generated.

10

In a further related example then in the event that on start-up the indicator indicates that the stored merged command list is not valid, a new merged command list is generated from the two or more existing command lists, and the new merged command list is used to define the start-up operations to be performed.

15

Another example of the invention provides a computer program or suite of computer programs so arranged such that when executed by a computing device it/they cause the computer to operate so as to generate a merged command list defining device start-up operations to be performed on the start-up of a computing device, the generating including merging two or more existing command lists, each defining one or more start-up operations; storing the generated merged command list in non-volatile storage in the computing device; and then on device start-up, using the stored merged command list to define the device start-up operations to be performed.

20

25

A further example of the invention provides a computer readable storage medium storing the computer program or at least one of the suite of computer programs as described in the example above.

30

Another example of the invention provides a computing device, comprising: non-volatile data storage; and a processor; the non-volatile data storage and the processor being configured in use to cause the computing device to: i) merge two or more existing command lists, each defining one or more start-up operations so as to generate a merged

command list defining device start-up operations to be performed on the start-up of a computing device; ii) store the generated merged command list in the non-volatile storage; and iii) on device start-up, use the stored merged command list to define the device start-up operations to be performed.

5

In one example the merged command list is generated by the device during a device shut-down procedure. In another example the merged command list is generated by the device when an update of software stored on the device occurs. In a further example the merged command list is generated by the device on command.

10

A further example of the invention provides a computing device having means for generating a merged command list defining starting operations to be performed when the device is started-up from two or more existing command lists; means for storing the merged command list such that it persists after device shutdown; and start-up control means that uses the merged command list to define the start-up operations that are performed when the device is started up.

15

Brief Description of the Drawings

20 Further features and advantages of examples of the invention will become apparent from the following description of specific embodiments of the invention, presented by way of example only, and by reference to the accompanying drawings, wherein like reference numerals refer to like parts, and wherein: -

25 Figure 1 is a block diagram of a computing device according to one example of the invention;

Figure 2 is a flow diagram illustrating the operation of a computing device according to an example of the invention;

30 Figure 3 is a flow diagram illustrating the operation of a computing device according to an example of the invention;

Figure 4 is a diagram illustrating the merging of static command lists in an example of the invention;

Figure 5 is a block diagram of a computing device according to a second example of the invention;

Figure 6 is a diagram illustrating a static command list in an example of the invention;

Figure 7 is a flow diagram illustrating the operation of a computing device according to an example of the invention;

Figure 8 is a flow diagram illustrating the operation of a computing device according to

5 another example of the invention;

Figure 9 is a diagram illustrating the contents of a static command list in an example of the invention; and

Figure 10 is a flow diagram illustrating the operation of a computing device according to another example of the invention.

10

Description of Specific Embodiments

Various examples of the invention will now be described with respect to the accompanying figures.

15

Figure 1 is a block diagram of a typical computing device according to a first example of the invention. The computing device 2 comprises a processor 22, and has two types of memory, being read only memory 26, and random access memory 28. The read only memory 26 would typically include the device operating system, which is loaded and run by the processor 22 on device start-up. Other various software components may also be stored in ROM 26. In this example ROM 26 may be Flash memory, and hence, whilst referred to as “read-only” in fact can also be written to. However, the parts of the memory where the operating system and other essential executables and data are stored are preferably read-only.

25

In this example the term “computing device” should be expansively construed to include any device in which a boot or start-up routine is undertaken, and particularly although not exclusively one in which various commands are executed or software components loaded. In this respect, the computing device may be any of, for example, a desktop computer, laptop computer, netbook, mobile telephone, smartphone, digital camera, digital media player, positioning system, audio-visual equipment, set-top box, signal processing equipment, or any device with a start-up routine.

30

During operation of the computing device 2, the processor 22 loads various software components and data into RAM 28 as needed. Usually, software components that are presently being run are stored in RAM, although they may be kept in ROM, if the ROM is execute in place (XIP) ROM. Executables and data are provided to the processor 22 from the ROM or RAM as required, depending on the task set for the computing device by the user. The computing device 2 also includes an input and output interface 24, which the processor uses to receive user commands, and to output data in various forms, such as, for example, as audio data, as video data, or as data of other types, such as documents, spreadsheets, etc. etc. In this respect, the IO interface 24 represents in this example the various outputs and inputs available to and from the processor 22. In a practical implementation, there would typically be several such input and output interfaces of different types.

Pertinent to the present example, the ROM 26 stores several static command lists 262.

Each static command list contains a list of software components that are to be launched when the computing device 2 is started up, or other commands that need to be run on start-up. In this description, the contents of an SCL, whether a command to be run or a software component to be launched will collectively be referred to as the “components” of the SCL. Usually, a start-up static command list is fixed at the ROM build time when the device is first manufactured. At device start-up, after loading the initial elements of the operating system, the processor 22 will process each static command list, and launch all the indicated software components or run the indicated commands. In the present example, multiple SCLs are stored in the ROM 26, but in other examples multiple start-up SCLs can be located on any internal fixed drive. There are no restrictions regarding the total number of start-up SCLs. On device start-up, the processor 22 processes each SCL, and launches or runs the components indicated in each SCL. The components may have a priority assigned indicating in which order the components are to be launched or run. Alternatively, the priority of a component could be indicated by its position in the SCL list.

One problem with processing multiple SCLs on device start-up is that the device start-up time is impacted. That is, it takes longer to process multiple SCLs which may be in different places than to process a single SCL. To get around this problem, and improve device start-up time, in this example the individual SCLs are merged to produce a

merged command list (MCL) which contains all of the components from the multiple start-up SCLs. There are no restrictions regarding the total number of start-up SCLs that can be used to create a merged command list. In this example, to reduce the impact of creating a merged command list at device start-up, the merged command list is created (if needed) by the processor 22 when the device shuts down. The generated merged command list is then stored in non-volatile memory, and can be accessed by the processor 22 at the next device start-up. The advantage of such operation is that device start-up times can be improved, because the processor 22 need only process components from the merged command list, and does not need to process each individual static command list.

In addition, when a static command list is updated while the device is switched on, such as by a "Firm-Ware-Over-The-Air" (FOTA) update, then in one example the merged command list is recreated on the next device shut-down, such that on device start-up, the MCL that is used to launch necessary software components is always up to date. In another example of the invention an MCL may be generated whenever a software update occurs.

Further details of the operation of the present example will become apparent from Figures 2 and 3, described next.

More particularly, Figure 2 illustrates the operation of the computing device 2 in the present example to generate a merged command list. Firstly, at step 2.2 the processor 22 determines that the device is to shut-down. Here, shut-down is intended to be a "graceful" shut-down, i.e. a controlled shut-down either commanded by the user, or by the device itself, for example because battery power is becoming low. In this respect, Figure 2 does not apply when an accidental incident shut-down takes place, such as is the case when power is suddenly removed from the computing device 2, whether accidentally or intentionally. For example, where the computing device 2 is a portable computing device, if the battery falls out then power will be instantly removed, and the device will immediately shut-down.

Therefore, when a graceful shut-down has been commanded at step 2.2, in order to produce the merged command list the following steps are undertaken. It will be

understood that other steps are also undertaken during a graceful device shut-down, but that herein we describe only those steps that pertain to the present example of the invention.

5 At step 2.4 an evaluation is undertaken as to whether a merged command list already exists. For example, a merged command list may have been generated by a previous device shut-down procedure. If no such merged command list exists, which may be the case where the merged command list has been deleted, or where this is the first graceful shut-down of the device, then it is necessary to create a merged command list, and
10 processing proceeds to step 2.8. Alternatively, where a merged command list already exists, then processing proceeds to step 2.6, wherein a further evaluation is undertaken to determine whether any of the static command lists present in the device have been updated since the last device start-up.

15 The static command lists may be updated in several ways. For example, the static command list may be updated by a "Firmware-Over-The-Air" update, wherein the device firmware is updated automatically by receiving updated firmware over a communications link. Alternatively, the static command list may have been updated by a user loading new software onto the computing device 2. In a further example, the
20 static command list may have been updated via an automatic update of the software on the device. For example, the operating system may be automatically updated from time to time, by the computing device automatically contacting an update server to see if updates exist. Alternatively, an update server may push updated software to the device, for example when the device is connected to a communications network. There are thus
25 several mechanisms by which the static command list may be altered during the device operation.

If at step 2.6 it is determined that no update has been performed of any of the static command lists since the last device start-up, then there is no need to create another
30 merged command list. In this case, processing proceeds to step 2.12, wherein a "MCL_valid" flag 266, stored with the already existing merged command list 264 is set, to indicate that the merged command list is valid. Thereafter, the device continues shutting down at step 2.14. In such a case the existing MCL will be used at the next

device start-up, as the MCL_valid flag indicates that the existing MCL is already up-to-date.

Returning to step 2.6, however, if it is determined that any of the static command lists in the device have been updated since the last device start-up, then it will be necessary to recreate the merged command list that already exists, and in this case processing proceeds to step 2.8.

At step 2.8 a new merged command list is generated, by merging together the various commands in the individual static command lists stored on the device. The static command lists may be stored on any fixed drive in the device, and hence at step 2.8 the individual static command lists are accessed in turn, and the components therein copied into a single merged command list file. Various further examples of the different ways in which the static command list commands may be merged together to create the merged command list will be described later. Once the merged command list has been generated, it is then stored at step 2.10. In this example, the merged command list is stored in the ROM 26. Processing then proceeds to step 2.12, wherein the "MCL_valid" flag is set, to indicate that the newly generated merged command list is valid, and up to date. The MCL_valid flag is used by the device on device start-up, to check whether the MCL is up to date.

In this respect, as mentioned previously in this example ROM 26 is typically flash memory, such as NAND Flash Memory and has a part of the flash memory that is protected, and that cannot be written over. This part will typically store the operating system code. However, another part of the flash memory is writable, and it is in this part that the SCLs 262, and any generated MCL and MCL_valid flag 266 is stored.

Figure 3 illustrates how the generated MCL is used at device start-up, in the present example. At step 3.2 the device begins its start-up procedure. This would typically include loading elements of the operating system from the read-only part of ROM 26, so as to enable the device to begin operating. As part of the device start-up procedure, the following steps are undertaken. It will be understood that many other steps will also be undertaken during the device start-up procedure, but which are not pertinent to the present example of the invention.

At step 3.4 the processor 22 examines whether the MCL_valid flag has been set. This is to determine whether a valid MCL, that reflects the contents of the present SCLs stored on the device has been generated. If such an MCL has been generated, then the device
5 can access the MCL, and process the MCL to launch various software components, rather than process the individual SCLs. That is, if, at step 3.4, it is determined that the MCL_valid flag has been correctly set, then the processor 22 knows that it can trust the existing MCL to be up to date. In this case processing proceeds to step 3.8, wherein the MCL_valid flag is reset to show that it has been read, but that the MCL is still up to
10 date, and then processing proceeds to step 3.10, wherein software components and commands are executed according to the existing MCL list. As noted previously, processing the stored MCL is quicker than processing various separate SCLs, which may be stored on different drives.

15 Returning to step 3.4, if the MCL_valid flag has not been correctly set, then processing proceeds to step 3.6. In this case, as there is no valid MCL already stored, the device 2 proceeds to generate an MCL during the start-up procedure. Here, during the device start-up procedure the various SCLs are accessed, and the commands therein merged together to produce a single merged command list. The merging of the SCLs at step 3.6
20 is the same as described previously with respect to step 2.8, and further examples of the SCL merging procedures that may be used will be given later. Once an MCL has been generated, then the MCL_valid flag is set at step 3.8, and then components are run according to the generated MCL list at step 3.10. In this case, however, the device start-up time is considerably longer, as there is the extra step of having to generate the MCL
25 from the individual SCLs.

Figure 4 illustrates an example of how multiple SCLs may be merged together into a single MCL. In this case, there are four SCLs 262, which are merged together into a single example MCL 264. As one example of how the SCLs may be merged together,
30 in this example the SCLs are each accessed in turn, and the first command or software component in each SCL list is copied into the MCL list. Next, the second component from each SCL is taken and copied into the MCL. Then, the third component from each SCL is taken and copied into the MCL, and so on, until all SCL components have been copied into the MCL. In this example, therefore, it can be seen that the first three

commands of the SCL 262 shown appear at the first, fifth, and ninth positions in the MCL list, with commands or components from the other SCLs interspersed between. It should be noted that this is but one example of how the SCL commands can be merged together, and further examples will be given later.

5

In this example the procedures of Figures 2 and 3 are implemented as software modules within the operating system. That is, separate software components are implemented within the operating system that when executed result in the operations of Figures 2 and 3. As noted, the operating system is typically stored in the ROM 26, as shown as nodule 10 268 in Figure 2, and the software components implementing Figures 2 and 3 are contained therein.

In this first example, therefore, the computing device 2 generates a merged command list from individual static command lists, and then processes the merged command list 15 on device start-up, so as to launch software components as required. As noted previously, the processing of the merged command list rather than the separate static command lists results in a quicker device start-up.

Another example of the invention will now be described with respect to Figure 5. In this 20 example the computing device is a smart phone 10 comprising hardware to perform the telephony functions, together with an application processor in corresponding support hardware to enable the phone to have other functions which are desired by a smart phone, such as messaging, calendar, word processing functions, and the like. In Figure 5 the telephony hardware is represented by the RF processor 102 which provides an RF 25 signal to antenna 126 for the transmission of telephony signals, and the receipt therefrom. Additionally provided is baseband processor 104, which provides signals to and receives signals from the RF processor 102. The baseband processor 104 also interacts with a subscriber identity module 106, as is well known in the art.

30 Also provided in this example is a display 116, and a keypad 118. These are controlled by an application processor 108, which is often a separate integrated circuit from the baseband processor 104 and RF processor 102, although in the future it is anticipated that single chip solutions will become available. A power and audio controller 120 is provided to supply power from a battery (not shown) to the telephony subsystem, the

application processor, and the other hardware. Additionally, the power and audio controller 120 also controls input from a microphone 122, and audio output via a speaker 124.

5 In order for the application processor 108 to operate, various different types of memory are often provided. Firstly, the application processor 108 may be provided with some random access memory (RAM) 112, into which data and program code can be written and read from at will. Code placed anywhere in RAM can be executed by the application processor 108 from the RAM.

10

Additionally provided is separate user memory 110, which is used to store user data, such as user application programs (typically higher layer application programs which determine the functionality of the device), as well as user data files, and the like.

15 In order for the application processor 108 to operate, an operating system is provided, which is started as soon as the smart phone system 10 is first switched on. The operating system code is commonly stored in a read only memory, and in modern devices the read only memory is often NAND flash ROM 114. The ROM will store the necessary operating system components in order for the device 10 to operate, but other
20 software programs might also be stored, such as application programs, and the like, and in particular those application programs which are mandatory to the device, such as, in the case of a smart phone, communications applications and the like. In addition, as shown in Figure 5, the smart phone 10 in the present embodiment also stores various Static Command Lists (SCLs) 1142 in the NAND flash ROM 114, as well as a generated
25 Merged Command List (MCL) 1144, and the MCL_valid flag 1146. An SCL 1102 is also stored in user memory 110. The generation of the MCL 1144 from the various SCLs 1142 and 1102 in the present example embodiment will be described further below.

30 In the present example the smart phone 10 generally operates in the same manner as the computing device 2 of the previously described example. That is, the smart phone 10 can operate in accordance with Figures 2 and 3 as previously described, in order to generate and use merged command lists, from multiple static command lists. However, in another example, the smart phone 10 may not operate generally in accordance with

Figure 3 on device start-up, and in particular need not perform step 3.6 to generate a merged command list in the case that the MCL_valid flag is not set at step 3.4. Instead, where the MCL_valid flag is not set, the smart phone 10 may simply start-up by accessing the individual static command lists separately, and processing each SCL separately. Then, when the smart phone 10 shuts down, and the procedure of Figure 2 is performed, an MCL will be generated at that time, which can then be used during a subsequent start-up procedure.

Generally, therefore, smart phone 10 of the present example operates in accordance with Figure 2, to generate a merged command list when the smart phone 10 is shut-down. The generated merged command list is then stored, and accessed at next device start-up, and used in place of the individual static command lists. In this respect, the operation of smart phone 10 is generally the same as the computing device of the previous example.

Also similarly to the previous example, smartphone 10 has its O/S stored in NAND Flash ROM 114, as shown in Figure 5. The procedure of Figure 2 is, in this example, implemented as a software component that is part of the operating system, stored on the NAND Flash ROM.

However, in the present example, smart phone 10 can perform additional actions in generating a merged command list from the individual static command lists. Figure 6 gives an example of an enhanced static command list, wherein each component or command in the static command list also has an associated condition associated with it. The condition indicates a particular condition which may occur during device start-up, as to whether the individual component or command in the static command list should be launched or run as appropriate. For example, one command or component in a static command list may have the condition “always” which means that it should always be run on each start-up. Other commands or components may have a condition associated with them that they should only be launched or run when the device performs a start-up in response to a particular condition. One such condition is an “alarm” condition. For example, the user of the smart phone 10 may have placed the device into a sleep mode, but set an alarm to wake the device up at a particular time. In this case, during the start-up procedure based on an alarm condition, different components may be launched, or different commands performed, than during a normal device start-up. In the present

example, when such conditions are contained within an SCL, then those conditions are maintained when an SCL component is copied into an MCL during its generation.

Turning now to Figure 7, there will be described in more detail an example SCL merging process, that may be used for example in step 2.8 or step 3.6 of Figures 2 and 3 described previously. It should be understood that this is one example of how the SCL merge process may operate. Other examples will be described later.

Generally, the procedure as set out in Figure 7 acts to access each SCL in turn, and then, for each command or component in the SCL, a check is performed as to whether that command or component has been added to a merged command list that is being generated, and if it has not been so added, then the command or component is added to the list. However, if the command or component already exists in the list in identical form, then it is not added to the MCL list. This therefore prevents unnecessary multiple instances of the same command being performed, or software component being launched.

In more detail, at step 7.2 a FOR processing loop is started to process each SCL in turn. For example, in the smart phone 10, the SCL 1102 and the two SCLs 1142 would be processed in turn. Next, at step 7.4 a further FOR processing loop is started to process each component member of the present SCL that is presently being processed by the outer processing loop.

Within these nested processing loops, firstly, at step 7.6 an evaluation is performed as to whether the MCL that is presently being generated in fact has any members. This evaluation will return false only for the first iteration of the processing loops, as thereafter a command or component will have been added to the MCL that is being generated. If the MCL list is not populated, then at step 7.8 the present SCL command or component member that is presently being processed is added to the MCL list that is being generated. Conversely, if this is not the first iteration of the process, then the MCL list will be populated, and processing will proceed to step 7.10.

At step 7.10 a further nested processing loop is started, to process each member of the MCL that is presently being generated. Then, at step 7.12 an evaluation is undertaken

to determine whether the present SCL member component that is being processed is the same as the present MCL member component that is being processed. If it is not the same, then the present SCL member component is added to the MCL list, at step 7.14. If it is the same, then it means that the present SCL component that is being processed
5 has already been added to the MCL list, and hence does not need to be added again. In this case, processing proceeds to step 7.16. Likewise, from step 7.14 processing also proceeds to step 7.16.

At step 7.16 an evaluation is undertaken as to whether there is a next MCL number to
10 process. If this is the case, then the next MCL member is selected, and processing proceeds back to step 7.12. If it is not the case, then all presently existing MCL member commands or components have been looked at with respect to the SCL member that is presently being processed, and hence processing proceeds to step 7.18. In this case, the
15 next SCL member of the SCL that is presently being processed is selected, where one exists, and processing proceeds back to step 7.6. Conversely, once each member of the SCL that is presently being processed has been looked at, then step 7.18 will allow the procedure to proceed to step 7.20, wherein the next SCL to be processed is selected. Processing then reverts back to step 7.4.

20 Therefore, as described, this procedure looks at each SCL in turn, and copies each SCL member into the MCL, unless the SCL member component has already been included in the MCL.

Within the above example, where an SCL command or component member has
25 conditions attached to it, for example as described previously with respect to Figure 6, then those conditions are also copied into the MCL.

A further example of how SCLs may be merged will next be described with respect to Figure 8. Again, the procedure to be described with respect to Figure 8 can be used in
30 the MCL generation steps in Figures 2 and 3, described previously.

In the previous example, SCL components were copied into the MCL in the order in which they appeared in the SCL, but because each SCL is processed completely one after the other, the components of one SCL which is processed after another SCL will

appear in the MCL list after the components of the SCL that was processed first. As an alternative, in the present example, the first component from each SCL to be merged is included first in the MCL list, and then the second component of each SCL is included after all of the first components. Thus, all the first components of each SCL are
5 positioned higher in the MCL list than second components. Similarly, second components of each SCL are positioned higher in the MCL list than third components, and so on. Thus, where position in an SCL list indicates priority in terms of which command or component should be executed or run first, then this priority can be maintained in the MCL.

10

In order to achieve such an outcome, firstly in the procedure of Figure 8 processing is undertaken to determine the SCL with the highest number of components. This is necessary so that the procedure knows how many iterations need to be performed. Therefore, at step 8.2, a processing loop is started to process each SCL that is to be
15 merged together into an MCL. At step 8.4 the number of components in the present SCL is determined, and is assigned to a variable "count". Next, at steps 8.6 and 8.8 an evaluation is performed to determine whether a variable "highest_count" is less than the variable count, and if it is, then the variable highest_count is set to be equal to the variable count. This procedure is performed for every SCL, and at the end of it the
20 variable highest_count should contain a count of the number of components in the SCL with the most components therein.

Thereafter, at steps 8.12 to 8.24, the MCL is generated by taking each first component in turn from each SCL and copying into the MCL, followed by each second component,
25 followed by each third component, and so on. In particular, at step 8.12 a processing loop is started based on a variable n , and is performed a number of times, incrementing n between 1 and the "highest_count" variable previously mentioned. Within the outer loop, at step 8.14 a second nested processing loop is started to process each SCL for the present variable n . Within this nested loop at step 8.16 an evaluation is performed as to
30 whether the n th component exists in the present SCL, and if it does then the n th component is copied to the MCL list at step 8.18. These two loops then cycle round via the evaluations at step 8.20 and 8.22. Thereafter, each SCL component will have been copied into the MCL component list. However, the MCL component list may contain duplicate entries from the various SCLs, and hence at step 8.24 these duplicate entries

are removed from the MCL. Typically, the duplicate entries lower in the list are removed, such that the highest place entry is retained.

5 Using the example merged process of Figure 8, MCLs such as described previously with respect to Figure 4 may be obtained.

In addition, in another example step 8.24 need not be performed. In this case, the MCL may contain duplicate entries, but depending on the specific application this may be acceptable.

10

Another example of how the merge process may be undertaken will now be described with respect to Figures 9 and 10.

15 In Figure 9, the SCL 1142 stores priority information for each command or component in its list. The priority information indicates the order in which the commands or components should be run. In this case, the commands or components may be in any order in the static command list, with the attendant priority information then indicating to the device the order in which commands should be run on start-up.

20 In another example merged process, therefore, this priority information may be used to determine the order in which components are copied to the MCL. Figure 10 illustrates an example procedure which makes use of this priority information to merge SCLs together. Again, as with the previous examples of merge procedures, the merge procedure of Figure 10 may be used as the merge step in steps 2.8 or 3.6 of Figures 2
25 and 3, described previously. The procedure of Figure 10 generally acts to look at each SCL, and copy higher priority components of each SCL into the MCL before lower priority components. That is, each component of priority 1 will be copied from each SCL before components of priority 2 from each SCL, and so on.

30 The merge procedure operates in this example as follows. Firstly, at step 10.2 a processing loop is started to process each priority type. That is, components of priority 1 are processed first, followed by components of priority 2, up to the highest priority number in the SCLs. In this respect, reference to "highest_priority" is potentially misleading, as in fact a component with a priority of value "highest_priority" will in

fact be the lowest priority component. In this case, the “highest_priority” variable is a numerical value.

Next, a further processing loop is started at step 10.4, to process each SCL in turn.

5 Thereafter, at step 10.6 the components of present priority n are copied into the MCL. These processing loops continue via steps 10.8 and 10.12 such that each SCL is looked at, and each priority is looked at. The result is an MCL wherein the SCL components of the highest priority are copied first, followed by SCL components of the next highest priority, and so on. At step 10.14 the MCL is examined further and duplicate
10 components are removed therefrom, if necessary. However, in other examples, this step may not be performed.

There are thus numerous ways in which the static command lists can be merged together to perform a merged command list. Other examples will be apparent to the intended
15 reader, and examples of the invention are not limited to any of the specific examples set out herein.

In addition, in the examples described above, the generation of the merged command list is performed preferably at device shut-down, or alternatively may be performed at
20 device start-up, if no valid MCL is available. Typically, MCL generation at start-up may be performed on the very first device start-up. In other examples, MCL generation can be undertaken at any time during the operation of the device. For example, it need not be undertaken at device shut-down, but may be undertaken during normal device operation. In one example, MCL generation may be scheduled to be performed
25 regularly or periodically during normal device operation. Alternatively, in another example MCL generation may be performed when the device is commanded to do so by a user. In a further example MCL generation may be performed when the device is commanded to by another device, or by a network. For example, where the device is a mobile telephone, the device may receive a command to generate an MCL from the
30 mobile network.

In another example, MCL generation may be performed whenever new software is installed on the device, or when software already on the device is updated. In a further example MCL generation may be performed during idle periods, when the device is

switched on and operating, but is not being used by a user for any particular task. For example, when the processor usage has been below a particular level, say 20%, for a period of time such as an hour, then MCL generation may be performed during such periods of low use.

5

In the preferred example, the generated MCL is stored in non-volatile memory, such that it persists while the device is shut-down, and is then available for use on the next device start-up.

10 In a further example, because the SCLs are no longer used during the device start-up, they may be written in human-readable XML or some other higher level language, or with human readable comments. This is because the time taken to parse human readable SCLs on start-up is not relevant, where a suitable MCL has been generated. Writing the SCLs in human-readable language will make it easier for device development to occur.

15

In addition, in another example where the SCLs are written in human-readable language, then during the merge process for such SCL components a translation from the SCL human readable component into binary equivalent is made, and the binary equivalent stored in the MCL. This allows the MCL to be read more quickly on device
20 start-up than would be the case if the human readable XML or the like was copied directly into the MCL.

The example merge processes of Figure 7, 8, and 10 described above are, in the described examples, typically implemented as software components that form part of
25 the operating system. That is, when used in the smartphone 10 of the example above described with respect to Figure 5, the merge processes form part of the operating system stored on NAND Flash ROM 114.

Various further modifications, whether by way of addition, deletion, or substitution will
30 be apparent to the intended reader, being a person skilled in the art, to provide further examples, any and all of which are intended to fall within the appended claims.

Claims

1. A method comprising:
merging two or more existing command lists, each command list defining one or
5 more start-up operations, to generate a merged command list defining a computing
device start-up operations to be performed on start-up of a computing device;
storing the merged command list in the computing device; and
on computing device start-up, defining computing device start-up operations to
be performed, based at least in part on the stored merged command list.
10
2. A method according to claim 1, wherein the merged command list is generated
during the computing device shut-down procedure.
3. A method according to claim 1, wherein the merged command list is generated if
15 software stored on the computing device is updated.
4. A method according to claim 1, wherein the merged command list is generated
based at least in part on a command.
- 20 5. A method according to claim 1 further comprising:
storing an indicator indicating that the stored merged command list is valid; and
examining the indicator, on the computing device start-up, before the stored
merged command list is used to define the computing device start-up operations to be
performed, and wherein the stored merged command list is used if the stored merged
25 command list is valid.
6. A method according to claim 5, wherein the indicator is re-set if a new merged
command list is generated.
- 30 7. A method according to claims 5 further comprising generating a new merged
command list from the two or more existing command lists on start-up, if the indicator
indicates that the stored merged command list is invalid, and the new merged command
list is used to define the start-up operations to be performed.

8. A computer program arranged such that when executed by a computing device to cause the computing device to operate in accordance with the method of any of the preceding claims.

5 9. A computer readable storage medium storing the computer program according to claim 8.

10. A computing device comprising:

at least one memory; and

10 at least one processor;

the at least one memory and the at least one processor are configured in use to cause the computing device to:

merge two or more existing command lists, each defining one or more start-up operations, to generate a merged command list defining a computing device start-up operations to be performed on the start-up of the computing device;

15

store the generated merged command list; and

on computing device start-up, define computing device start-up operations to be performed, based at least in part on the stored merged command list.

20 11. A computing device according to claim 10, wherein the merged command list is generated by the computing device during the computing device shut-down procedure.

12. A computing device according to claim 10, wherein the merged command list is generated by the computing device if software stored on the computing device is updated.

25

13. A computing device according to claim 10, wherein the merged command list is generated based at least in part on a command.

30

14. A computing device according to any of claims 10 further comprising a stored indicator indicating that the stored merged command list is valid, wherein on the computing device start-up the indicator is examined before the stored merged command list is used to define the computing device start-up operations to be performed, and
5 wherein the stored merged command list is used if the stored merged command list is valid.

15. A computing device according to claim 14, wherein the indicator is re-set if a new merged command list is generated.

10

16. A computing device according to claims 13, wherein the at least one memory and the at least one processor are further configured in use to cause the computing device to generate a new merged command list from the two or more existing command lists on start-up, if the indicator indicates that the stored merged command list is invalid,
15 and the new merged command list is used to define the start-up operations to be performed.

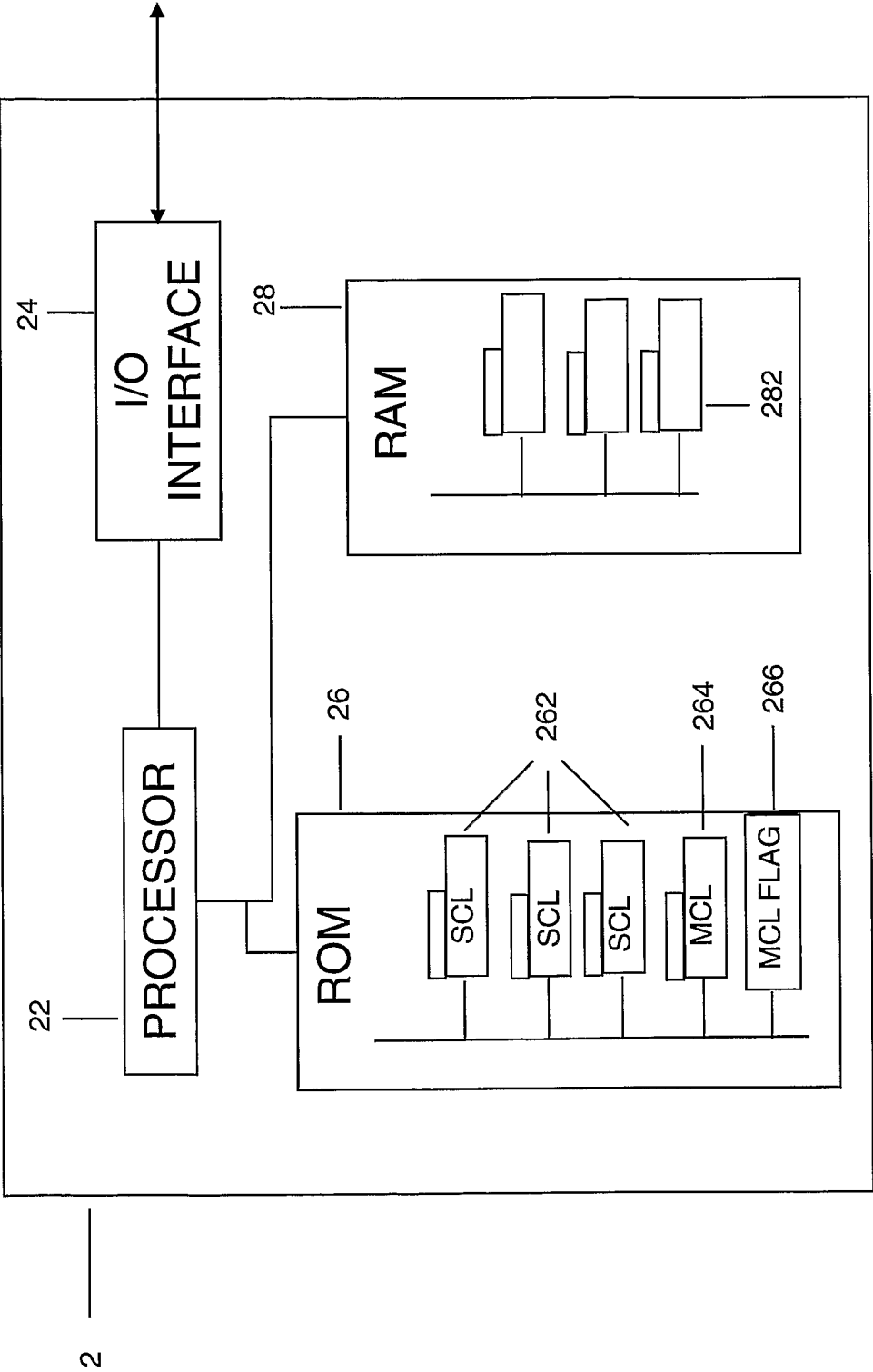


Figure 1

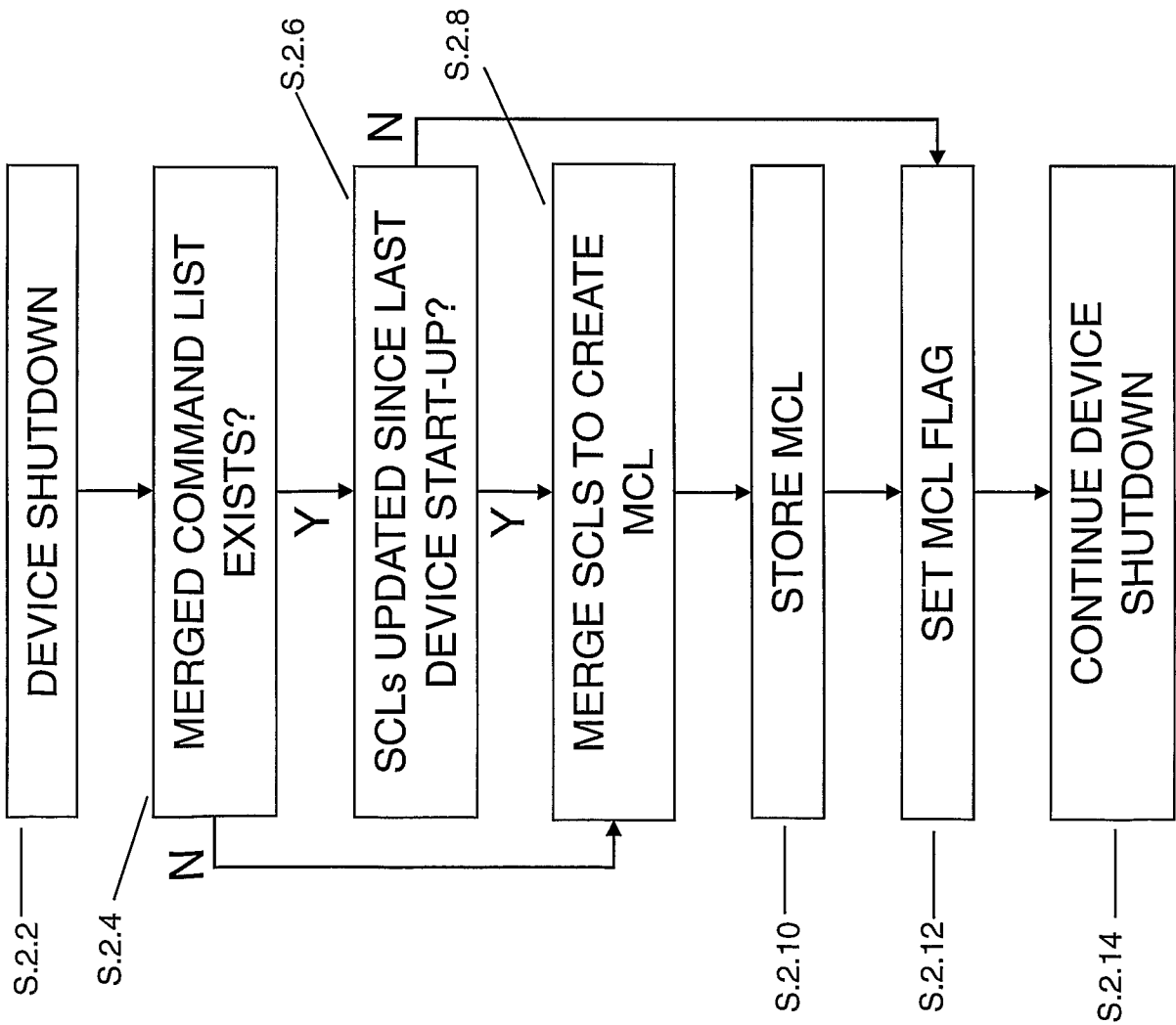


Figure 2

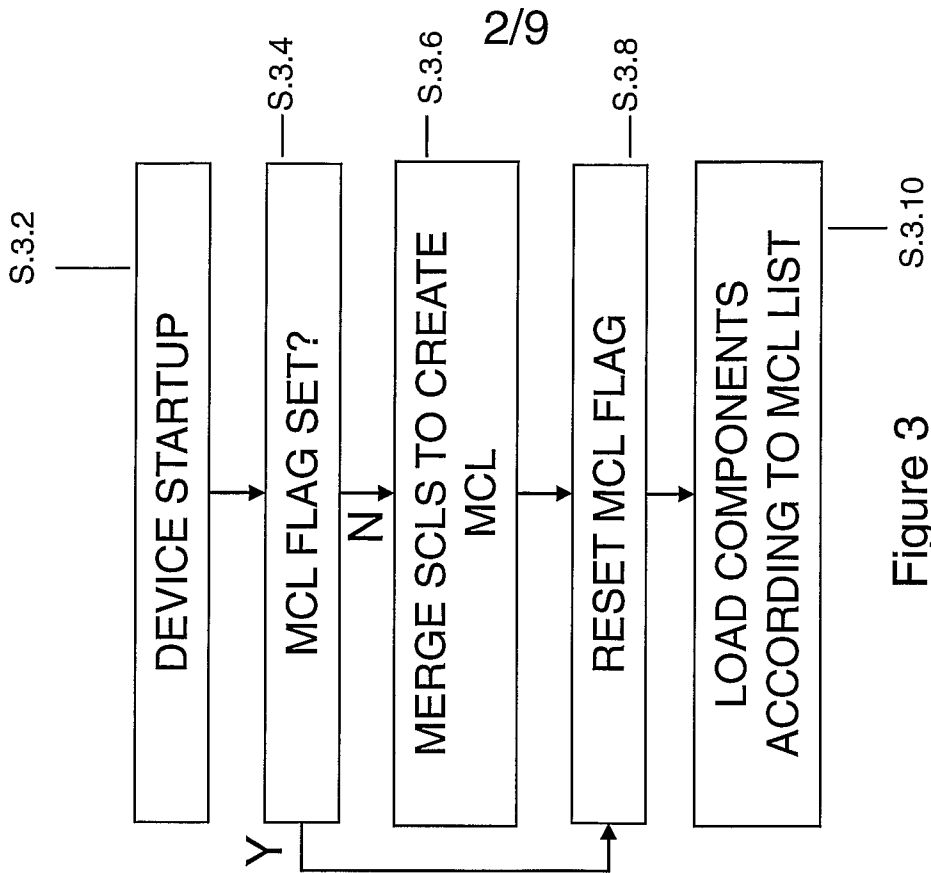


Figure 3

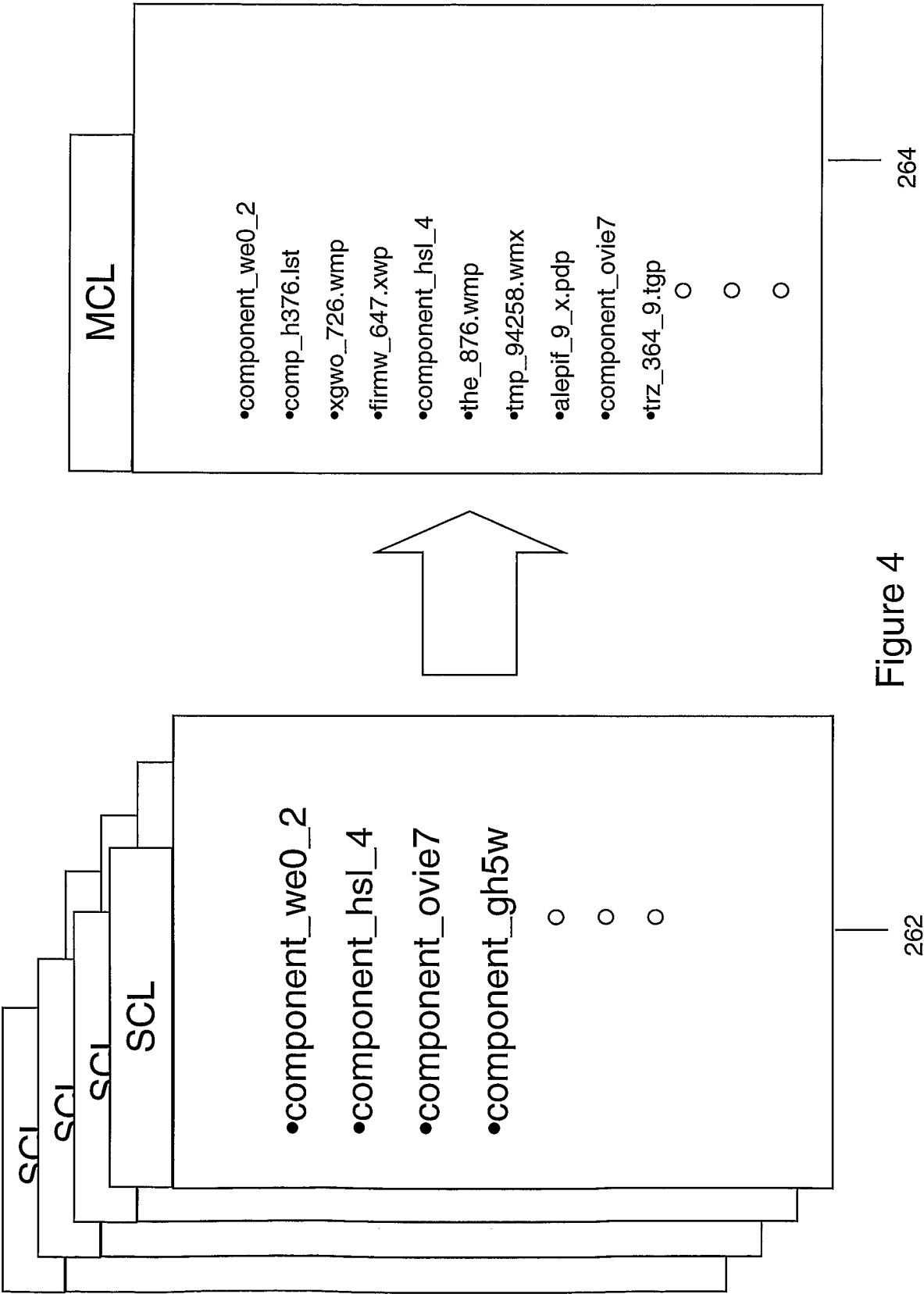


Figure 4

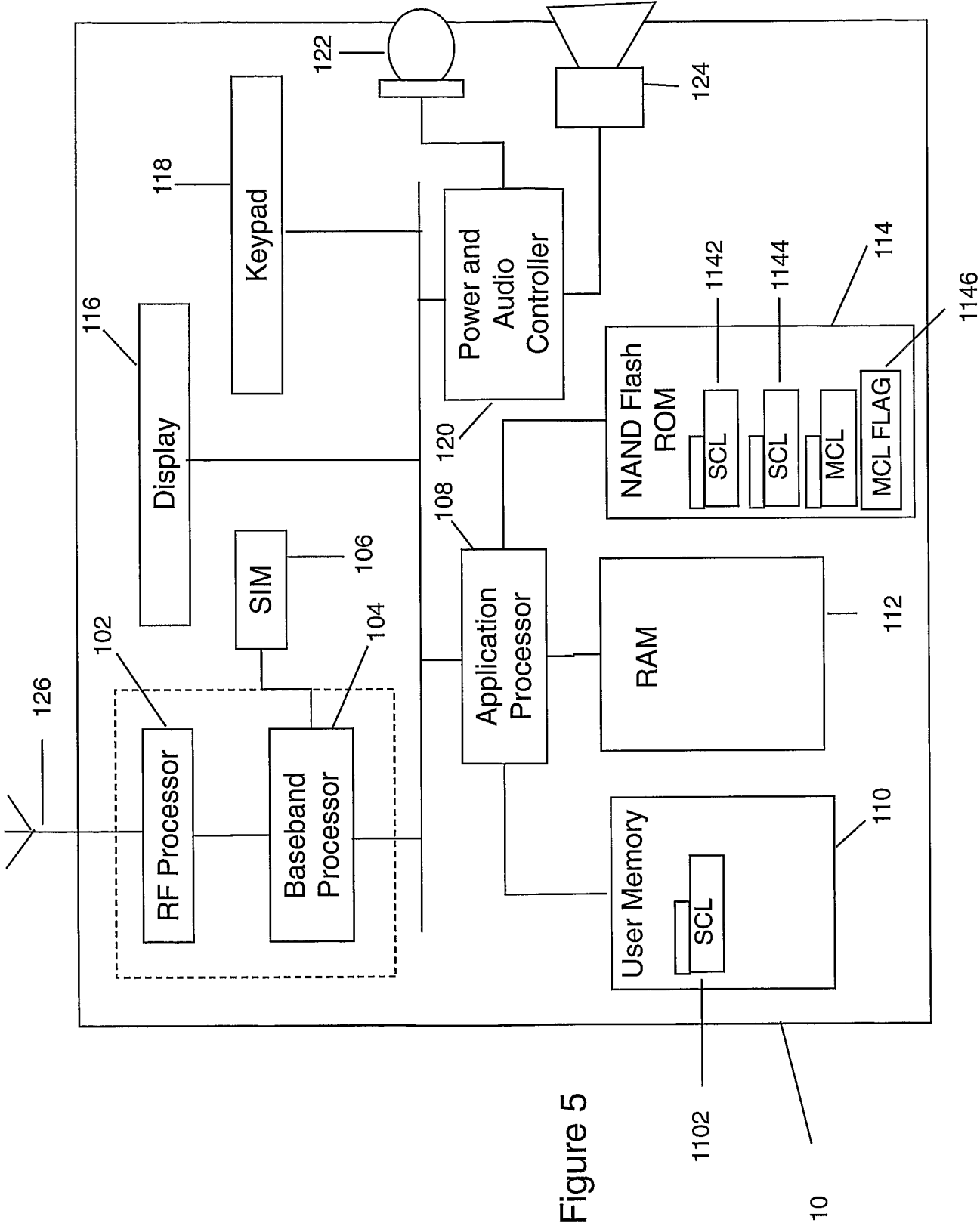
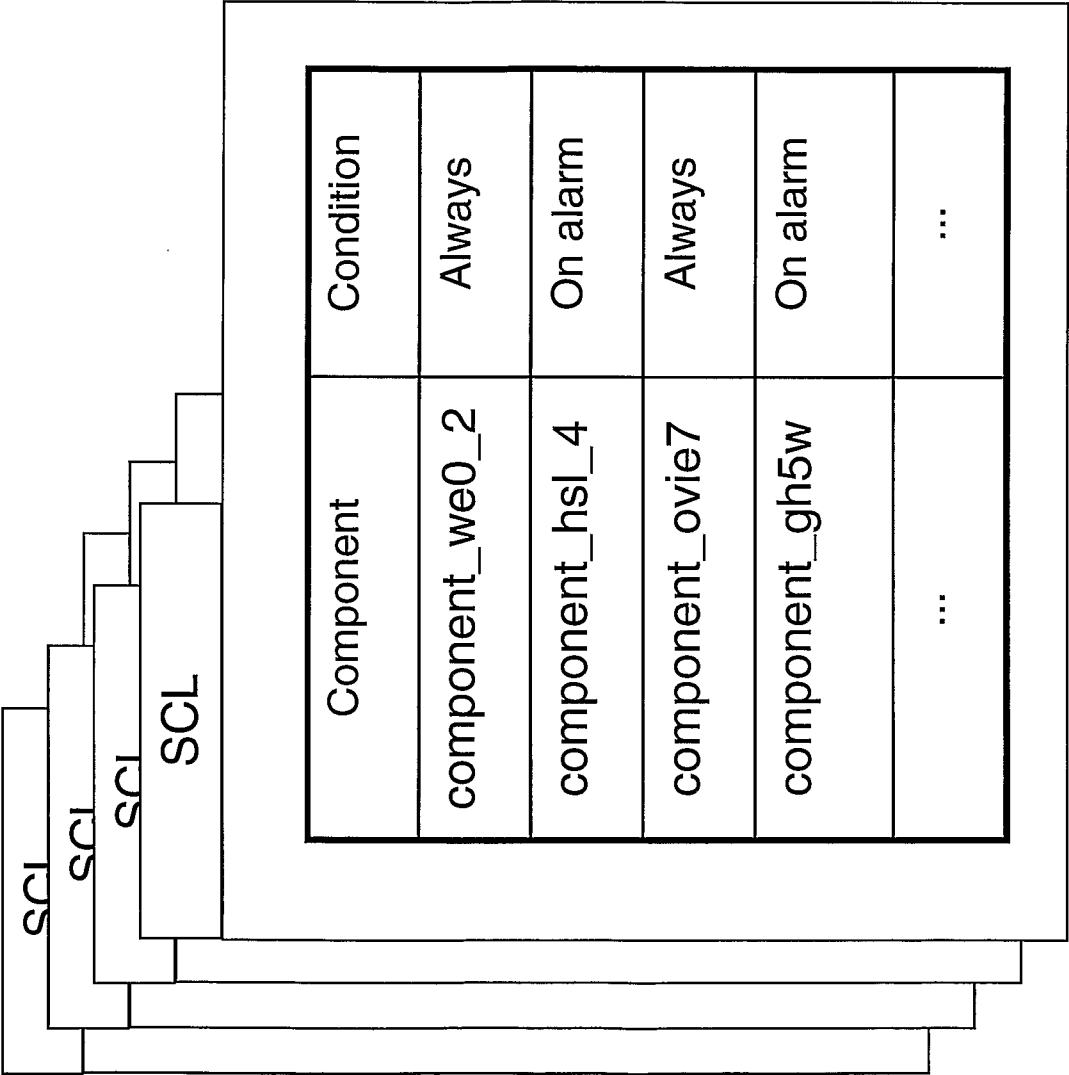


Figure 5



1142

Figure 6

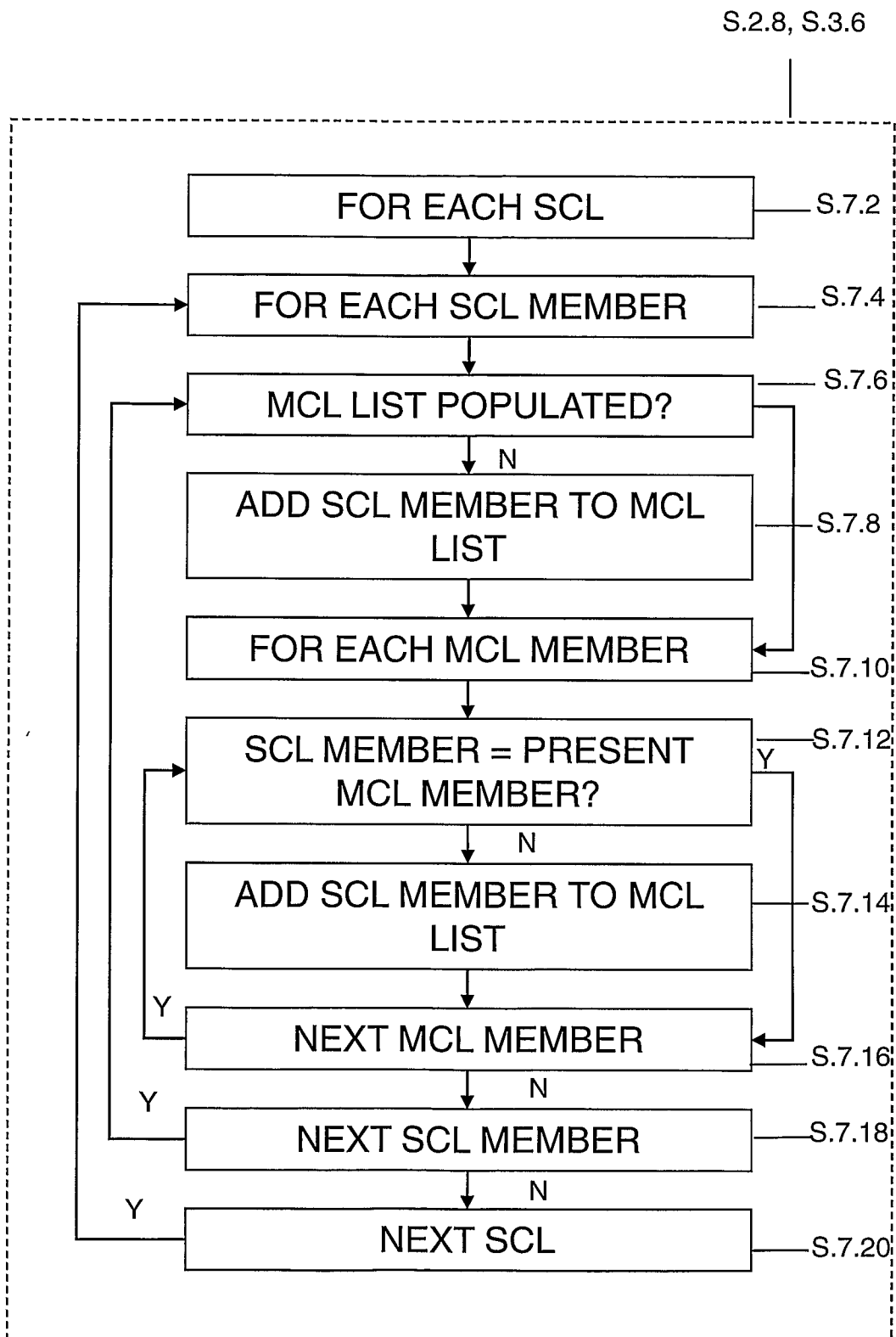


Figure 7

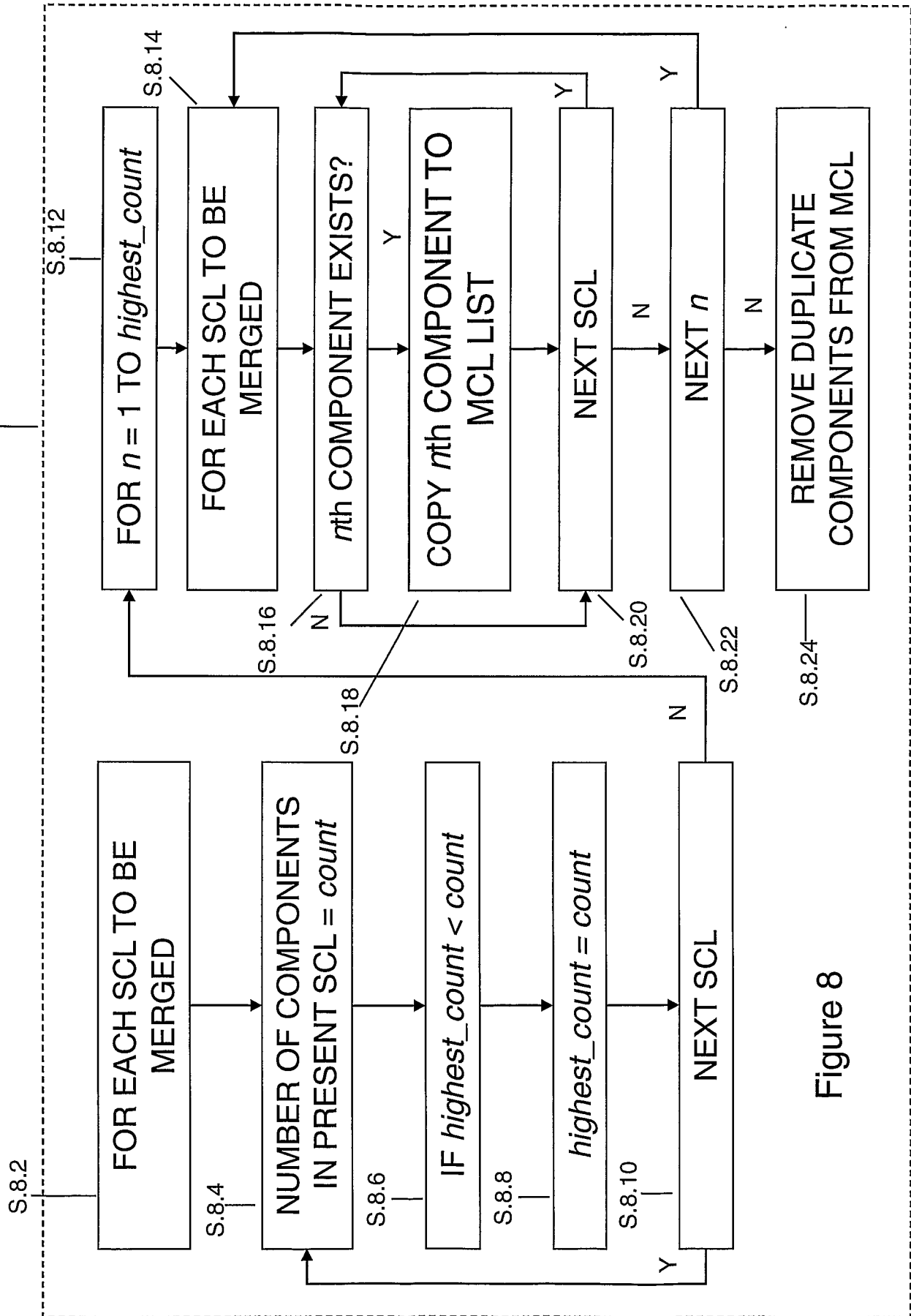


Figure 8

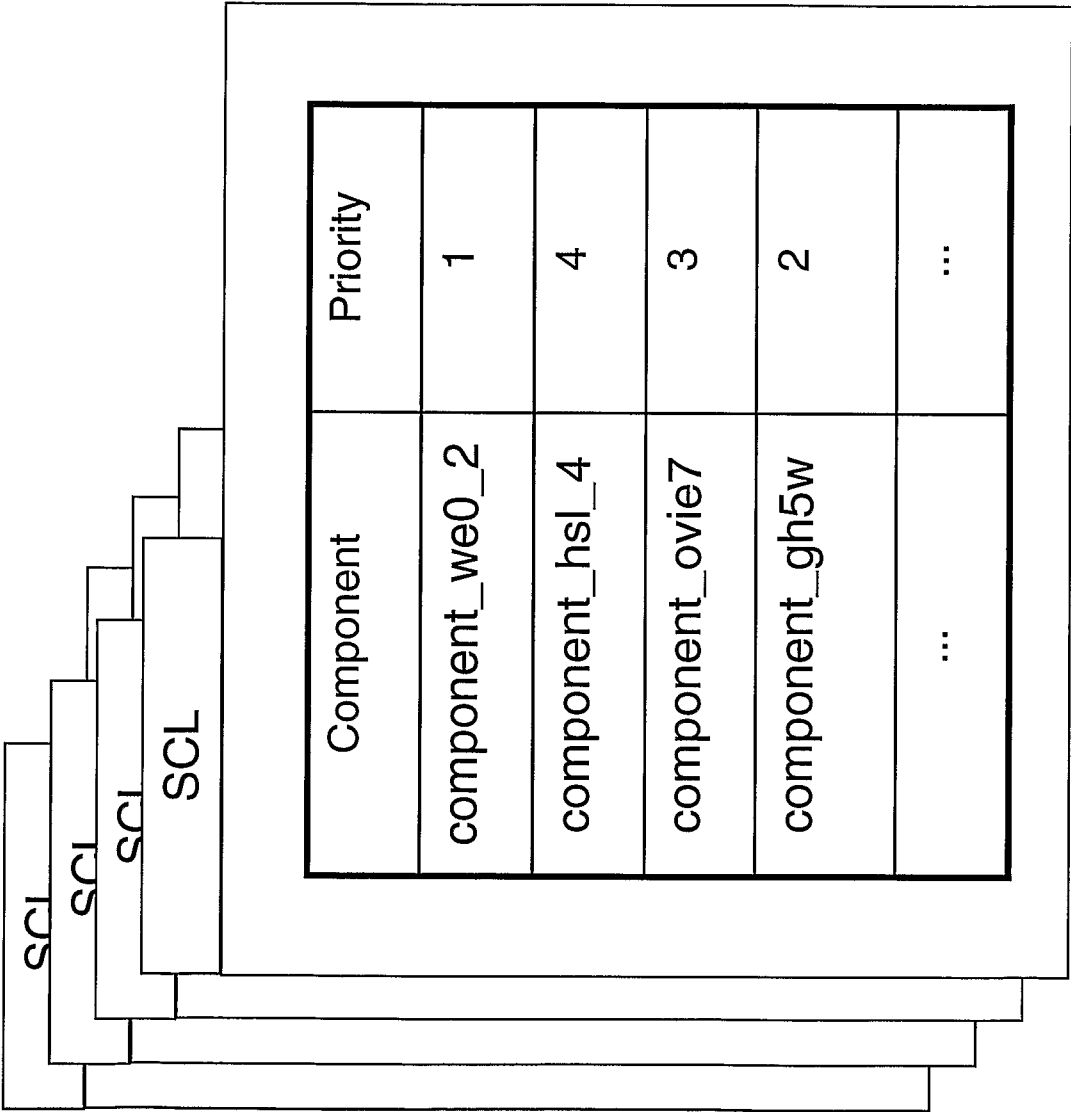


Figure 9

1142

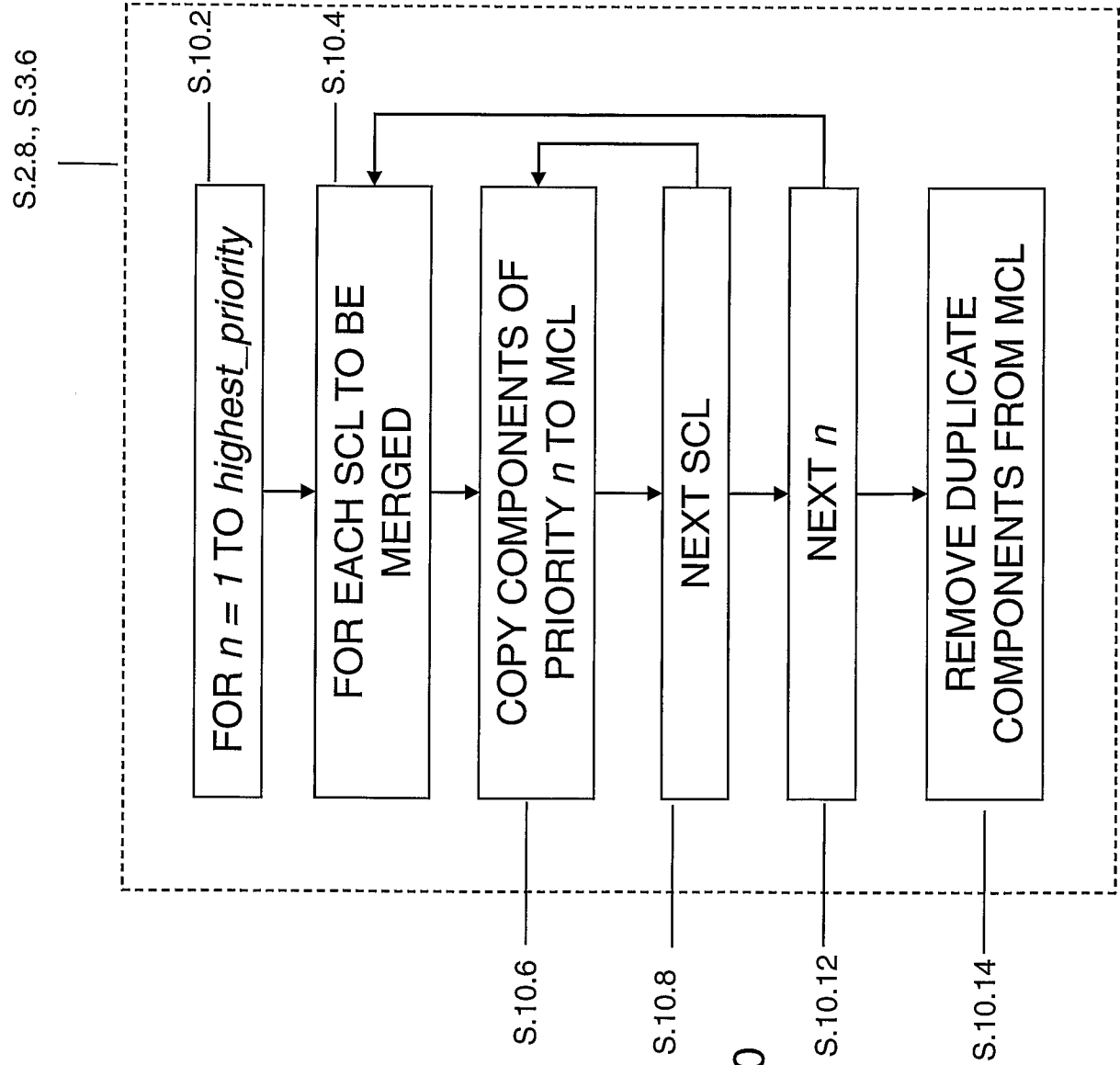


Figure 10

INTERNATIONAL SEARCH REPORT

International application No.

PCT/IB2010/001178

A. CLASSIFICATION OF SUBJECT MATTER

IPC: see extra sheet

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC: H06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE,DK,FI,NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-INTERNAL, WPI DATA, PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 20060277402 A1 (WAKABAYASHI), 7 December 2006 (07.12.2006), paragraphs (0003),(0010)-(0012) --	1-16
X	US 20040139439 A1 (MACHIDA ET AL), 15 July 2004 (15.07.2004), paragraphs (0002),(0027),(0146) --	1-16
X	US 20020169949 A1 (SHENASSA ET AL), 14 November 2002 (14.11.2002), paragraphs (0003), (0004),(0042) --	1-16
A	EP 1267550 A2 (SUN MICROSYSTEMS, INC.), 18 December 2002 (18.12.2002), paragraph (0012) --	1-16

☒ Further documents are listed in the continuation of Box C.☒ See patent family annex.

* Special categories of cited documents:

- "A" document defining the general state of the art which is not considered to be of particular relevance
- "E" earlier application or patent but published on or after the international filing date
- "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- "O" document referring to an oral disclosure, use, exhibition or other means
- "P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

4 October 2010

Date of mailing of the international search report

11 -10- 2010

Name and mailing address of the ISA/
Swedish Patent Office
Box 5055, S-102 42 STOCKHOLM
Facsimile No. +46 8 666 02 86

Authorized officer

Sture Elnäs / JA A
Telephone No. +46 8 782 25 00

INTERNATIONAL SEARCH REPORT

International application No.

PCT/IB2010/001178

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6212672 B1 (KELLER ET AL), 3 April 2001 (03.04.2001), column 21, line 22 - line 40 --	1-16
A	US 20070234302 A1 (SUZUKI ET AL), 4 October 2007 (04.10.2007), paragraphs (0129),(0130) -- -----	1-16

International patent classification (IPC)**G06F 9/445** (2006.01)**Download your patent documents at www.prv.se**

The cited patent documents can be downloaded:

- From "Cited documents" found under our online services at www.prv.se (English version)
- From "Anförda dokument" found under "e-tjänster" at www.prv.se (Swedish version)

Use the application number as username. The password is **BCJYAKROIQ**.

Paper copies can be ordered at a cost of 50 SEK per copy from PRV InterPat (telephone number 08-782 28 85).

Cited literature, if any, will be enclosed in paper form.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/IB2010/001178

US	20060277402	A1	07/12/2006	NONE			
US	20040139439	A1	15/07/2004	NONE			
US	20020169949	A1	14/11/2002	NONE			
EP	1267550	A2	18/12/2002	NONE			
US	6212672	B1	03/04/2001	AU	6671698 A	22/09/1998	
				WO	9839698 A	17/12/1998	
US	20070234302	A1	04/10/2007	NONE			