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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 December 2003 (04.12.2003)

PCT

(10) International Publication Number
WO 03/100614 A2

(51) International Patent Classification[7]: G06F 9/54

(21) International Application Number: PCT/US02/16371

(22) International Filing Date: 24 May 2002 (24.05.2002)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **MEDIUS, INC.** [US/US]; Maritime Building, Suite 508, 911 Western Avenue, Seattle, WA 98104 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **PRESTON, Dan, Alan** [US/US]; 11621 Meadowmeer Circle, NE, Bainbridge Island, WA 98110 (US). **LUTTER, Robert, Pierce** [US/US]; 2909 N. 32nd Street, Tacoma, WA 98407 (US).

(74) Agent: **FORD, Stephen, S.**; Marger Johnson & McCollom, P.C., 1030 SW Morrison Street, Portland, OR 97205 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

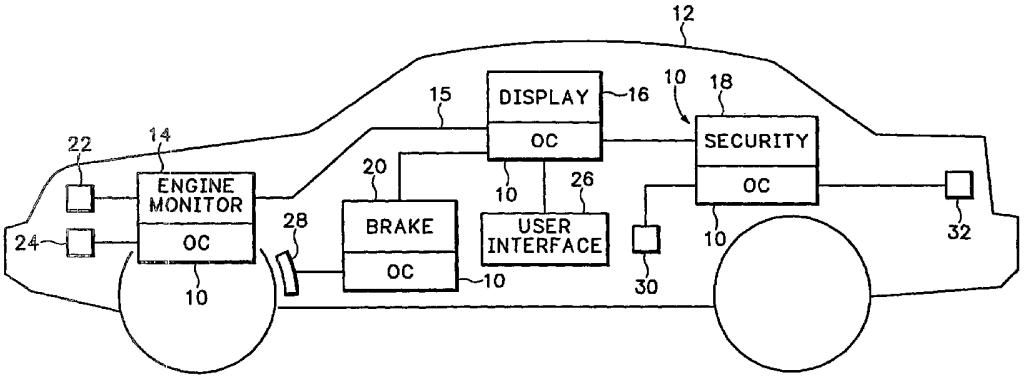
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

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

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

(54) Title: VEHICLE MONITORING SYSTEM



(57) Abstract: A communication system for a mobile vehicle, home, or office environment includes multiple processors. The same or different communication links can connect the multiple processors together. The multiple processors each run an open communication system that enables data or messages to be effectively transferred and processed for real-time applications performed by the multiple processors. A dynamic configuration system runs on the multiple processors and includes a device manager, configuration manager, and data manager. The device manager automatically detects and adds new devices to the multiprocessor system, and the configuration manager automatically reconfigures which processors run the real-time applications. The data manager identifies the type of data generated by the new devices and identifies which devices in the multiprocessor system are able to process the data.

WO 03/100614 A2

VEHICLE MONITORING SYSTEM

BACKGROUND

Cars include many different electro-mechanical and electronic applications. Examples include breaking systems, electronic security systems, radios, Compact Disc (CD) players, internal and external lighting systems, temperature control systems, locking systems, seat adjustment systems, speed control systems, mirror adjustment systems, directional indicators, etc. Generally the processors that control these different car applications do not talk to each other. For example, the car radio does not communicate with the car heating system or the car breaking system.

This means that each one of these car applications has to provide a separate standalone operating system. For example, separate processors and separate user interfaces are required for the car temperature control system and for the car audio system. Many of these different car processors may be underutilized since they are only used intermittently for one or a few applications.

Even when some processors in the car do talk to each other, they are usually so tightly coupled together that it is impossible to change any one of these processors without disrupting all of the systems that are linked together. For example, some cars may have an interface on the dashboard that controls both internal car temperature and a car radio. The car radio cannot be replaced with a different model and still work with the dashboard interface and the car temperature controller.

Integration of new applications into a car is also limited. Car applications are designed and selected well before the car is ever built. A custom wiring harness is

then designed to connect all the car applications selected for the car. A car owner can not later incorporate new applications into the existing car system. For example, a car may not originally come with a car navigation system. An after market navigation system from another manufacturer cannot be integrated into the existing car control system.

Because after market devices can not be integrated into car control and interface systems, it is often difficult for the driver to try and operate these after market devices. For example, the car driver has to operate the after market navigation system from a completely new interface, such as the keyboard and screen of a laptop computer. The driver then has to operate the laptop computer, not from the front dashboard of the car, but from the passenger seat of the car. This makes many after market devices both difficult and dangerous to operate while driving.

The present invention addresses this and other problems associated with the prior art.

SUMMARY OF THE INVENTION

A multiprocessor system used in a car, home, or office environment includes multiple processors that run different real-time applications. A dynamic configuration system runs on the multiple processors and includes a device manager, configuration manager, and data manager. The device manager automatically detects and adds new devices to the multiprocessor system, and the configuration manager automatically reconfigures which processors run the real-time applications. The data manager identifies the type of data generated by the new devices and identifies which devices in the multiprocessor system are able to process the data.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram of a car that has multiple processors that each run an open communication system.

FIG. 2 is a block diagram of the open communication system shown in FIG. 1.

FIG. 3 is a flow diagram showing how a priority manager processes outgoing data in the open communication system.

FIG. 4 is a flow diagram showing how the priority manager receives data in the open communication system.

FIG. 5 is a flow diagram showing how a logging manager processes data in the open communication system.

FIG. 6 is a flow diagram showing how a security manager processes data in the open communication system.

FIG. 7 is a diagram showing one example of how the open communication system is used by different processors.

FIG. 8 is a diagram of a tracking report that is generated by the open communication system.

FIG. 9 is a flow diagram showing how different image data is processed and transmitted using the open communication system.

FIG. 10 is a flow diagram showing how the transmitted image data in FIG. 9 is received and processed using the open communication system.

FIG. 11 is a block diagram showing another example of how the open connection system operates.

FIG. 12 is a diagram of a car that has multiple processors that each run a Dynamic Configuration (DC) system.

FIG. 13 is a detailed diagram of the dynamic configuration system shown in FIG. 12.

FIGS. 14 and 15 are diagrams showing an example of how the DC system operates.

FIGS. 16 and 17 are diagrams showing how a device manager in the DC system operates.

FIGS. 18-21 are diagrams showing how a reconfiguration manager in the DC system operates.

FIGS. 22 and 23 are diagrams showing how a data manager in the DC system operates.

FIG. 24 is a diagram showing different multiprocessor systems that can use the DC and OC system.

DETAILED DESCRIPTION

FIG. 1 shows a car 12 that includes multiple processors 14, 16, 18 and 20. The engine monitor processor 14 in one configuration monitors data from different sensors 22 and 24 in the car engine. The sensors 22 and 24 can be any sensing device such as sensors that monitor water temperature, oil temperature, fuel consumption, car speed, etc. The break control processor 20 monitors and controls an Automatic Breaking System (ABS) 28. The display processor 16 is used to control and monitor a graphical or mechanical user interface. The security processor 18 monitors and controls latches and sensors 30 and 32 that are used in a car security system.

Typical networks, such as in an office network environment, enable multiple computers to communicate with each other. Applications such as printing jobs can be launched from any one of the networked computers. If one of the networked

computers crashes or is busy, a user must manually send the job to another computer. The other computer then handles the task like any other locally received task.

In a car environment, tasks must be processed with different priorities in real-time. For example, the breaking tasks in the break processor 20 have to be processed with a high priority while a radio selection task performed in the display processor 16 can be processed with a relatively low priority. The processors 14, 16, 18 and 20 all include software that runs an Open Communication (OC) system 10 that enables the multiple processors to transfer data and exchange messages for performing these real-time car applications.

If the processor 20 currently running the high priority breaking application fails, the OC system 10 allows the breaking tasks to be offloaded to another processor in car 12, such as the display processor 16. The OC system 10 automatically assigns a high priority to the breaking tasks that allow the breaking tasks to override lower priority tasks, such as the radio application, that are currently being performed in display processor 16.

The OC system 10 also ensures that data in each processor is processed in a secure manner for the car environment. The security portion of the OC system 10 prevents unauthorized devices from accessing the different car processors. The OC system 10 also includes a logging portion that allows data in the car system to be automatically logged. This is important for accident reconstruction purposes. The OC system 10 also allows different processors to communicate over different communication protocols and hardware interfaces. Any processor that includes an OC system 10 can be integrated in the system shown in FIG. 1. This allows different processors and different applications can be seamlessly replaced and added to the overall multiprocessor system.

The description below gives only a few examples of the different processors and different applications that can implemented using the OC system 10. However, any single or multiprocessor system located either inside or outside of car 12 can communicate and exchange data using the OC system 10. It should also be understood that the OC system 10 can be used in any real-time environment such as between processors in appliances and computers used in the home.

FIG. 2 is a block diagram of the communication managers used in the OC system 10 described in FIG. 1. The different communication managers in the OC system 10 are configured to provide the necessary control for operating a distributed processor system in a real-time car environment. Applications 48 are any of the different applications that can be performed for the car 12 shown in FIG. 1. For example, applications can include car displays, breaking control, security systems, sensor monitoring, airbag deployment, etc. One or more applications can be run in the same processor at the same or at different times.

A car interface manager 46 operates as an Application Programmers Interface (API) that can be implemented in any variety of different languages such as Java, C++, Extensible Markup Language (XML) or HyperText Markup Language (HTML), etc. The car interface manager 46 enables applications 48 to be written in any variety of different languages. This prevents the applications 48 from having to be written specifically for the car environment or for a specific communication protocol. Thus, applications written for other systems can be reused in the car system. The car interface manager 46 reads basic processing and data transfer commands needed to transfer data and messages between different processors and storage mediums inside or outside the car 12.

For clarity the terms 'message' and 'data' are used interchangeably below. After a message passes through the generic car interface manager 46, a priority manager 44 determines a priority value for the message that determines how the message is processed both in the local processor 50 and in other processors such as processor 50. Referring to FIG. 3, an outgoing message is identified by the priority manager 44 in block 60. A priority for the message is identified in block 62 by reading a priority value that the generic car interface manager 46 has attached to the message.

In block 64, the priority manager 44 compares the priority value for the outgoing message with the priority values for other messages in the processor. The priority manager 44 ranks the outgoing message with respect to the other messages and then sends the message to the logging manager 42 in block 66. For example, there may be several messages that either need to be output or received by a particular processor. An output message with a high priority value, such as a crash indication message, will have higher priority than other messages and will therefore be immediately transmitted by the processor.

FIG. 4 shows how the priority manager 44 receives messages from other processors. There may be multiple applications running on the same processor and multiple messages and data sent from other processors to those applications. For example, multiple sensors may be sending different types of data to a video display application running on one of the processors. That same processor may also be receiving different types of sensor data for running an airbag deployment application. The priority manager 44 determines the order that messages are processed by the different applications that reside on that processor.

In block 68, the priority manager 44 reads the priority labels for incoming messages. If the priority of the message is not high enough to run on the processor in block 70, the data or message is rejected in block 76. The priority manager 44 may send out a message to the sending processor indicating the message has been rejected. In some situations, the message or data may have such a low priority that an acknowledge message does not have to be sent back to the sending processor. For example, inside temperature data from a temperature sensor may be sent to one or more processors with no requirement that the processor accept or acknowledge the data. In this case the temperature data is sent with a very low priority value that indicates to the priority manager 44 that no message needs to be sent back to the temperature sensor even if the data is rejected.

The priority manager 44 in block 72 ranks the priority of the incoming message in relation to the priorities of all the other messages in the processor. The priority manager in block 74 decides according to the ranking whether the message should be put in a queue or sent directly to the application for immediate processing. For example, a crash indication message may have a high enough priority to cause the priority manager 44 to delay all data currently being processed by all other applications in the same processor. The priority manager 44 directs all the applications to wait while the current high priority crash indication message is processed. The other data and messages are re-queued in the processor after the crash indication message has been processed.

Referring to FIGS. 2 and 5, a logging manager 42 controls what data is logged by different processors. It may be important to log critical failures that occur during an accident. For example, it may be important to verify that a particular processor sent an air bad deployment message and that another processor successfully received

the airbag deployment message. This would allow insurance companies and other entities to reconstruct accidents by identifying when and where different messages were sent and received.

The logging manager 42 receives either an incoming message over a communications link for sending to an application 48 or receives an outgoing message from one of the applications 48 for sending out over the communications link to another processor in block 80. The logging manager 42 reads a logging label in the message in block 82. If the logging label indicates that no logging is required, the message is sent on to the next communication manager in block 88. If it is an outgoing message it is sent to the security manager 40 (FIG. 2). If it is a incoming message it is sent to the priority manager 44. If the message requires logging, the logging manager 42 stores the message in a memory in block 86. The logging label may indicate a particular type of memory for logging, such as a nonvolatile Flash memory or, if available, a high volume hard disk peripheral memory.

The logging manager 42 in each processor, provides the OC system 10 with the unique ability to track when and where messages are sent and received at different processors in the multiprocessor car system. This is important in accident reconstruction since the logging managers 42 identify which processors and applications were disrupted or damaged and also the sequence that the different processors and associated applications failed.

The logging manager 42 can also track unauthorized messages and data that may have caused any of the processors in the car to crash. For example, an audio processor that handles audio applications in the car may crash due to unauthorized downloading of MP3 music from a laptop computer. The logging manager 42 can log the unauthorized data received from the laptop MP3 player. The logging manager 42

logs any data that does not have a particular security or priority label value. A system administrator can then down load the MP3 data to identify what caused the audio processor to crash.

Referring to FIGS. 2 and 6, a security manager 40 provides security for applications both receiving and transmitting messages. For instance, a laptop computer may be connected to a Ethernet port in the car 12 (FIG. 1). If the laptop computer does not use the OC system 10, data from that laptop application is not allowed to access certain processors or certain applications in the car 12. For example, audio data should not be sent to a processor that performs car breaking control.

The security manager 40 in block 90 reads a message either received from an application on the same processor or received over a communication link from another processor. The security manager 40 determines if there is a security value associated with the message in block 92. If there is no security value associated with the data, the security manager 40 may drop the data in block 100. However, some applications, such as a processor that plays audio data may not require a security label. In this case, the security manager in block 94 allows the data to be passed on to the application in block 98.

In other instances the data or message may have a security value, but that security value is not sufficient to allow processing on the present applications. For example, data for car security monitoring may be sent to a processor that controls air bag deployment and an automatic breaking system. The two currently running applications may set a minimum security level for receiving data. If data received from other processors do not have that minimum security level in block 96, the data is dropped in block 100. Otherwise, the data or message is passed on to the next

communication layer for further processing in block 98. Thus the security manager 40 prevents unauthorized data or messages from effecting critical car applications.

Referring back to FIG. 2, an operating system layer 38 identifies the communication platform used for communicating the data or message over a link identified in a hardware/link interface 36. The operating system 38 formats the message for the particular communication stack and medium used by the identified link 54. For example, the operating system layer 38 may identify a first message being transmitted over a Bluetooth wireless link and a second message transmitted over a Transmission Control Protocol/Internet Protocol (TCP/IP) packet switched link. The data or message adds whatever headers and formatting is necessary for transmitting the first message over the Bluetooth wireless link and the second message over the TCP/IP hardwired link.

The hardware/link interface 36 includes the software and hardware necessary for interfacing with different communication links 54. For example, the two processors 50 and 52 may communicate over a Ethernet link, 802.11 wireless link, or hardwired Universal Serial Bus link, etc. The software necessary for the two processors to communicate over these different interfaces is known to those skilled in the art and is therefore not described in further detail.

FIG. 7 describes one example application that uses the OC system 10 described above in FIGS. 1-6. A car 102 includes an radar sensor 104 that is controlled by a radar processor 106 . The radar sensor 104 is located in the front grill of car 102. An InfraRed (IR) sensor 110 is controlled by an IR processor 112 and is located on the front dash of car 102. A breaking system 123 is controlled by a break control processor 122. The IR processor 112 is connected to a fusion processor 114 by an Ethernet link 116 and the radar processor 106 is connected to the fusion

processor 114 by a 802.11 wireless link 108. The break processor 122 is connected to the fusion processor 114 by a CAN serial link 120. The fusion processor 114 is also coupled to a display screen 118.

The radar sensor 104 in combination with the radar processor 106 generates Radar Track Reports (RTRs) 130 that are sent to the fusion processor 114. The IR sensor 110 in combination with the IR processor 112 generate Infrared Track Reports (ITRs) 128 that are sent to the fusion processor 114.

Referring to FIG. 8, each track report 128 and 130 includes communication link headers 132 for communicating over an associated interface medium. In this example, the radar track report 130 includes the link headers 132 necessary for transmitting data over the 802.11 link 108. Similarly, the infrared track report 128 includes the link headers 132 for transmitting data over the Ethernet link 116.

The track reports 128 and 130 include Open Communication (OC) labels 133 for performing the OC operations described above. A security label 134 is used by the security manager for preventing unauthorized data from being downloaded into one of the car processors and disrupting applications. A logging label 136 is used by the logging manager to identify data that needs to be logged in a local memory. The priority label 138 is used by the priority manager for scheduling messages or data to the applications run by the processors. The radar processor 106 and IR processor 112 also send a time of measurement 140 and other data 142 from the radar sensor 104 and IR sensor 110, respectively. The data 142 can include kinematic states of objects detected by the sensors.

FIGS. 9 and 10 show one example of how the radar and infrared sensor data is processed by the OC system 10. One or both of the radar processor 106 and the IR processor 112 may generate image data 150 and 152 for the area in front of the car

102 (FIG. 7). For simplicity, the discussion below only refers to an image generated by radar sensor 104. At a first time $t = t_1$, sensor 104 detects a small far away object 154. At another time $t = t_2$, sensor 104 detects a large up-close object 156.

An image processing application in the processor 106 identifies the object 154 as a small far away object in block 158. The image and kinematic data for the object is output by the OC system 10 as a radar track report 130. The security manager 40 (FIG. 2) in the radar processor 106 adds a security label 134 to the report in block 160 and the logging manager 42 may or may not add a logging label to the report in block 162. In this example, the object 154 has been identified by the image processing application as a small far away object. Therefore, the logging manager does not label the track report for logging. The priority manager 44 (FIG. 2) adds a priority label 138 (FIG. 8) to the report in block 164. Because the image processing application identifies the object 154 as no critical threat (small far away object), the priority label 138 is assigned a low priority value in block 164.

The OC system 10 then formats the radar track report in block 168 according to the particular link used to send the report 130 to the fusion processor 114. For example, the operating system 38 and the hardware/link interface 36 (FIG. 2) in the radar processor 106 attaches link headers 132 to the track report 130 (FIG. 8) for transmitting the report 130 over the 802.11 link. The track report 130 is then sent out over the link 108 in block 168 to the fusion processor 114.

Referring next to FIGS. 7-10, the fusion processor 114 includes a wireless interface 119 that communicates with the wireless 802.11 link 108 and an Ethernet interface 117 that communicates with the Ethernet link 116. The hardware/link interface 36 in the fusion processor OC system 10 uses the link headers 132 (FIG. 8)

to receive the radar track report 130 in block 182 and process the reports in block 184 (FIG. 10).

The OC system 10 reads the security label in block 186 to determine if the track report has authority to be processed by the fusion processor 114. If the track report passes the security check performed by the security manager in block 186, the logging manager in block 188 checks to see if either the received radar data needs to be logged. In this example, the image processing application in the radar processor identified the object 154 (FIG. 9) to be within a particular size range and distance range that does not indicate a critical crash situation. Therefore, the track report 130 was not labeled for logging. The fusion processor 114 therefore does not log the received report in block 188.

Because the image 150 was identified as noncritical, the priority label 138 (FIG. 8) for the track report 130 is given a low priority value. The fusion processor 114 ranks the track report with the other data that is being processed and then processes the report according to the ranking.

Different applications in the fusion processor 114 may or may not be performed depending on the track report. For example, the object 154 may be sent to a video display in block 194. However, the fusion processor 114 will not send a break command in block 196 to the car breaking system 123. This is because the image has been identified as non-critical. Similarly, no audio warning is sent to the car audio system in block 198 because the object has been identified as non-critical.

Referring back to FIG. 9, in another example, the IR processor 112, the radar processor 106, or both, in block 170 detect at time t_2 an object 156 that is large and close to the car 102. For simplicity, it is assumed that only the IR processor 112 has identified object 156. The IR processor 112 generates a track report 128 in block 170

and the OC system in the IR processor 112 adds a security label 134 (FIG. 8) to the report in block 172. Because the object 156 has been identified as being within a predetermined size and within a predetermined range of car 102 (critical data), the logging manager in the IR processor 112 assigns a logging label value 136 to the IRT 128 that directs all processors to log the image data 142. The image data is logged by the IR processor 112 in a local memory in block 174.

Because the IR track report 128 has been identified as critical data, the priority manager 44 in the IR processor 112 assigns a high priority label value 138. This high priority value is read by the operating system 38 and interface hardware 36 (FIG. 2) in blocks 178 and 180. Accordingly the IR track report 128 is given preference when being formatted in block 178 and transmitted in block 180 over Ethernet link 116 to the fusion processor 114.

Referring again to FIG. 10, the IR track report 128 is received by the fusion processor 114 in block 182 and the link processing performed in block 184. This link processing is known to those skilled in the art and is therefore not described in further detail The report may be given higher link processing priority in the fusion processor 114 based on a priority value assigned in the link headers 132.

The security manager 40 in the fusion processor 114 confirms there is an acceptable value in the security label in block 186 and then passes the IR track report 128 to the logging manager in block 188. The logging manager 42 in the fusion processor 114 reads the logging label and accordingly logs the image data in a local nonvolatile memory. This provides a history of the image 156 that was detected by the IR sensor 110.

The logged image data may then be used in subsequent accident analysis. For example, an accident reconstruction specialist can download the logged image data or

message in both the IR processor 112 and in the fusion processor 114 to determine when the image data 140 and 142 was first detected. It can then be determined whether the image data was sent by the IR processor 112 and received by the fusion processor 114.

The priority manager reads the priority label 138 in block 190 and determines that the IR track report has a high priority. Accordingly, the track report is immediately sent to different applications in block 192. The priority manager 44 may first send the track report to the break control application in block 196. The break control application immediately sends a break command 125 (FIG. 7) to the break processor 122.

The logging manager 42 in the fusion processor 114 adds a logging label 136 to the outgoing break command 125. Both the fusion processor 114 and the break control processor 122 will then both log the break command 125. Thus, not only is the sequence of transmissions of the image data and messages logged in both the IR processor 112 and fusion processor 114 but also the sequence of the break message 125 from the fusion processor 114 to the break processor 122. This further adds to any accident analysis data that may need to be obtained from the car if an accident occurs.

The IR data may also be sent to an audio application in block 198 that immediately sends out an audio alarm over the car stereo system or out over a car horn. This automatically warns both the car driver and the object 156 in front of car 102 of a possible collision. In a third application, the fusion processor 114 may send the IR image data to an image display 118 in block 194.

FIG. 11 is a block diagram showing another example of how the OC 10 exchanges information according to the type of data independently of the physical

links that connect the different applications together. A processor A runs an application 202. In this example, the application 202 is an IR processing application that receives IR data from an IR sensor 200 and outputs the IR data as a sensor report. A processor B runs a fusion processing application 220 that controls other car functions in part based on the IR sensor report.

The OC system 208 includes a control table 212 that includes several parameters associated with a SENSOR REPORT 210. For example, the SENSOR REPORT 210 may need to include a priority label, a security label or a logging label. The security label also includes one or more locations where the SENSOR REPORT 210 should be sent. The IR application 202 includes a CONNECT TO SEND (SENSOR REPORT) command that the OC 10 then uses to establish a slot in memory for the SENSOR REPORT. When IR data is received from the IR sensor 200, the IR application 202 generates sensor data (FIG. 8) for the SENSOR REPORT 210 and stores that sensor data in the memory slot established by the OC system 10. The sensor data is contained within the application data section 139 of the sensor report shown in FIG. 8. The IR application 202 then issues the SEND(SENSOR REPORT) command 206 to notify the OC 10 that there is a SENSOR REPORT in the reserved slot in memory.

The OC system 10 attaches a security label 134, logging label 136 and priority label 138 to the SENSOR REPORT 210 as described previously in FIG. 8. The OC system 10 then adds the necessary link headers 132 (FIG. 8) that are required to send the SENSOR REPORT 210 to other identified applications. The control table 212 includes security parameters associated with the SENSOR REPORT data type. One of the SENSOR REPORT security parameters, in addition to a security value, is an identifier 213 for the fusion application 220 running in processor B. The identifier

213 identifies whatever address, format, and other protocol information is necessary for transmitting the SENSOR REPORT 210 to the fusion application 220. The OC system 10 attaches the link headers 132 to the SENSOR REPORT 210 and then sends the report through a hardware interface 209 over a link 211 to processor B.

The fusion application 220 works in a similar manner and initiates a CONNECT TO RECEIVE (SENSOR REPORT) command to the OC system 10 running in the same processor B. The OC system 10 reserves a slot in local memory for any received SENSOR REPORTs 210. The fusion application 220 issues a WAIT ON (SENSOR REPORT) command that continuously waits for any SENSOR REPORTs 210 sent by the IR application 202. The OC system 10 control table 214 also identifies from the SENSOR REPORT data type the communication link 211, hardware interface 215 and other associated communication protocols used for receiving the SENSOR REPORT 210.

Whenever a SENSOR REPORT 210 is received, the OC system 10 in processor B performs the security, logging and priority management operations described above based on the labels 134, 136, and 138 in the sensor report 210 (FIG. 8). The OC system 10 then places the sensor data from the SENSOR REPORT 210 in the memory slot reserved in local memory. The OC system 10 detects the data in the reserved memory slot and processes the sensor data. Another portion of the fusion application 220 may send out a BRAKE command based on the sensor data. The control table 214 for the OC system 10 in processor B also includes the necessary system parameters for sending a BRAKE REPORT to another processor in the multiprocessor system, such as a brake processor.

The communication link between the fusion application 220 and the brake application may be completely different than the link between the IR application 202

and the fusion application 220. However, the fusion application 220 outputs the SENSOR REPORT and the BRAKE REPORT in the same manner. The OC system 10 then uses stored link information in the control table 214 to communicate to the IR application 202 and the brake application over different links.

Thus, the IR application 202 and the fusion application 220 do not need to know anything about the physical links, addresses, or any of the other operations that are used to transmit data over different communication links.

FIG. 12 shows a car 12 that includes a car multiprocessor system 8 having multiple processors 14, 16, 18 and 20. An engine monitor processor 14 monitors data from different sensors 22 and 24 in the car engine. The sensors 22 and 24 can be any sensing device such as sensors that monitor water temperature, oil temperature, fuel consumption, car speed, etc. A brake control processor 20 monitors and controls an Automatic Braking System (ABS) 28. A display processor 16 is used to control and monitor a graphical user interface 26. A security processor 18 monitors and controls latches and sensors 30 and 32 that are used in a car security system.

The processors 14, 16, 18 and 20 all include software that run a Dynamic Configuration (DC) system 5510 that enables new processors or devices to be automatically added and removed from the car multiprocessor system 8. The DC system 5510 also automatically reconfigures the applications running on different processors according to application failures and other system processing requirements. The DC system 5510 can be combined with OC system 10 described above.

The processor 20 may currently be running a high priority brake control application. If the processor 20 fails, the DC system 5510 can automatically download the braking application to another processor in car 12. The DC system

5510 automatically identifies another processor with capacity to run the braking control application currently running in processor 20. The DC system 5510 then automatically downloads a copy of the braking control application to the identified processor. If there is no extra reserve processing resources available, the DC system 5510 may replace a non-critical application running on another processor. For example, the DC system 5510 may cause the display processor 16 to terminate a current non-critical application and then download the brake control application along with any stored critical data.

The DC system 5510 also automatically incorporates new processors or applications into the multiprocessor system 8. For example, a laptop computer 5538 can communicate with the engine monitor processor 14 through a hardwired link 5534 or communicate to the display processor 16 through a wireless link 5536. The DC system 5510 automatically integrates the laptop computer 5538, or any other processor or device, into the multiprocessor system 8. After integrated into the multiprocessor system 8, not only can the laptop computer 5538 transfer data with other processors, but the laptop computer may also run car applications normally run by other processors in car 12.

The DC system 5510 allows the car driver to manage how different applications are processed in the car 12. As described above, a car operator may have to run an aftermarket navigation system through a GPS transceiver attached to the laptop computer 5538. The car driver has to place the laptop computer 5538 in the passengers seat and then operate the laptop computer 5538 while driving.

The DC system 5510 in the display computer 16 can automatically detect the navigation application running on the laptop computer 5538. The display computer 16 notifies the car operator through the user interface 26 that the navigation

application has been detected. The car operator can then control the navigation application through the user interface 26. Since the user interface 26 is located in the dashboard of car 12, the car operator no longer has to take his eyes off the road while operating the navigation application.

The description below gives only a few examples of the different processors, devices and applications that can be implemented using the DC system 5510. Any single or multiprocessor system located either inside or outside of car 12 can communicate and exchange data using the OC system 10. It should also be understood that the DC system 5510 can be used in any real-time environment such as between processors in different home or office appliances and different home and office computers.

FIG. 13 is a block diagram showing in more detail the Dynamic Control (DC) system 5510 located in a processor 5540 that makes up part of the multiprocessor system 8 in car 12 (FIG. 12). The DC system 5510 includes a device manager 5546 that establishes communications with new devices that are to be incorporated into the multiprocessor system 8. A configuration manager 5544 in the processor 5540 dynamically moves applications between different processors according to user inputs and other monitored conditions in the multiprocessor system 8. A data manager 5542 identifies a type of data input or output by a new processor and identifies other processors or devices in the multiprocessor system that can output data from the new device or input data to the new device.

In one example, sensors 5552 feed sensor data to processor 5540. The sensor data may include engine-monitoring data such as speed, oil temperature, water temperature, temperature inside the car cab, door open/shut conditions, etc. The sensors 5552 are coupled to processor 5540 through a link 5554, such as a proprietary

bus. A Compact Disc (CD) player 5550 is coupled to the processor 5540 through another link 5548, such as a Universal Serial Bus (USB). Graphical User Interface (GUI) 5556 displays the data associated with sensors 5552 and CD player 5550. The GUI 5556 displays the outputs from sensors 5552 using an icon 5560 to identify temperature data and an icon 5562 to identify car speed. The processor displays the CD player 5550 as icon 5562.

FIGS. 14 and 15 show an example of how two new applications are dynamically added to the multiprocessor system 8 in car 12 (FIG. 12). In FIG. 12, the DC system 5510 in processor 5540 previously detected a CD player 5550 and some sensors 5556. The CD player 5550 was displayed on GUI 5556 as icon 5558 and the temperature and speed data from sensors 5556 were displayed on GUI 5556 as icons 5560 and 5562, respectfully.

The processor 5540 is located in car 12 (FIG. 12). A passenger may bring a Digital Video Disc (DVD) player 86 into the car 12. The DVD 5586 sends out a wireless or wired signal 5588 to the processor 5540. For example, the DVD 5586 may send out signals using a IEEE 802.11 wireless protocol. The processor 5540 includes an IEEE 802.11 interface that reads the signals 5588 from DVD player 5586. If the 802.11 protocol is identified as one of the protocols used by processor 5540, the DC system 5510 incorporates the DVD player 5586 into a processor array 5557 that lists different recognized applications.

The DC system 5510 then automatically displays the newly detected DVD player 5586 on GUI 5556 as icon 5596. If capable, the car operator by selecting the icon 5596 can then display a video stream output from the DVD player 5586 over GUI 5556. The DVD player 5586 can now be controlled from the GUI 5556 on the car dashboard. This prevents the car driver from having to divert his eyes from the

road while trying to operate the portable DVD player 5586 from another location in the car, such as from the passenger seat.

Other processors or devices can also be incorporated into the multiprocessor system 8 in car 12. In another example, the car 12 drives up to a drive-in restaurant 5590. The drive-in 5590 includes a transmitter 5592 that sends out a wireless Blue tooth signal 5594. The processor 5540 includes a Blue tooth transceiver that allows communication with transmitter 5592. The DC system 5510 recognizes the signals 5594 from transmitter 5592 and then incorporates the drive-in 5590 into the multiprocessor system 8 (FIG. 12). The DC system 5510 then displays the drive-in 5590 as icon 5598 in GUI 5556.

Referring to FIG. 15, when the car operator selects the icon 5598, a menu 55102 for the driver-in 5590 is displayed on the GUI 5556. The car operator can then select any of the items displayed on the electronic menu 55102. The selections made by the car operator are sent back to the transceiver 5592 (FIG. 14). The amount of the order is calculated and sent back to the processor 5540 and displayed on menu 55102. Other messages, such as a direction for the car operator to move to the next window and pickup the order can also be displayed on the GUI 5556. At the same time, the drive-in transceiver 5592 (FIG. 14) may send audio signals that are received by the processor 5540 and played out over speakers in car 12.

FIG. 16 shows in more detail the operation of the device manager 5546 previously shown in FIG. 13. Multiple processors A, B, C and D all include device managers 5546. The device managers 5546 can each identify other devices in the multiprocessor system that it communicates with. For example, processors A, B, C and D communicate to each other over one or more communication links including a Ethernet link 5564, a wireless 802.11 link 5568, or a blue tooth link 5570.

Processor A includes a memory 5565 that stores the other recognized processors B, C and D. The data managers 5546 also identify any applications that may be running on the identified processors. For example, memory 5565 for processor A identifies an application #2 running on processor B, no applications running on processor C, and an application #4 running on processor D.

FIGS. 16 and 17 show how a new device is added to the multiprocessor system 8. Each of the existing processors A, B, C, and D after power-up are configured to identify a set or subset of the processors in the multiprocessor system 8. A new device 5572 is brought into the multiprocessor system 8 either via a hardwired link or a wireless link. For example, the device E may send out signals over any one or more of a 802.11 wireless link 5567, Blue tooth wireless link 5571 or send out signals over a hardwired Ethernet link 5569. Depending on what communication protocol is used to send signals, one or more of the processors A, B, C or D using a similar communication protocol detect the processor E in block 5574 (FIG. 17). All of the processors may be connected to the same fiber optic or packet switched network that is then used to communicate the information from processor E to the other processors.

One of the device managers 5546 in the multiprocessor system 8 checks the signals from processor E checks to determine if the signals are encrypted in a recognizable protocol in block 5576. The device manager in the processor receiving the signals from processor E then checks for any data codes from the new device signals in block 5576. The data codes identify data types used in one or more applications by processor E. A device ID for processor E is then determined from the output signals in block 5580.

If all these data parameters are verified, the device managers 5546 in one or more of the processors A, B, C and D add the new processor E to their processor arrays in block 5582. For example, processor A adds processor E to the processor array in memory 5565. After being incorporated into the multiprocessor system 8, the processor E or the applications running on the processor E may be displayed on a graphical user interface in block 5584.

FIG. 18 describes in further detail the operation of the reconfiguration manager 5544 previously described in FIG. 13. In the car multiprocessor system 8 there are four processors A, B, C and D. Of course there may be more than four processors running at the same time in the car but only four are shown in FIG. 18 for illustrative purposes. The processor A currently is operating a navigation application 55110 that uses a Global Positioning System (GPS) to identify car location. Processor B currently runs an audio application 55112 that controls a car radio and CD player. The processor C runs a car Automatic Braking System (ABS) application 55114 and the processor D runs a display application 55116 that outputs information to the car operator through a GUI 55118.

The processor D displays an icon 55120 on GUI 55118 that represents the navigation system 55110 running in processor A. An icon 55124 represents the audio application running in processor B and an icon 55122 represents the ABS application 55114 running in processor C.

The memory 55128 stores copies of the navigation application 55110, audio application 55112, ABS application 55114 and display application 55116. The memory 55128 can also store data associated with the different applications. For example, navigation data 55130 and audio data 55132 are also stored in memory 55128. The navigation data 55130 may consist of the last several minutes of tracking

data obtained by the navigation application 55110. The audio data 55132 may include the latest audio tracks played by the audio application 55112.

The memory 55128 can be any CD, hard disk, Read Only Memory (ROM), Dynamic Random Access (RAM) memory, etc. or any combination of different memory devices. The memory 55128 can include a central memory that all or some of the processors can access and may also include different local memories that are accessed locally by specific processors.

FIG. 19 shows one example of how the configuration manager 5544 reconfigures the multiprocessor system when a failure occurs in a critical application, such as a failure of the ABS application 55114. The configuration manager 5544 for one of the processors in the multiprocessor system 8 detects a critical application failure in block 55134.

One or more of the configuration managers 5544 include a watchdog function that both monitors its own applications and the applications running on other processors. If an internal application fails, the configuration manager may store critical data for the failed application. The data for each application if stored in the memory 55128 can selectively be encrypted so that only the car operator has the authority to download certain types of data.

The configuration manager detecting the failure initiates a reboot operation for that particular application. The application is downloaded again from memory 55128 and, if applicable, any stored application data. If the application continues to lockup, the configuration manager may then initiate a reconfiguration sequence that moves the application to another processor.

Failures are identified by the watchdog functions in one example by periodically sending out heartbeat signals to the other processors. If the heartbeat

from one of the processors is not detected for one of the processors, the configuration manager 5544 for the processor that monitors that heartbeat attempts to communicate with the processor or application. If the application or processor with no heartbeat does not respond, the reconfiguration process is initiated.

In another example, certain processors may monitor different applications. For example, a sensor processor may constantly monitor the car speed when the car operator presses the brake pedal. If the car speed does not slow down when the brake is applied, the sensor processor may check for a failure in either the braking application or the speed sensing application. If a failure is detected, the configuration manager initiates the reconfiguration routine.

When reconfiguration is required, one of the reconfiguration managers 5544 first tries to identify a processor that has extra processing capacity to run the failed application in block 55136. For example, there may be a backup processor in the multiprocessor system where the ABS application 55114 can be downloaded. If extra processing resources are available, the ABS application 55114 is downloaded from the memory 55128 (FIG. 18) to the backup processor in block 55142.

There may also be data associated with the failed application that is stored in memory 55128. For example, the brake commands for the ABS application 55114 may have been previously identified for logging in memory 55128 using a logging label described in co-pending U.S. application entitled: OPEN COMMUNICATION SYSTEM FOR REAL-TIME MULTIPROCESSOR APPLICATIONS, Serial No. 09/841,753 filed April 24, 2001 which is herein incorporated by reference. The logged brake commands are downloaded to the backup processor in block 55142.

If no backup processing resources can be identified in block 55136, the configuration manager 5544 identifies one of the processors in the multiprocessor

system that is running a non-critical application. For example, the configuration manager 5544 may identify the navigation application 55110 in processor A as a non-critical application. The configuration manager 5544 in block 55140 automatically replaces the non-critical navigation application 55110 in processor A with the critical ABS application 55114 in memory 55128. The processor A then starts running the ABS application 55114.

FIGS. 20 and 21 show an example of how the configuration manager 5544 allows the user to control reconfiguration for non-critical applications. The applications currently running in the multiprocessor system 8 are displayed in the GUI 118 in block 55150. A failure is detected for the navigation application 55110 running in processor A in block 55152. The configuration manager 5544 in processor A, or in one of the other processors B, C, or D detects the navigation failure. Alternatively, a fusion processor 55111 is coupled to some or all of the processors A, B, C and D and detects the navigation failure.

In block 55154 the configuration manager 5544 for one of the processors determines if there is extra capacity in one of the other processors for running the failed navigation application 55110. If there is another processor with extra processing capacity, the navigation application is downloaded from memory 55128 to that processor with extra capacity along with any necessary navigation data in block 55156. This reconfiguration may be done automatically without any interaction with the car operator.

If there is no extra processing capacity for running the navigation application 55110, the configuration manager 5544 displays the failed processor or application to the user in block 55158. For example, the GUI 118 in FIG. 20 starts blinking the

navigation icon 55120 in possibly a different color than the audio application icon 55124. A textual failure message 55125 can also be displayed on GUI 55118.

The configuration manager in block 55160 waits for the car operator to request reconfiguration of the failed navigation application to another processor. If there is no user request, the configuration managers return to monitoring for other failures. If the user requests reconfiguration, the configuration manager 5544 in block 55164 displays other non-critical applications to the user. For example, the GUI 55118 only displays the audio application icon 55124 in processor B and not the ABS application icon 55122 (FIG. 18). This is because the audio application is a non-critical application and the ABS application 55114 is a critical application that cannot be cancelled.

If the car operator selects the audio icon 55124 in block 55166, the configuration manager in block 55168 cancels the audio application 55112 in processor B and downloads the navigation application 55110 from memory 55128 into processor B. A logging manager in processor A may have labeled certain navigation data for logging. That navigation data 55130 may include the last few minutes of position data for the car while the navigation application 55110 was running in processor A. The logged navigation data 55130 is downloaded from memory 55128 along with the navigation application 55110 into processor B. The navigation icon 55120 in GUI 55118 then shows the navigation application 55110 running on processor B. At the same time the audio application icon 55124 is removed from GUI 55118.

Referring back to FIG. 13, a processor or application is accepted into the multiprocessor system by one or more of the device managers 5546. The configuration managers 5544 in the processors reconfigure the multiprocessor system

to incorporate the processor or application. The data manager 5542 then detects what type of data is transmitted or received by the new device and determines the different processors and input/output devices in the multiprocessor system that can receive or transmit data to the new application or processor.

FIG. 22 shows in further detail how the data manager 5542 in FIG. 13 operates. In block 55170, the data manager for one of the processors determines the data standard for the data that is either transmitted or received by a new device. For example, the new device may be a MP3 player that outputs streaming audio data. In another example, the new device may be a DVD player that outputs streaming video data in a MPEG format.

One or more of the data managers 5542, identifies the device by its data and the data, if applicable, is displayed on the graphical user interface in block 55172. The data manager then identifies any devices in the multiprocessor system that can output or transmit data to the new device in block 55174. For example, a newly detected audio source may be output from a car speaker. The data manager monitors for any user selections in block 55176. For example, the car operator may select the output from a portable CD player to be output from the car speakers. The data manager controlling the CD player and the data manager controlling the car speakers then direct the output from the CD player to the car speakers in block 55178.

FIG. 23 gives one example of how the data managers 5542 in the multiprocessing system operate. A GUI 55180 displays the audio or video (A/V) sources in a car. For example, there are three devices detected in or around the car that are A/V sources. A cellular telephone detected in the car is represented by icon 55184, a radio is represented by icon 55186, and a DVD player is represented by icon 55188.

The A/V output devices in the car are shown in the lower portion of GUI 55180. For example, icons 55192, 55194, 55196, 55200, and 55204 show car audio speakers. An in-dash video display is represented by icon 55190 and a portable monitor is represented by icon 55198.

Currently, a car operator may be listening to the radio 55186 over speakers 55192, 55194, 55196, 55200 and 55204. However, a passenger may move into the backseat of the car carrying an MP3 player. The MP3 player runs the DC system 5510 described in FIG. 13 and sends out a signal to any other processors in the multiprocessor system 8 in the car. The device manager 5546 and configuration manager 5544 in one of the processors verify the data format for the MP3 player and configure the MP3 player into the multiprocessor system.

One of the data managers 5542 determines the MP3 player outputs a MP3 audio stream and accordingly generates the icon 55182 on the GUI 55180. The data manager 5542 also identifies a speaker in the MP3 player as a new output source and displays the speaker as icon 55202. The car operator sees the MP3 icon 55182 now displayed on GUI 55180. The car operator can move the MP3 icon 55182 over any combination of the speaker icons 55192, 55194, 55196, 55200 and 55204. The output from the MP3 player is then connected to the selected audio outputs.

Audio data can also be moved in the opposite direction. The speaker icon 55202 represents the output of the portable MP3 player that the passenger brought into the backseat of the car. The car operator also has the option of moving one or more of the other audio sources, such as the cellular telephone 55184 or the radio 55186 icons over the speaker icon 55202. If the car operator, for example, moves the radio icon 55186 over the MP3 player speaker icon 55202 and the MP3 player can

output the radio signals, the multiprocessor system redirects the radio broadcast out over the MP3 speaker.

It should be understood that the multiprocessor system described above could be used in applications other than cars. For example, FIG. 24 shows a first GUI 55210 that shows different processors and applications that are coupled together using the DC system 5510 in an automobile. A GUI 55212 shows another multiprocessor system comprising multiple processors in the home. For example, a washing machine is shown by icon 55214. The DC system allows the washing machine processor to communicate and be configured with a television processor 55216, toaster processor 55218, stereo processor 55220, and an oven processor 55222.

The system described above can use dedicated processor systems, micro controllers, programmable logic devices, or microprocessors that perform some or all of the communication operations. Some of the operations described above may be implemented in software and other operations may be implemented in hardware.

For the sake of convenience, the operations are described as various interconnected functional blocks or distinct software modules. This is not necessary, however, and there may be cases where these functional blocks or modules are equivalently aggregated into a single logic device, program or operation with unclear boundaries. In any event, the functional blocks and software modules or described features can be implemented by themselves, or in combination with other operations in either hardware or software.

Having described and illustrated the principles of the invention in a preferred embodiment thereof, it should be apparent that the invention may be modified in arrangement and detail without departing from such principles. Claim is made to all

modifications and variation coming within the spirit and scope of the following claims.

CLAIMS

1. A method for communicating between multiple processors in a mobile vehicle, comprising:

receiving messages from different processors in the mobile vehicle;

passing the messages through multiple communication managers to identify different message labels; and

using the message labels for performing different real-time mobile vehicle applications in the different processors.

2. A method according to claim 1 wherein the communication management layers include a security manager that prevents messages without a security label from being processed by certain applications or certain processors in the mobile vehicle.

3. A method according to claim 1 wherein one of the communication managers includes a logging manager that logs data according to a logging label.

4. A method according to claim 3 wherein a first processor adds the logging label to the message causing the first processor and a second processor receiving the message to both log the message.

5. A method according to claim 1 wherein the communication managers include a priority manager that submits the messages to different applications according to an identified priority label included with the message.

6. A method according to claim 5 wherein the priority manager in each processor identifies a priority value in the priority label, ranks the message with other messages in the same processor according to the priority value and submits the message to one or more applications in the processor according to the ranking.

7. A method according to claim 6 wherein one of the messages is a critical collision message that is assigned a high priority value that is processed before other mobile vehicle messages.

8. A method according to claim 1 including receiving the messages from different processors in the mobile device over different communication links.

9. A method according to claim 1 wherein the different communication links include a wireless communication link and a serial communication link.

10. A method according to claim 1 including:

receiving image sensor data from an image processor;

processing the sensor data when a security label is identified in the sensor data;

logging the image sensor data when a logging label is identified in the image sensor data; and

prioritizing the processing of the image sensor data according to a priority label in the image sensor data.

11. A method according to claim 10 including:

sending a break message to a break control processor when an object in the image data is identified as being within a predetermined range of the mobile vehical;

attaching a logging label to the break command; and

assigning a high priority value to a priority label in the break command.

12. A method according to claim 10 including sending an annunciation message to speakers in the mobile vechical when an object in the image data is identified as being within a predetermined range of the mobile vehical.

13. A communication system for a mobile vechical, comprising:

multiple processors;

multiple links connecting the multiple processors together; and

the multiple processors each operating an open communication system that controls how data or messages are received and processed in a real-time environment by the processors.

14. A communication system according to claim 13 including a communication manager in the open communication system that interfaces the data or messages with different applications.

15. A communication system according to claim 13 including a security manager that prevents data or messages from unauthorized applications from sending data to the multiple processors.

16. A communication system according to claim 13 including a logging manager that controls when data and messages from the different processors are logged.

17. A communication system according to claim 13 including a priority manager that determines a priority for processing the data and messages in the processors.

18. A communication system according to claim 13 wherein one of the processors is a sensor processor, one of the processors is a fusion processor and another one of the processors is a break control processor.

19. A communications system according to claim 18 wherein the fusion processor receives image data from the sensor processor and sends breaking commands to the break control processor according to the image data.

20. A communications system according to claim 19 wherein the fusion processor and the image processor log the image data or messages according to how close objects in the image data are from the mobile vehicle.

21. A communications system according to claim 20 including a second sensor processor that sends image data to the fusion processor.

22. A communications system according to claim 13 wherein a display processor displays sensor data from a sensor processor according to a security label, logging label and priority label attached to the sensor data by the sensor processor.

23. A communications system according to claim 13 wherein some of the multiple links use different communication protocols for transferring messages and data between the different processors.

24. A communications system according to claim 23 wherein one of the links is a wireless link and another one of the links is a hardwired serial link.

25. A multiprocessing system for an automobile, comprising:

multiple processors that run different automobile applications;

multiple links that couple the multiple processors together; and

a dynamic configuration system operating in at least some of the multiple processors that automatically incorporates new devices into the multiprocessing system and automatically reconfigures the multiprocessor system in real-time to run the automobile applications on different processors in the multiprocessing system.

26. A multiprocessing system according to claim 25 wherein the dynamic configuration system includes a device manager that detects signals generated by new devices and incorporates the new devices into the multiprocessor system when the signals conform a protocol used between the multiple processors.

27. A multiprocessing system according to claim 25 wherein the dynamic configuration system includes a configuration manager that tracks the different applications operating in the different processors and automatically reconfigures the multiprocessing system to run failed applications on different ones of the multiple processors.

28. A multiprocessing system according to claim 27 including storing a copy of the application that has failed and downloading and running the stored copy of the application to another processor when the failure is detected.

29. A multiprocessing system according to claim 28 including storing critical data generated by the failed application and downloading and running the stored critical data along with the copy of the application on another processor.

30. A multiprocessing system according to claim 27 including displaying applications that have failed and then displaying applications in the other processors that can be replaced with copies of the failed applications.

31. An automobile processing system according to claim 30 including identifying types of data transferred by different devices in the multiprocessing system and displaying the different devices in the multiprocessing system that can output the identified types of data.

32. An automobile processing system according to claim 30 including performing the following applications with the multiprocessor system:

automatic brake control;

audio player control;

video player control;

airbag deployment monitoring;

display control;

navigation control; and

sensor monitoring.

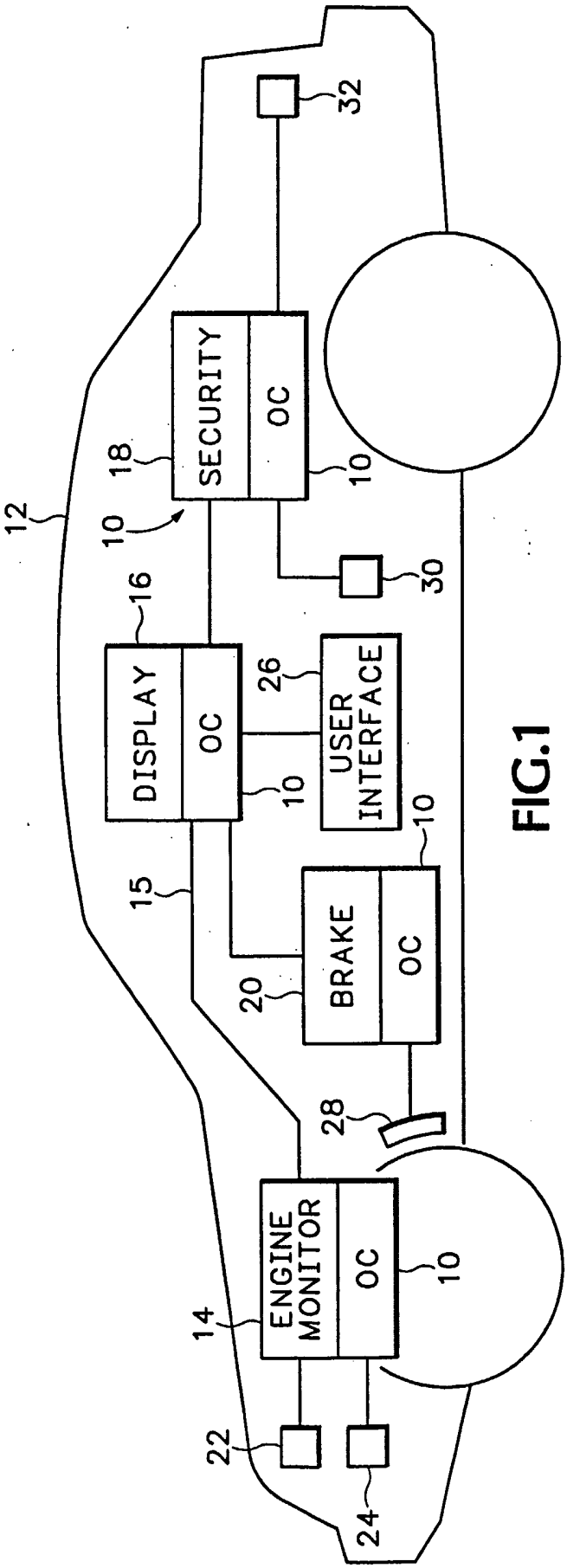

32
SECURITY
OC
18
10
12
10
30
16
DISPLAY
OC
26
USER INTERFACE
10
10
15
BRAKE
OC
10
20
28
ENGINE MONITOR
OC
10
14
22
24

FIG.1

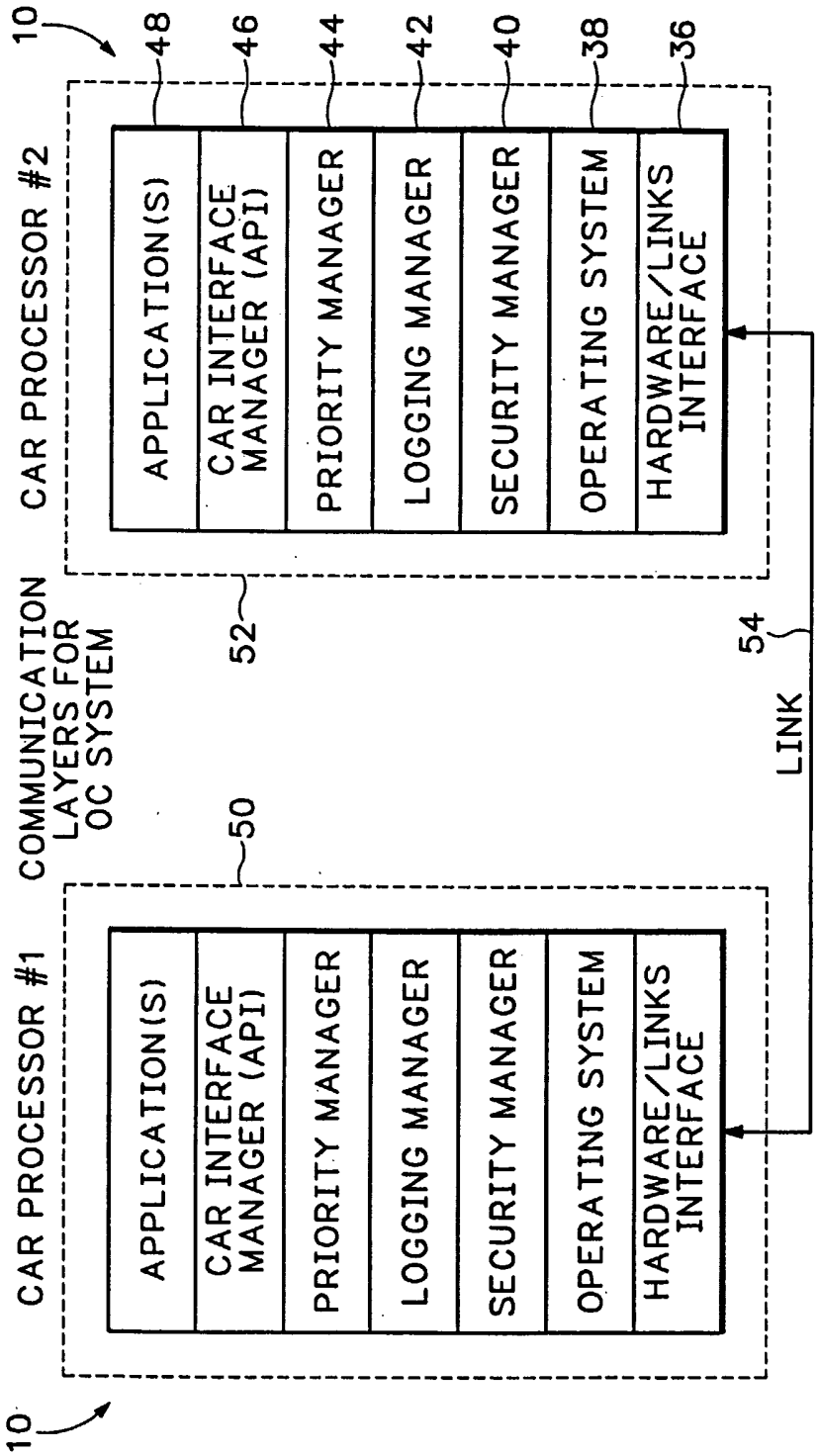

10
CAR PROCESSOR #2
APPLICATION(S)
CAR INTERFACE MANAGER (API)
PRIORITY MANAGER
LOGGING MANAGER
SECURITY MANAGER
OPERATING SYSTEM
HARDWARE/LINKS INTERFACE
48
46
44
42
40
38
36
52
54
LINK
COMMUNICATION LAYERS FOR OC SYSTEM
50
CAR PROCESSOR #1
APPLICATION(S)
CAR INTERFACE MANAGER (API)
PRIORITY MANAGER
LOGGING MANAGER
SECURITY MANAGER
OPERATING SYSTEM
HARDWARE/LINKS INTERFACE
10

FIG.2

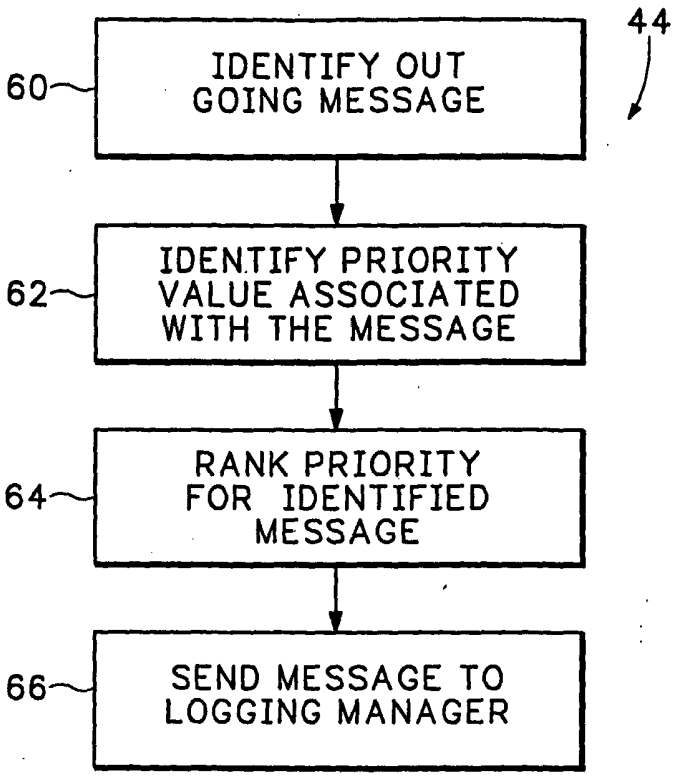

44
60
IDENTIFY OUT GOING MESSAGE
62
IDENTIFY PRIORITY VALUE ASSOCIATED WITH THE MESSAGE
64
RANK PRIORITY FOR IDENTIFIED MESSAGE
66
SEND MESSAGE TO LOGGING MANAGER

FIG.3

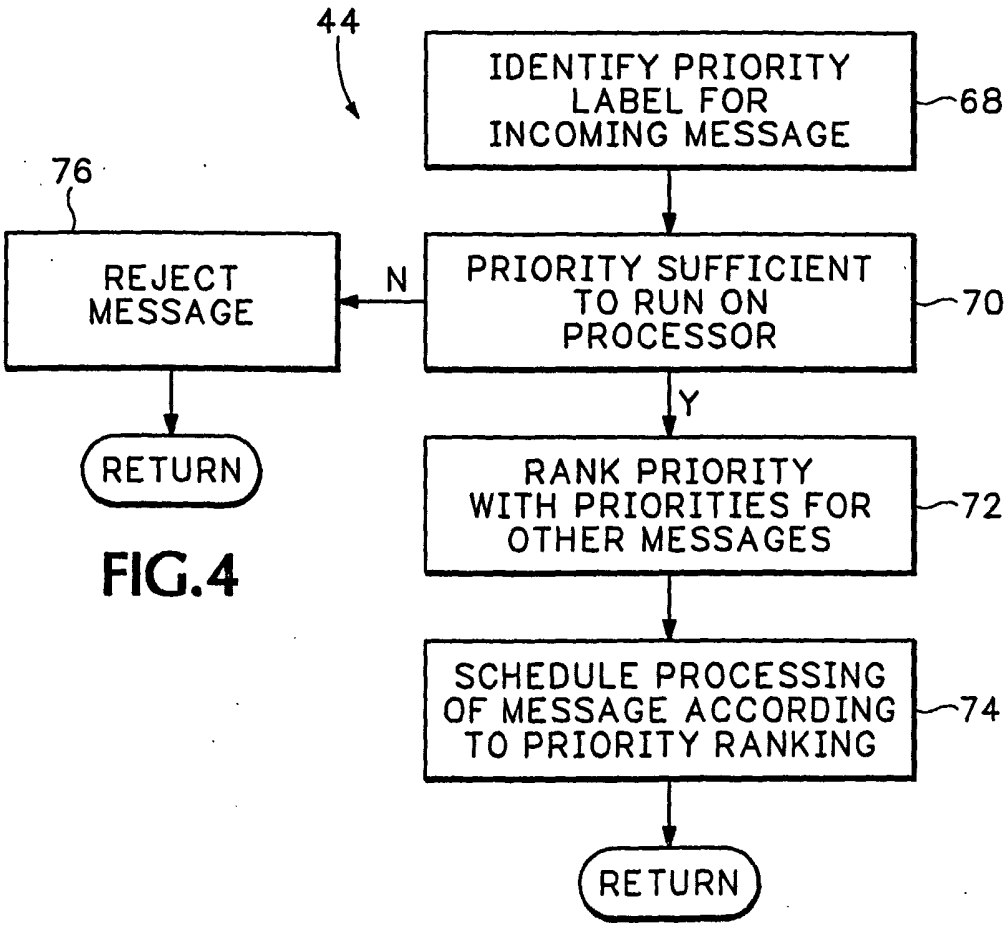

44
IDENTIFY PRIORITY LABEL FOR INCOMING MESSAGE
68
PRIORITY SUFFICIENT TO RUN ON PROCESSOR
70
N
Y
76
REJECT MESSAGE
RETURN
RANK PRIORITY WITH PRIORITIES FOR OTHER MESSAGES
72
SCHEDULE PROCESSING OF MESSAGE ACCORDING TO PRIORITY RANKING
74
RETURN

FIG.4

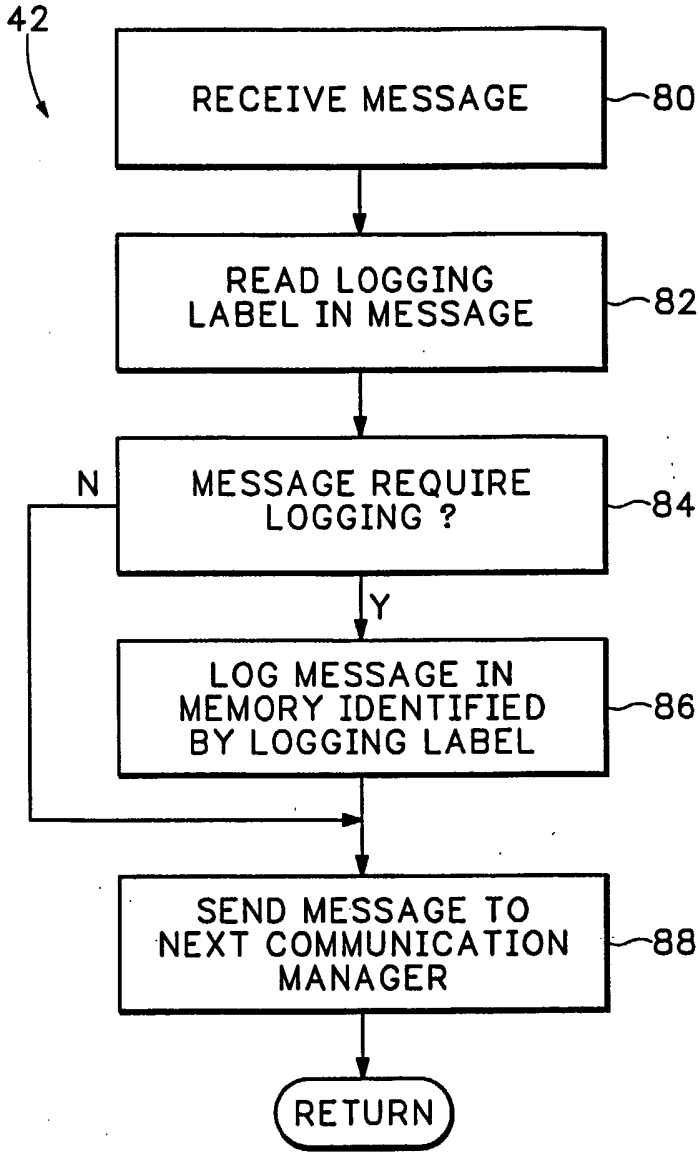

42
RECEIVE MESSAGE
80
READ LOGGING LABEL IN MESSAGE
82
MESSAGE REQUIRE LOGGING ?
84
N
Y
LOG MESSAGE IN MEMORY IDENTIFIED BY LOGGING LABEL
86
SEND MESSAGE TO NEXT COMMUNICATION MANAGER
88
RETURN

FIG.5

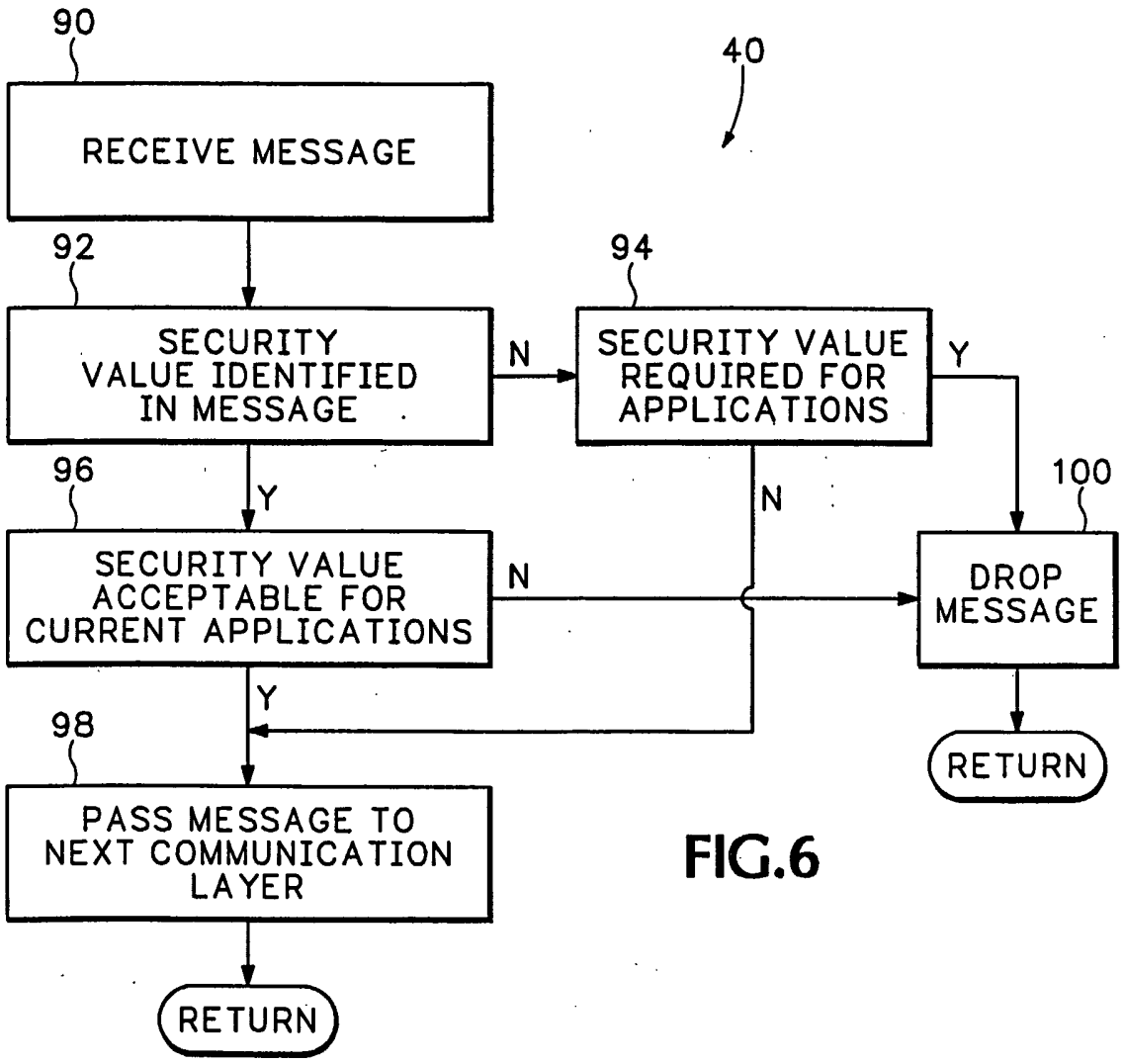

90
40
RECEIVE MESSAGE
92
94
SECURITY VALUE IDENTIFIED IN MESSAGE
N
SECURITY VALUE REQUIRED FOR APPLICATIONS
Y
96
Y
N
100
SECURITY VALUE ACCEPTABLE FOR CURRENT APPLICATIONS
N
DROP MESSAGE
Y
98
RETURN
PASS MESSAGE TO NEXT COMMUNICATION LAYER
RETURN

FIG.6

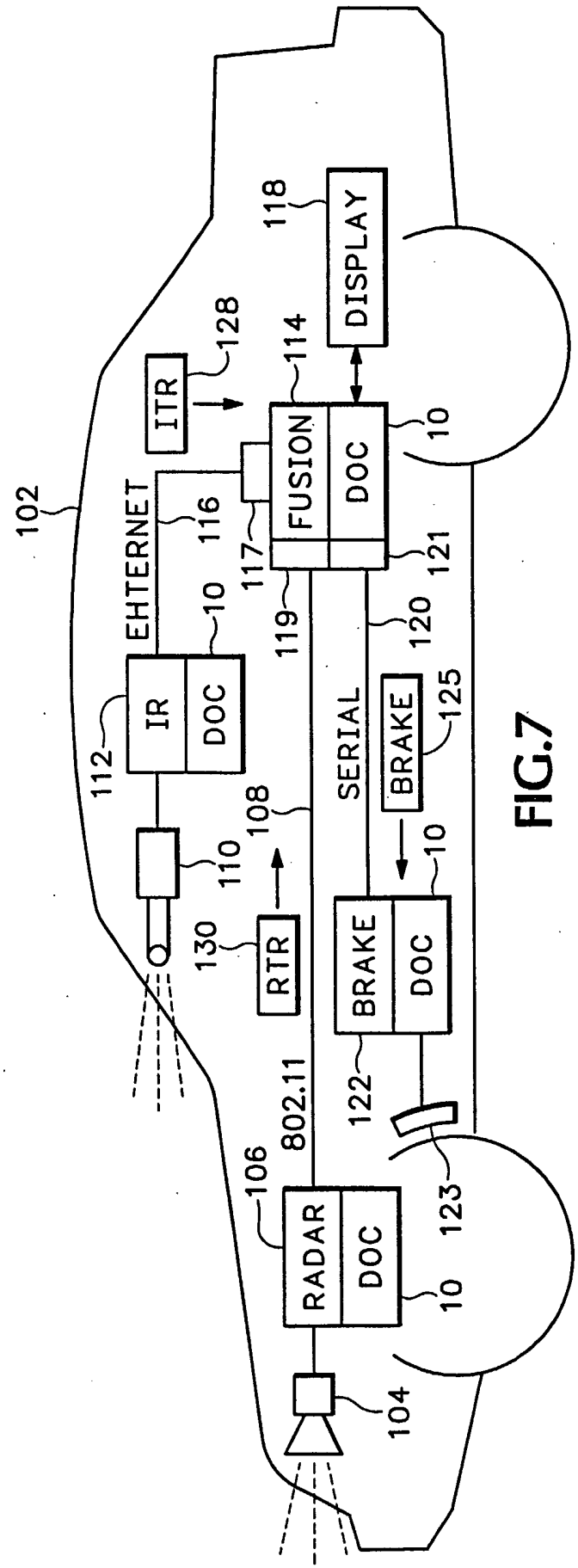

102
ITR
128
DISPLAY
118
FUSION
DOC
114
10
EHTERNET
116
117
121
119
120
IR
DOC
112
10
SERIAL
BRAKE
125
108
110
10
130
RTR
BRAKE
DOC
802.11
122
106
123
RADAR
DOC
10
104

FIG.7

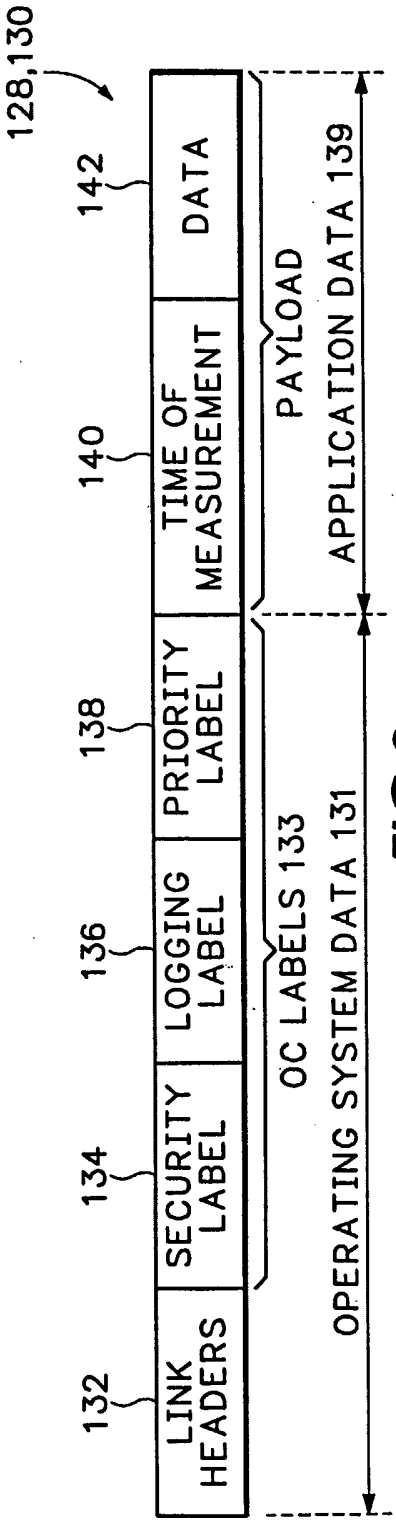

128,130
142
DATA
140
TIME OF MEASUREMENT
PAYLOAD
APPLICATION DATA 139
138
PRIORITY LABEL
136
LOGGING LABEL
OC LABELS 133
134
SECURITY LABEL
132
LINK HEADERS
OPERATING SYSTEM DATA 131

FIG.8

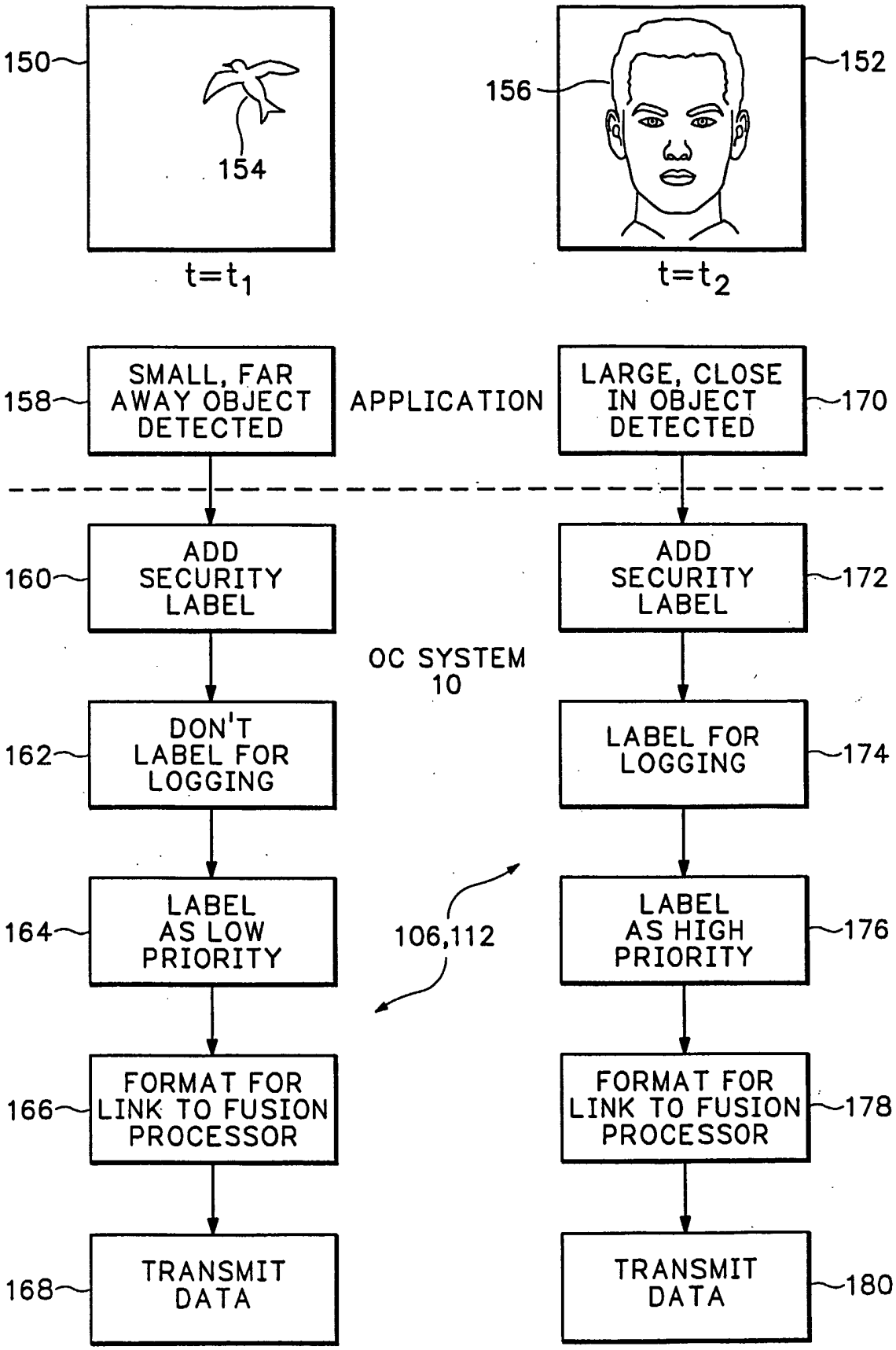

150
154
$t=t_1$
156
152
$t=t_2$
158
SMALL, FAR AWAY OBJECT DETECTED
APPLICATION
LARGE, CLOSE IN OBJECT DETECTED
170
160
ADD SECURITY LABEL
ADD SECURITY LABEL
172
OC SYSTEM 10
162
DON'T LABEL FOR LOGGING
LABEL FOR LOGGING
174
164
LABEL AS LOW PRIORITY
106,112
LABEL AS HIGH PRIORITY
176
166
FORMAT FOR LINK TO FUSION PROCESSOR
FORMAT FOR LINK TO FUSION PROCESSOR
178
168
TRANSMIT DATA
TRANSMIT DATA
180

FIG.9

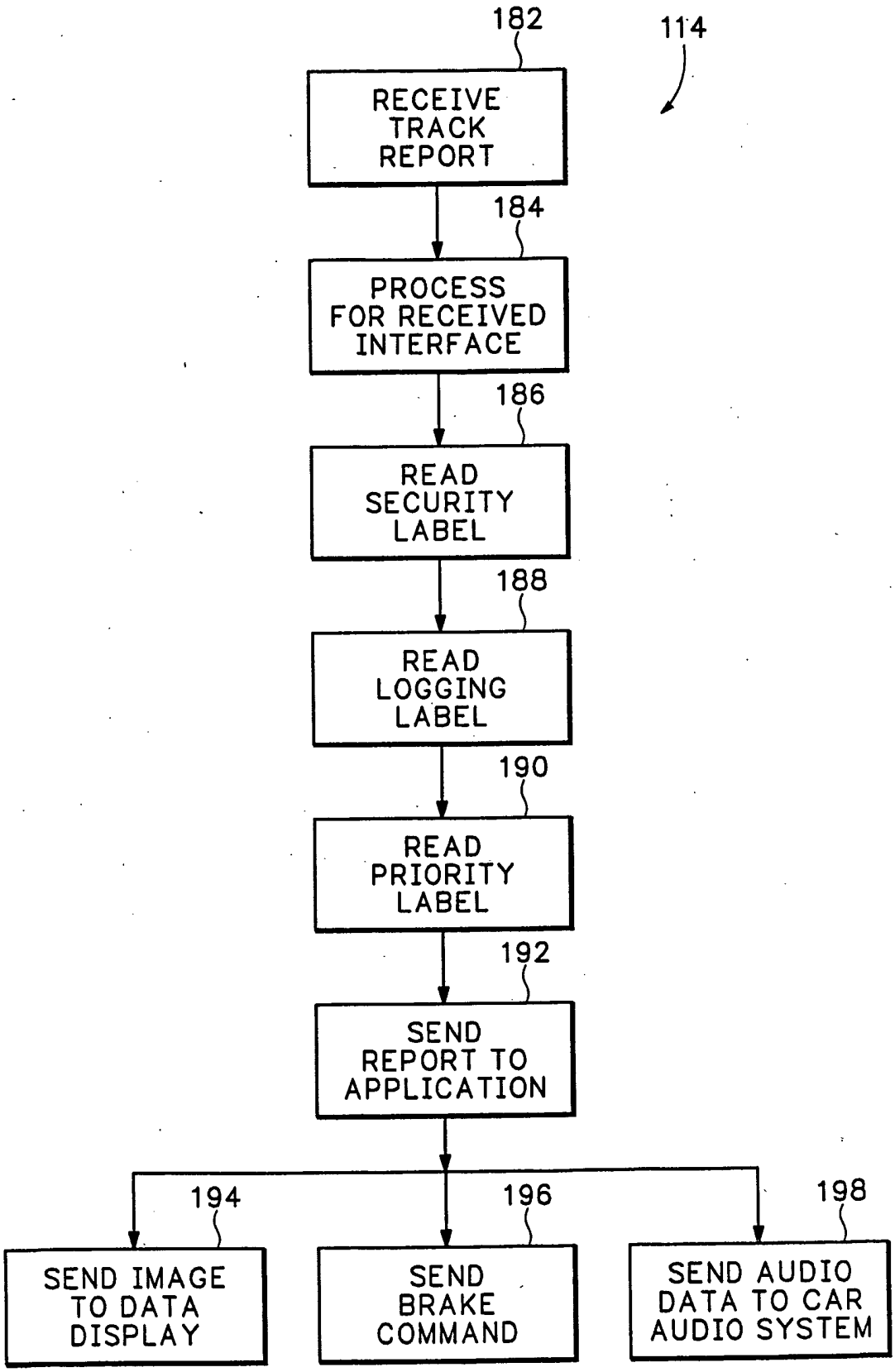

182
114
RECEIVE TRACK REPORT
184
PROCESS FOR RECEIVED INTERFACE
186
READ SECURITY LABEL
188
READ LOGGING LABEL
190
READ PRIORITY LABEL
192
SEND REPORT TO APPLICATION
194
196
198
SEND IMAGE TO DATA DISPLAY
SEND BRAKE COMMAND
SEND AUDIO DATA TO CAR AUDIO SYSTEM

FIG.10

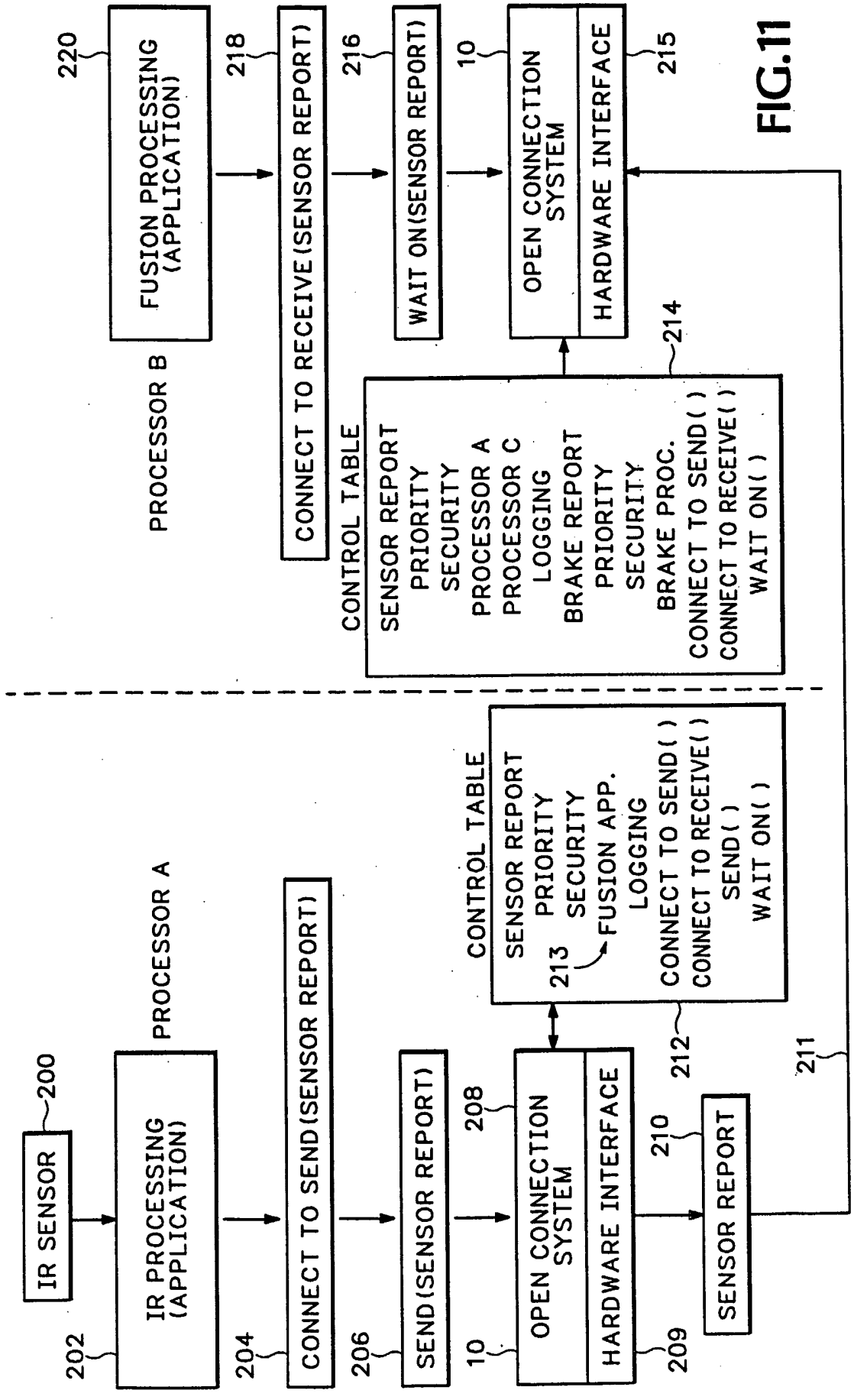

IR SENSOR
200
IR PROCESSING (APPLICATION)
PROCESSOR A
202
CONNECT TO SEND(SENSOR REPORT)
204
SEND(SENSOR REPORT)
206
10
OPEN CONNECTION SYSTEM
208
HARDWARE INTERFACE
210
209
SENSOR REPORT
212
211
CONTROL TABLE
SENSOR REPORT
PRIORITY
SECURITY
213
FUSION APP.
LOGGING
CONNECT TO SEND( )
CONNECT TO RECEIVE( )
SEND( )
WAIT ON( )
FUSION PROCESSING (APPLICATION)
220
PROCESSOR B
CONNECT TO RECEIVE(SENSOR REPORT)
218
WAIT ON(SENSOR REPORT)
216
10
OPEN CONNECTION SYSTEM
HARDWARE INTERFACE
215
CONTROL TABLE
SENSOR REPORT
PRIORITY
SECURITY
PROCESSOR A
PROCESSOR C
LOGGING
BRAKE REPORT
PRIORITY
SECURITY
BRAKE PROC.
CONNECT TO SEND( )
CONNECT TO RECEIVE( )
WAIT ON( )
214

FIG.11

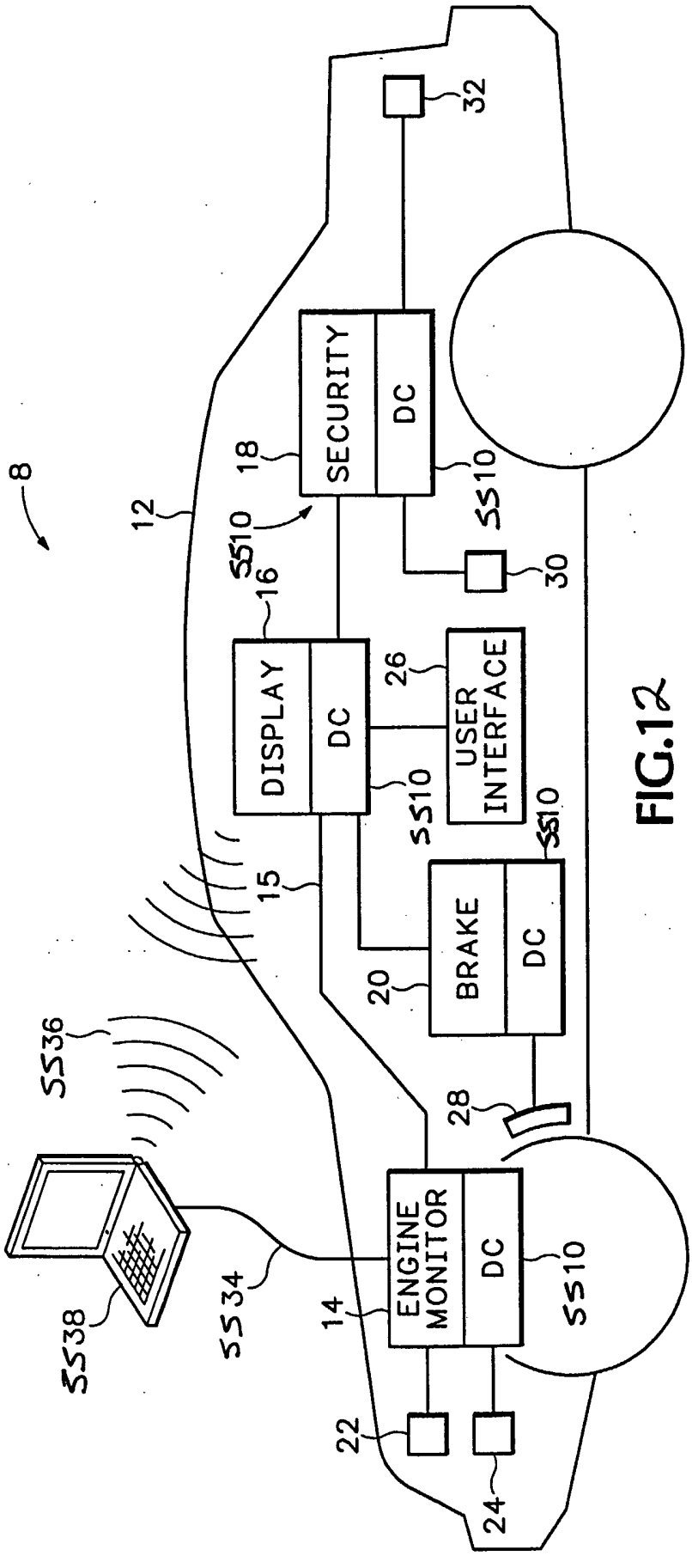

8
12
32
SECURITY
DC
18
S510
30
16
DISPLAY
DC
26
USER INTERFACE
S510
15
BRAKE
DC
20
S510
S536
28
ENGINE MONITOR
DC
S510
S534
14
S538
22
24

FIG.12

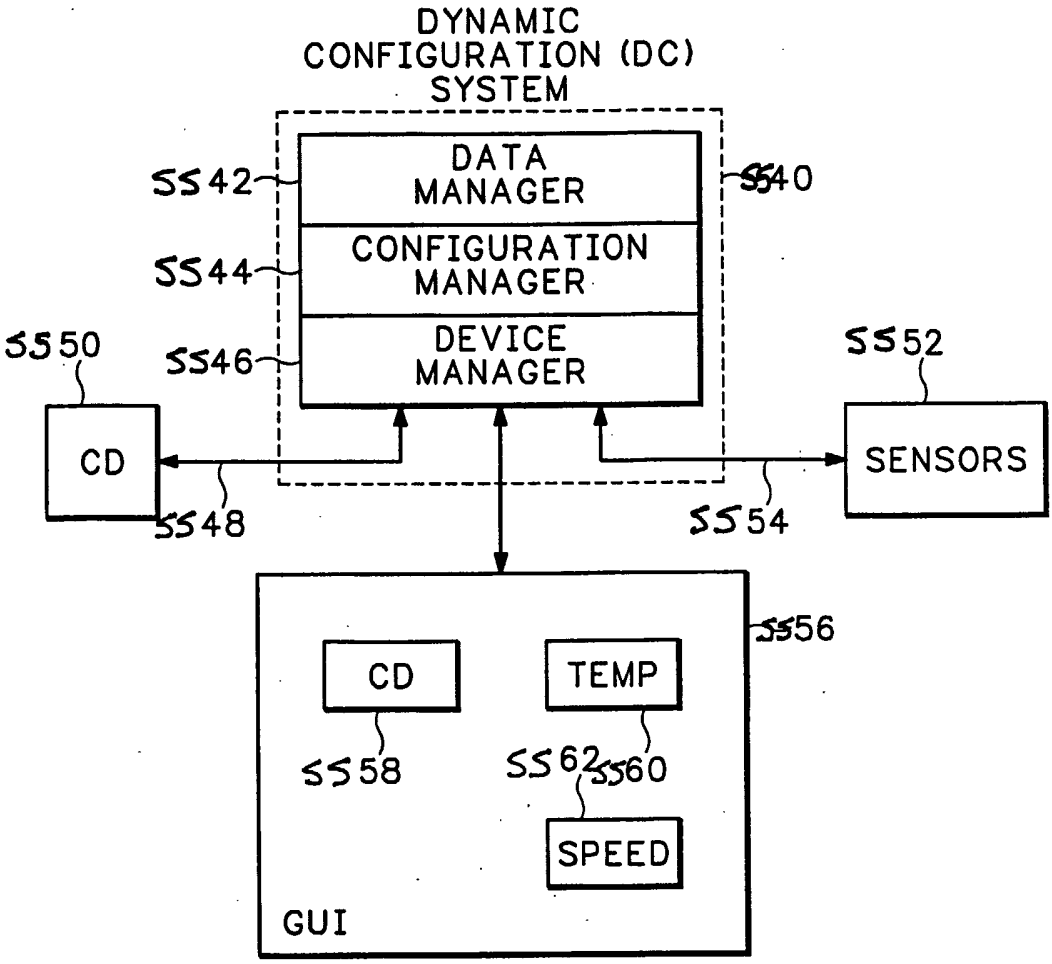

DYNAMIC CONFIGURATION (DC) SYSTEM
DATA MANAGER
CONFIGURATION MANAGER
DEVICE MANAGER
5542
5544
5546
5540
5550
5552
CD
SENSORS
5548
5554
5556
CD
TEMP
SPEED
5558
5562
5560
GUI

FIG. 13

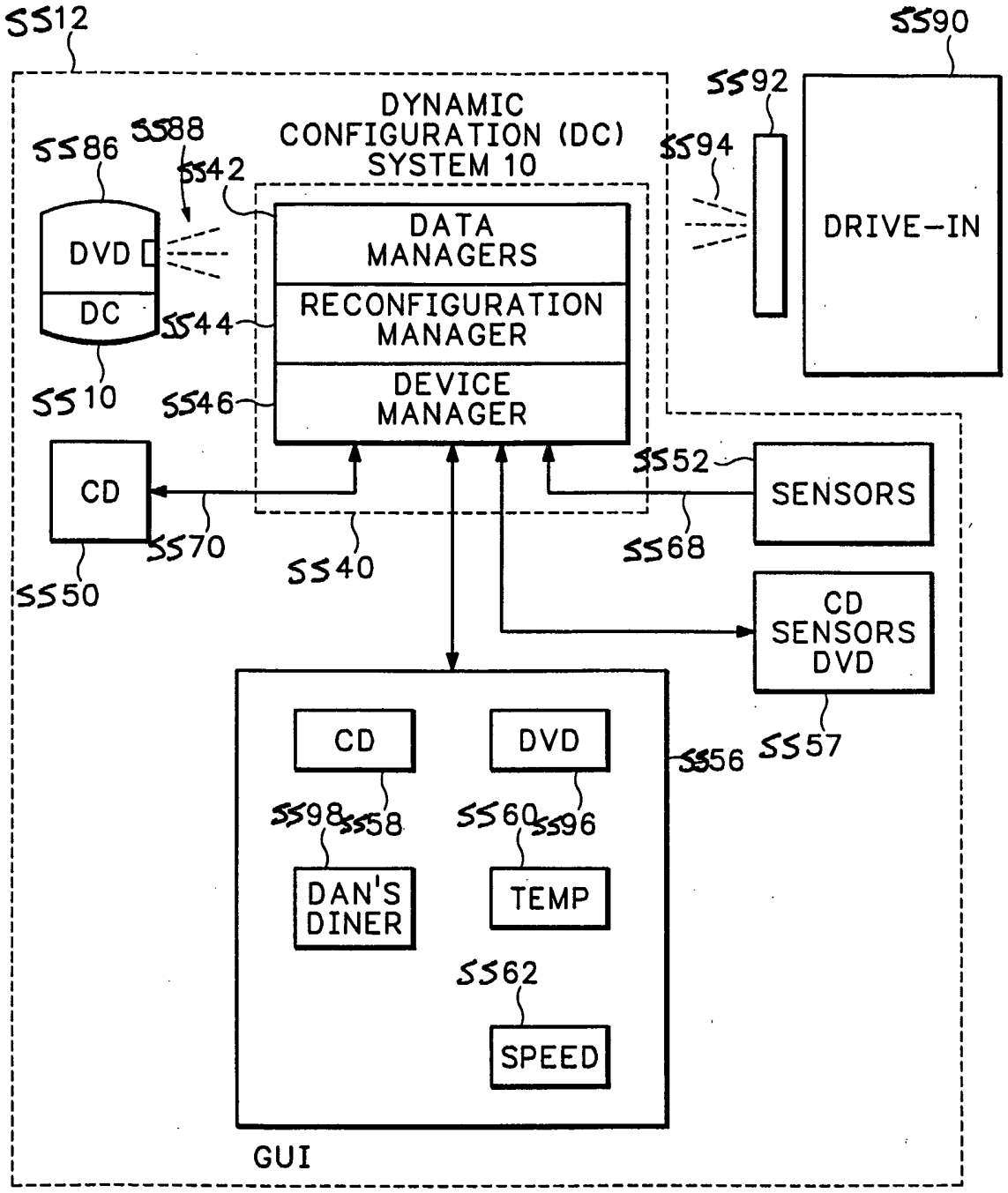

5512
5590
5592
DYNAMIC CONFIGURATION (DC) SYSTEM 10
5586
5588
5542
5594
DRIVE-IN
DVD
DC
DATA MANAGERS
RECONFIGURATION MANAGER
DEVICE MANAGER
5544
5546
5510
CD
5552
SENSORS
5570
5540
5568
5550
CD SENSORS DVD
5557
CD
DVD
5556
5598
5558
5560
5596
DAN'S DINER
TEMP
5562
SPEED
GUI

FIG. 14

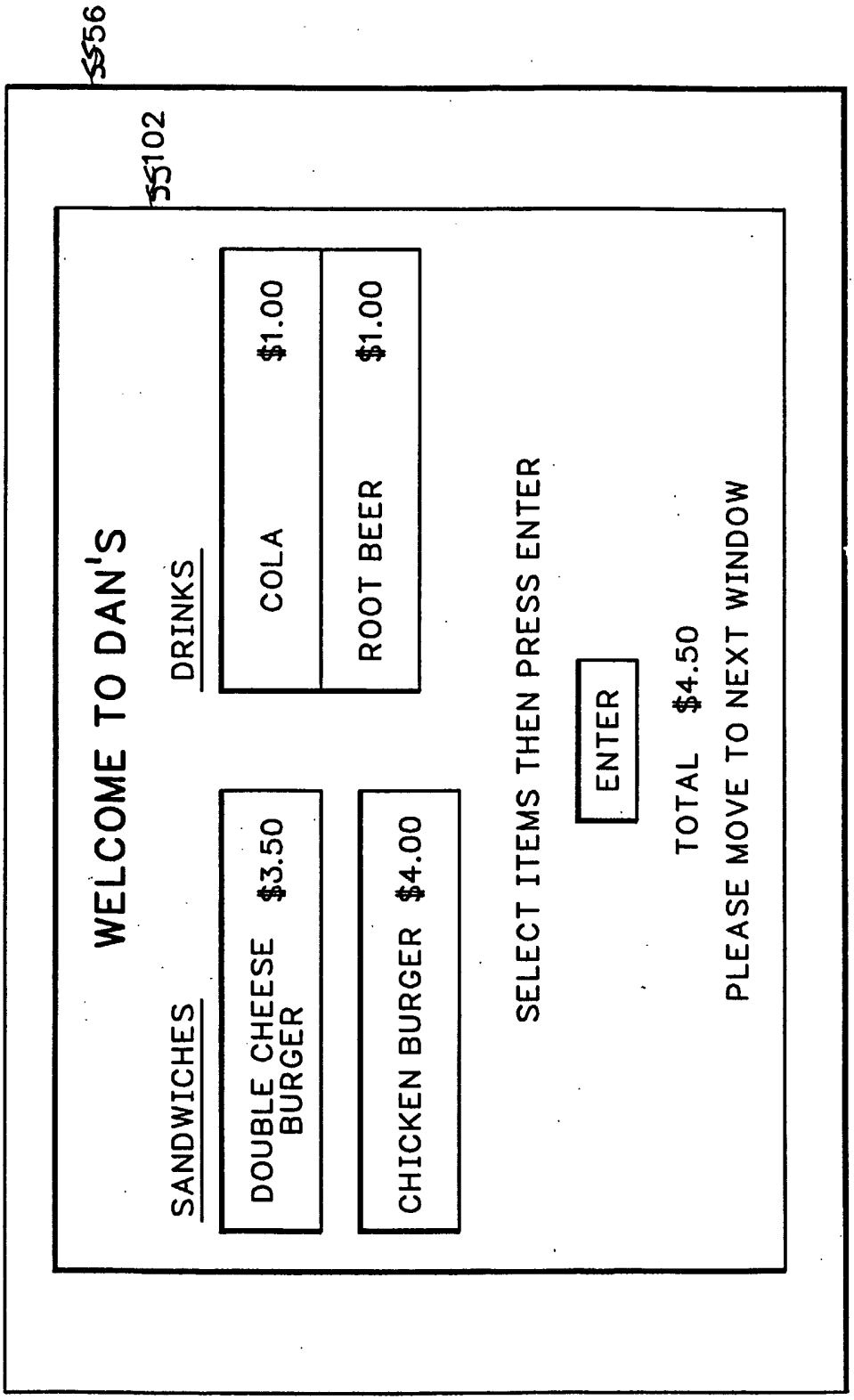

356
102
WELCOME TO DAN'S
SANDWICHES
DOUBLE CHEESE BURGER $3.50
CHICKEN BURGER $4.00
DRINKS
COLA $1.00
ROOT BEER $1.00
SELECT ITEMS THEN PRESS ENTER
ENTER
TOTAL $4.50
PLEASE MOVE TO NEXT WINDOW

FIG. 15

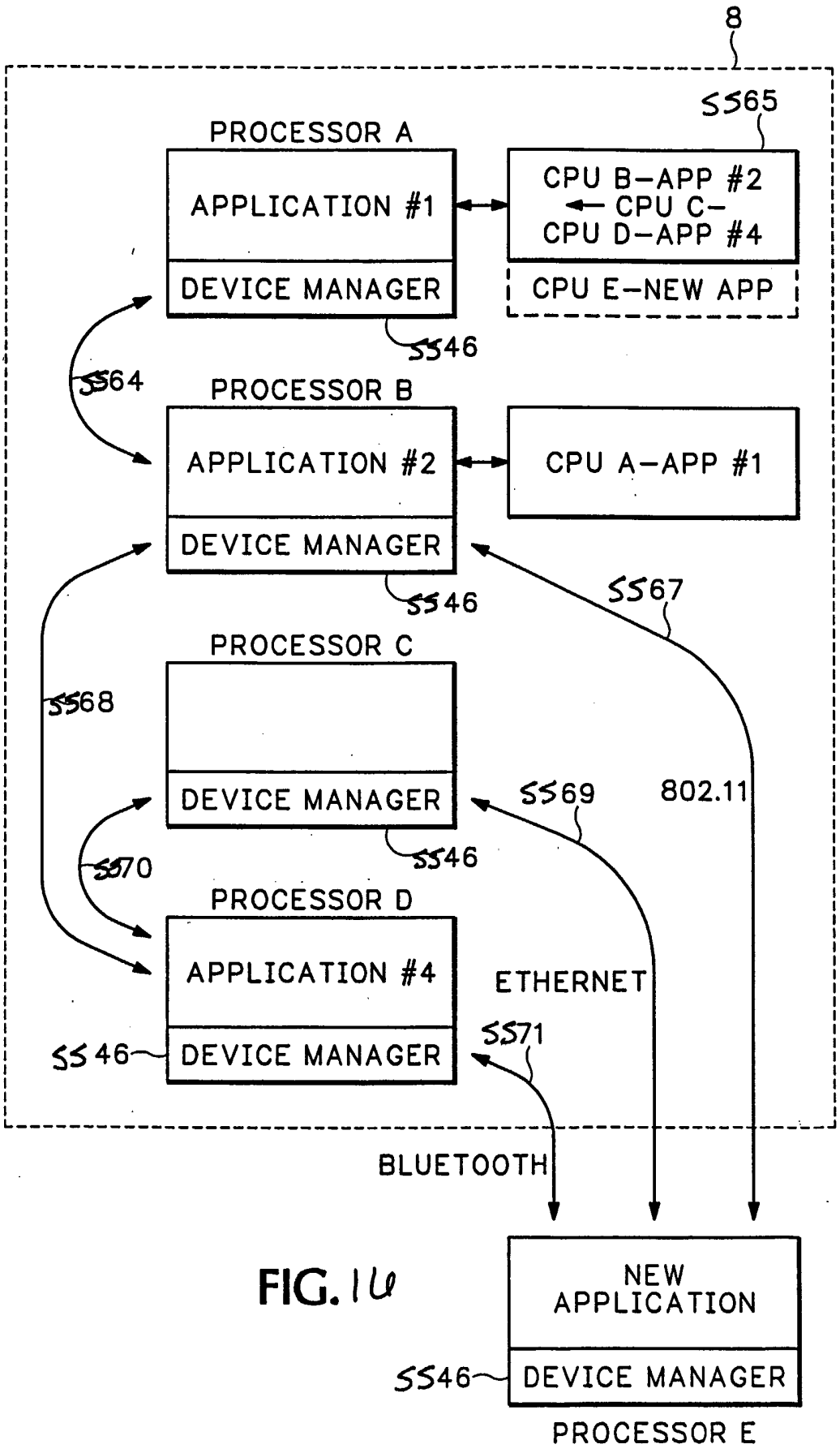

8
S565
PROCESSOR A
APPLICATION #1
DEVICE MANAGER
CPU B-APP #2
CPU C-
CPU D-APP #4
CPU E-NEW APP
S546
S564
PROCESSOR B
APPLICATION #2
DEVICE MANAGER
CPU A-APP #1
S546
S567
PROCESSOR C
S568
DEVICE MANAGER
S569
802.11
S546
S570
PROCESSOR D
APPLICATION #4
ETHERNET
S546
DEVICE MANAGER
S571
BLUETOOTH
NEW
APPLICATION
S546
DEVICE MANAGER
PROCESSOR E

FIG. 14

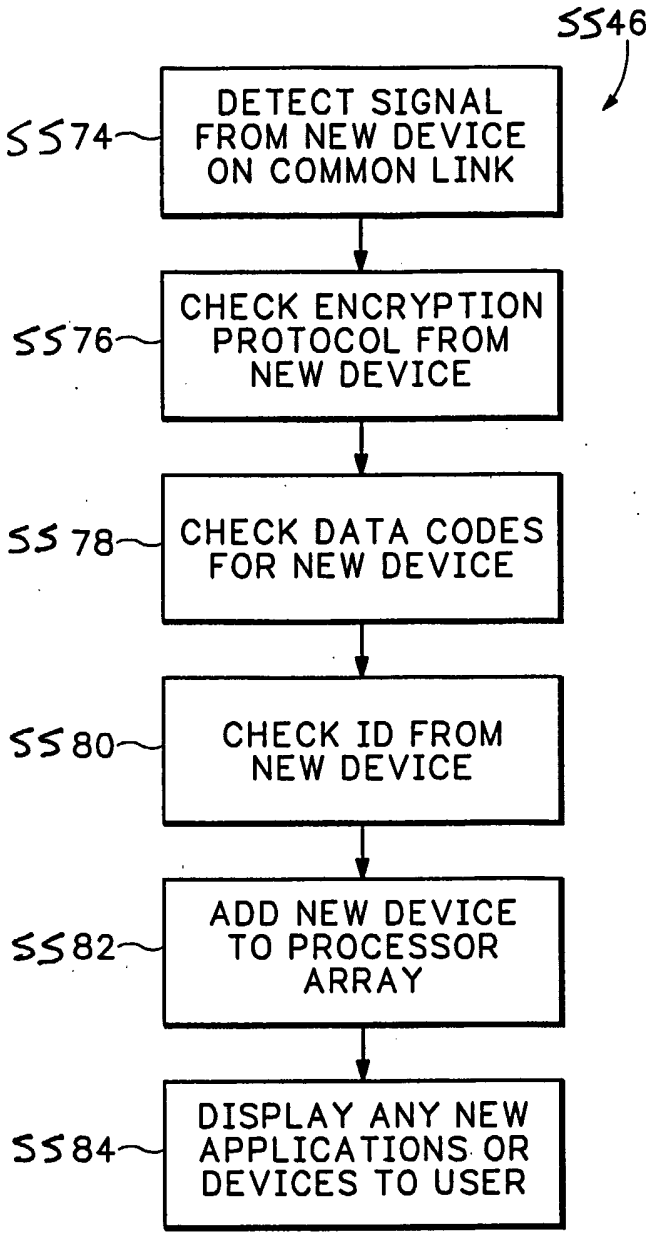

5546
5574
DETECT SIGNAL FROM NEW DEVICE ON COMMON LINK
5576
CHECK ENCRYPTION PROTOCOL FROM NEW DEVICE
5578
CHECK DATA CODES FOR NEW DEVICE
5580
CHECK ID FROM NEW DEVICE
5582
ADD NEW DEVICE TO PROCESSOR ARRAY
5584
DISPLAY ANY NEW APPLICATIONS OR DEVICES TO USER

FIG. 17

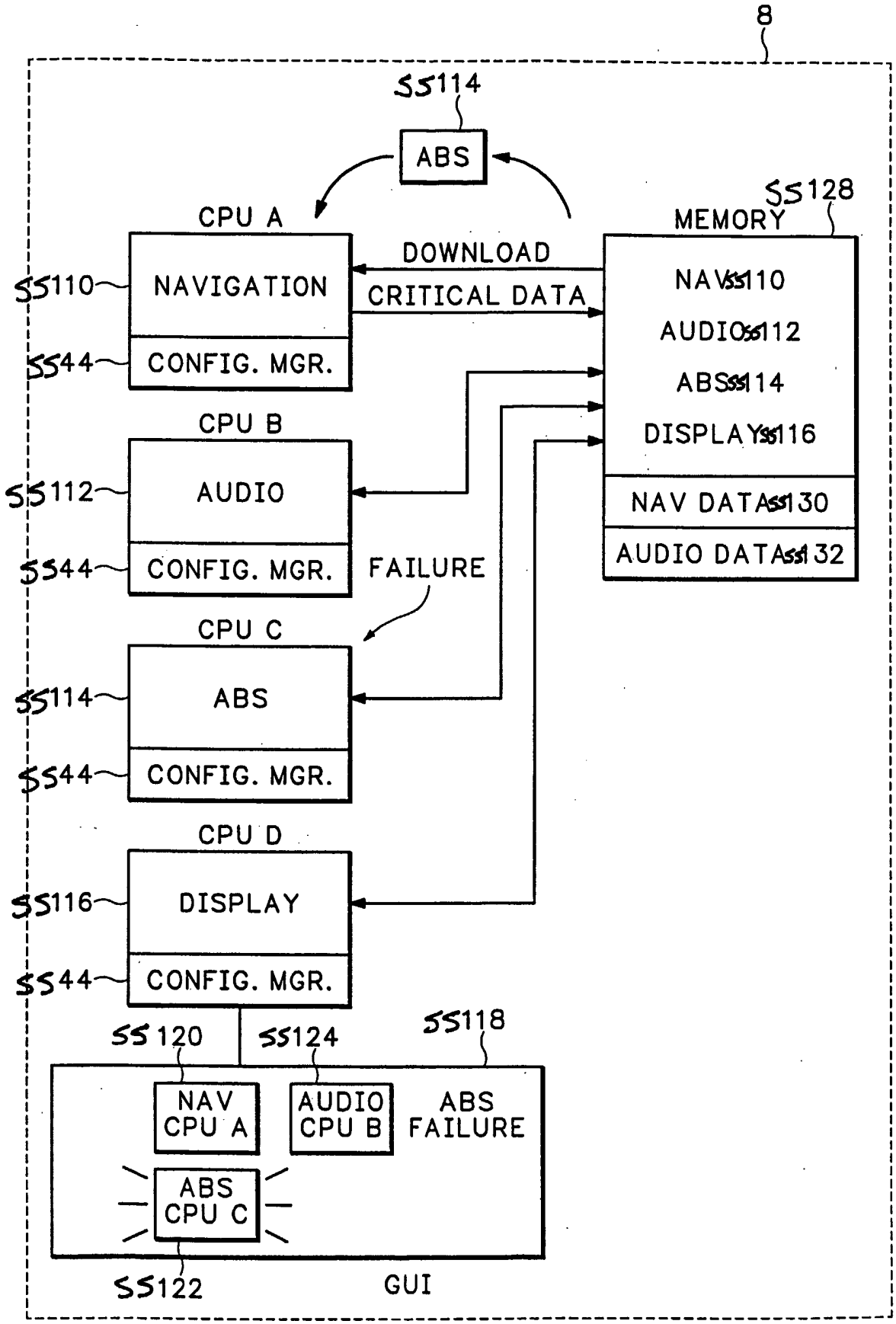

8
SS114
ABS
CPU A
SS128
MEMORY
DOWNLOAD
CRITICAL DATA
SS110
NAVIGATION
SS44
CONFIG. MGR.
NAV SS110
AUDIO SS112
ABS SS114
DISPLAY SS116
CPU B
SS112
AUDIO
SS44
CONFIG. MGR.
FAILURE
NAV DATA SS130
AUDIO DATA SS132
CPU C
SS114
ABS
SS44
CONFIG. MGR.
CPU D
SS116
DISPLAY
SS44
CONFIG. MGR.
SS120
SS124
SS118
NAV CPU A
AUDIO CPU B
ABS FAILURE
ABS CPU C
SS122
GUI

FIG. 18

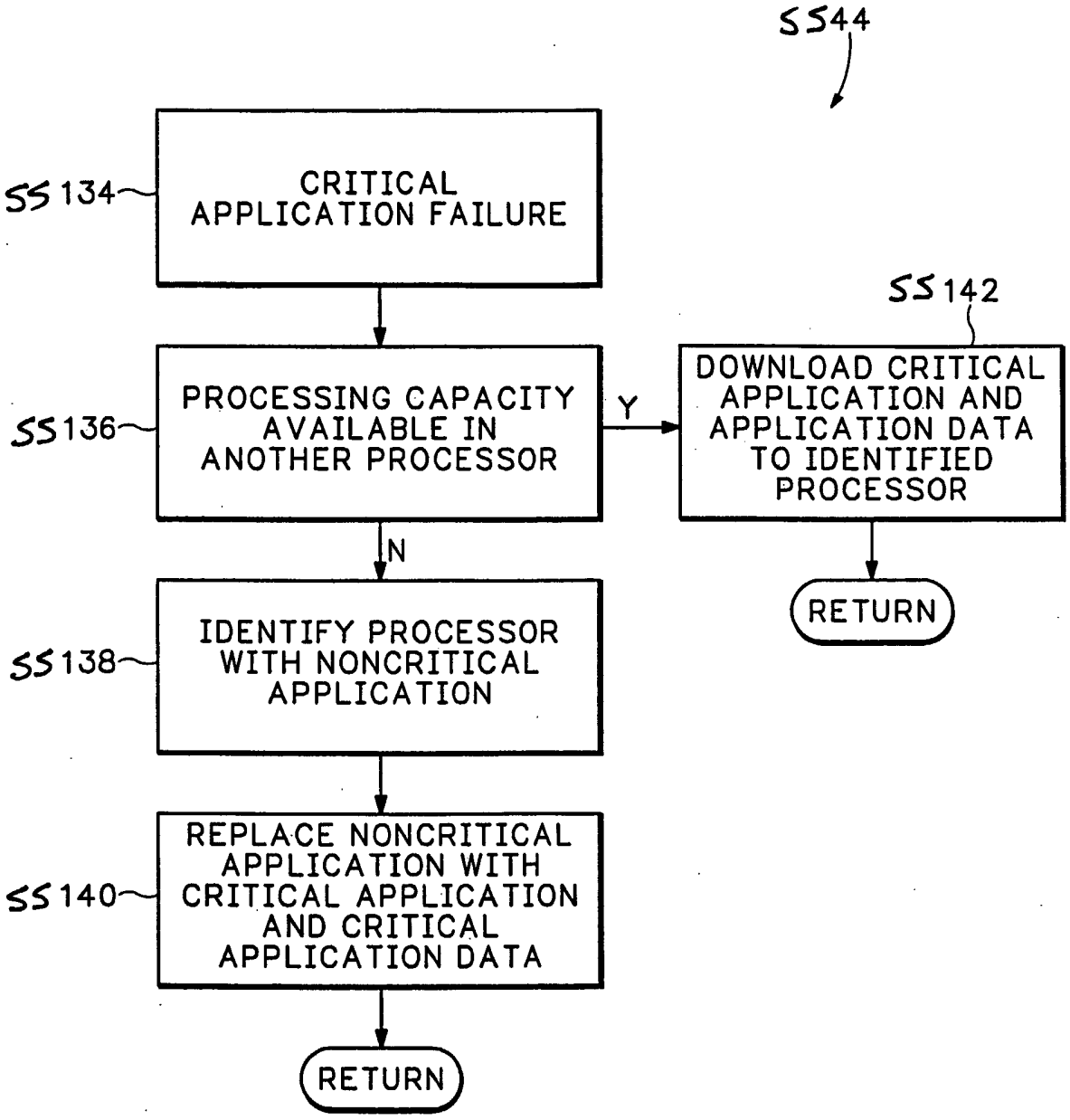

SS44
SS134
CRITICAL APPLICATION FAILURE
SS142
SS136
PROCESSING CAPACITY AVAILABLE IN ANOTHER PROCESSOR
Y
DOWNLOAD CRITICAL APPLICATION AND APPLICATION DATA TO IDENTIFIED PROCESSOR
N
RETURN
SS138
IDENTIFY PROCESSOR WITH NONCRITICAL APPLICATION
SS140
REPLACE NONCRITICAL APPLICATION WITH CRITICAL APPLICATION AND CRITICAL APPLICATION DATA
RETURN

FIG. 19

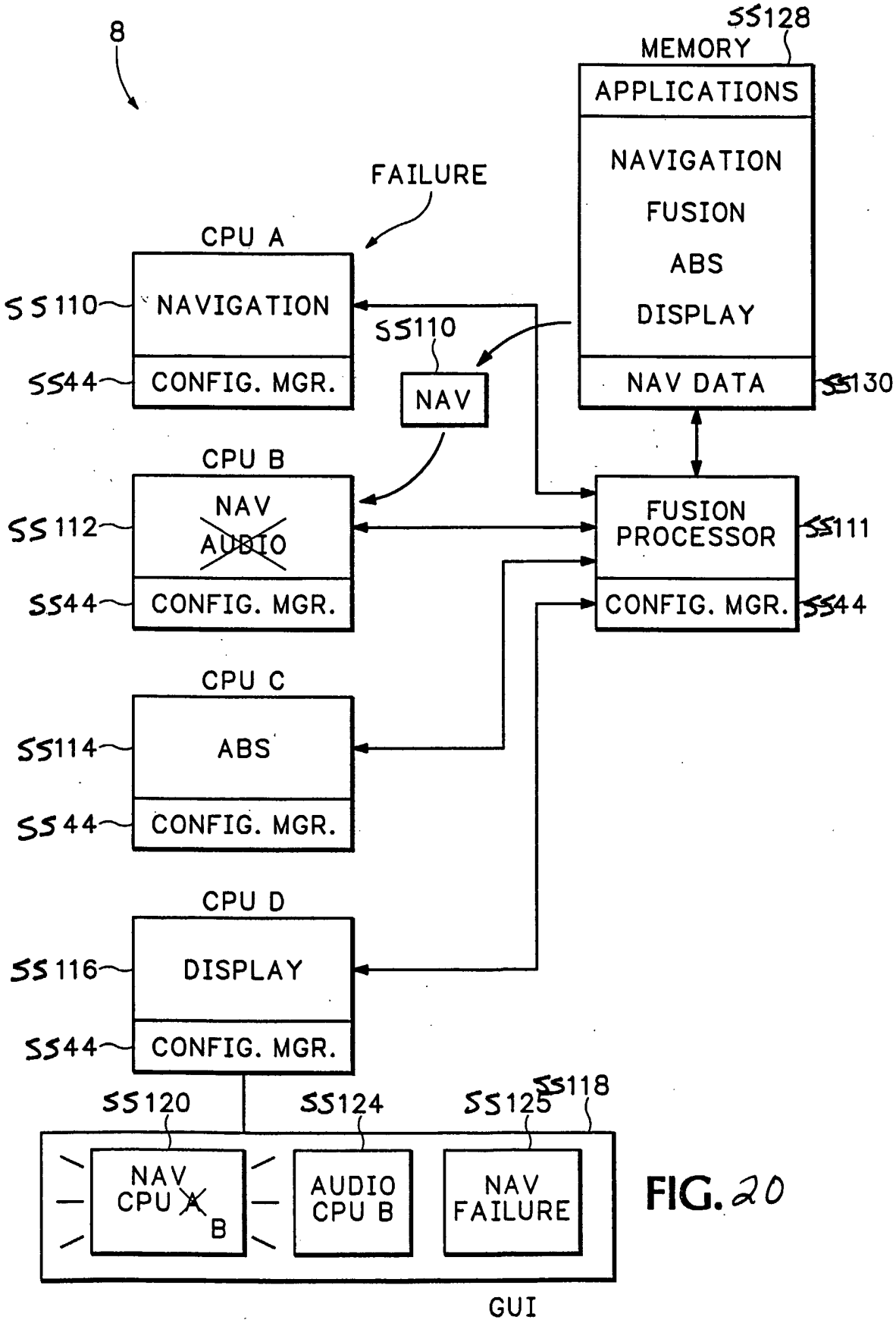

8
MEMORY
APPLICATIONS
NAVIGATION
FUSION
ABS
DISPLAY
NAV DATA
FAILURE
CPU A
NAVIGATION
CONFIG. MGR.
NAV
CPU B
NAV
AUDIO
CONFIG. MGR.
FUSION PROCESSOR
CONFIG. MGR.
CPU C
ABS
CONFIG. MGR.
CPU D
DISPLAY
CONFIG. MGR.
NAV CPU A B
AUDIO CPU B
NAV FAILURE
GUI

FIG. 20

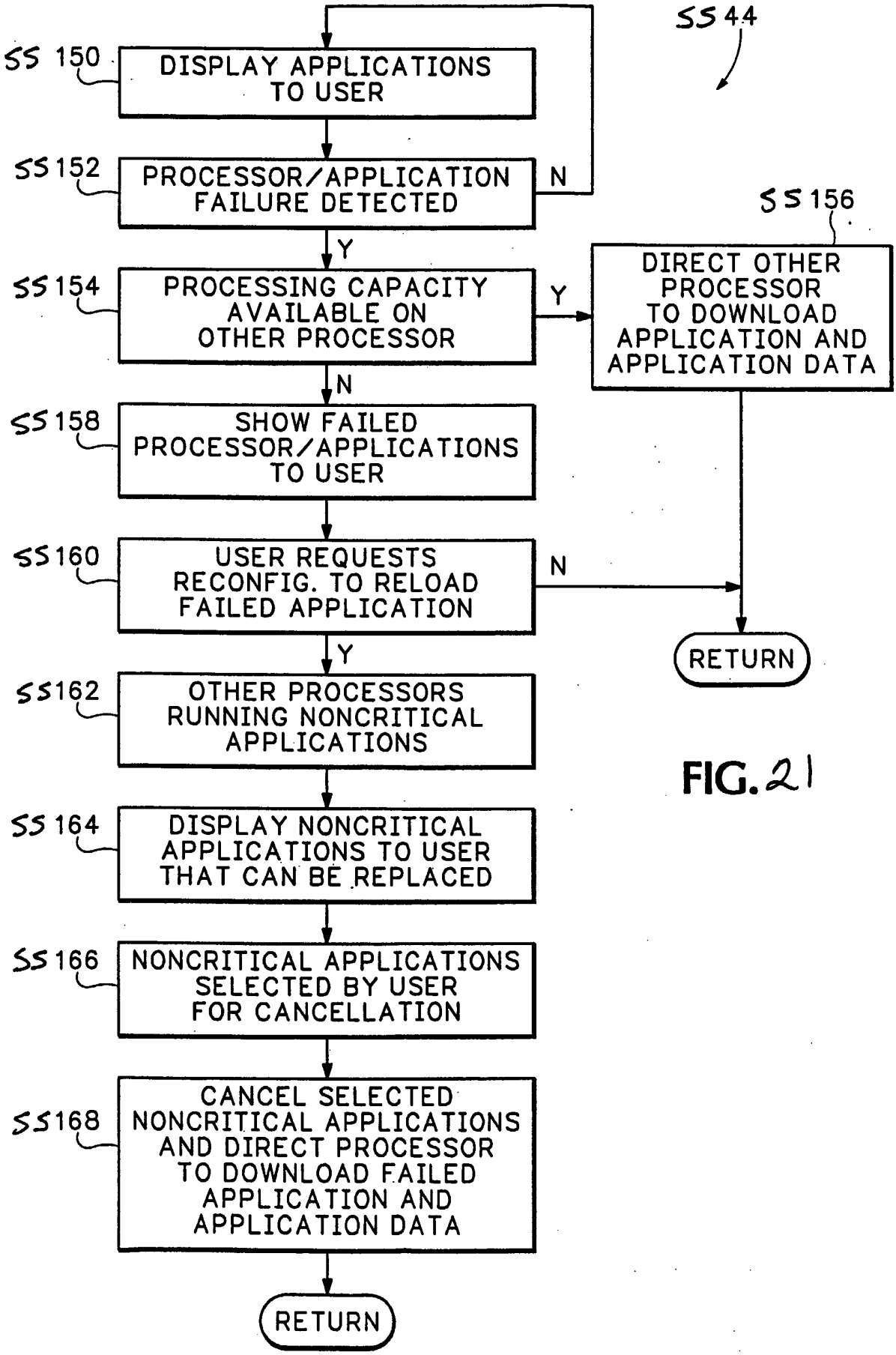

SS 44
SS 150
DISPLAY APPLICATIONS TO USER
SS 152
PROCESSOR/APPLICATION FAILURE DETECTED
N
Y
SS 154
PROCESSING CAPACITY AVAILABLE ON OTHER PROCESSOR
Y
SS 156
DIRECT OTHER PROCESSOR TO DOWNLOAD APPLICATION AND APPLICATION DATA
N
SS 158
SHOW FAILED PROCESSOR/APPLICATIONS TO USER
SS 160
USER REQUESTS RECONFIG. TO RELOAD FAILED APPLICATION
N
RETURN
Y
SS 162
OTHER PROCESSORS RUNNING NONCRITICAL APPLICATIONS
SS 164
DISPLAY NONCRITICAL APPLICATIONS TO USER THAT CAN BE REPLACED
SS 166
NONCRITICAL APPLICATIONS SELECTED BY USER FOR CANCELLATION
SS 168
CANCEL SELECTED NONCRITICAL APPLICATIONS AND DIRECT PROCESSOR TO DOWNLOAD FAILED APPLICATION AND APPLICATION DATA
RETURN

FIG. 21

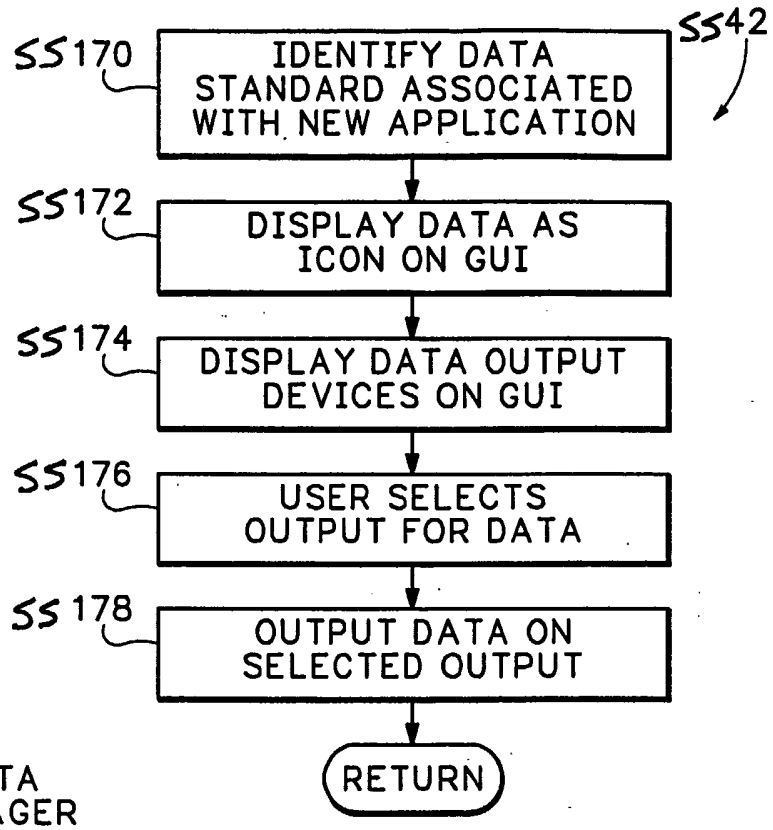

5542
5170
IDENTIFY DATA STANDARD ASSOCIATED WITH NEW APPLICATION
5172
DISPLAY DATA AS ICON ON GUI
5174
DISPLAY DATA OUTPUT DEVICES ON GUI
5176
USER SELECTS OUTPUT FOR DATA
5178
OUTPUT DATA ON SELECTED OUTPUT
RETURN

FIG. 22

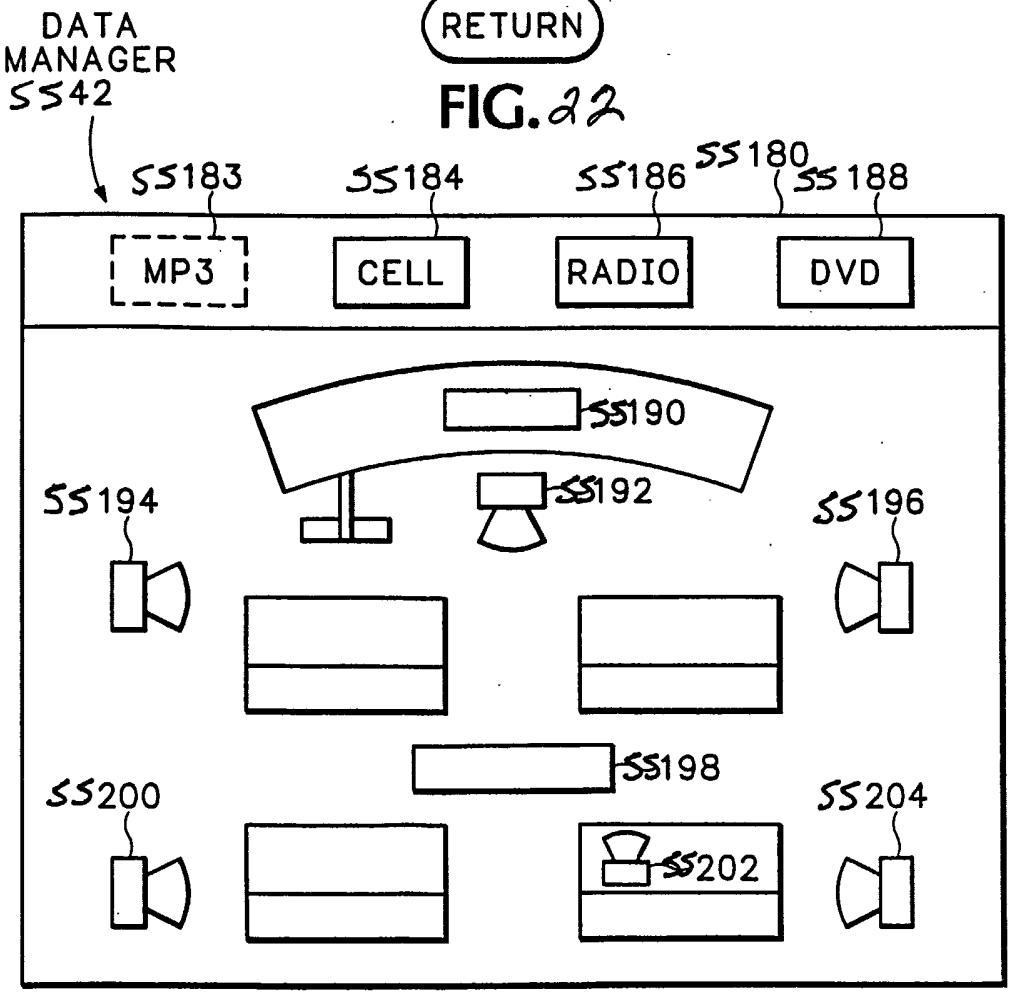

DATA MANAGER
5542
5183
5184
5186
5180
5188
MP3
CELL
RADIO
DVD
5190
5192
5194
5196
5198
5200
5202
5204

FIG. 23

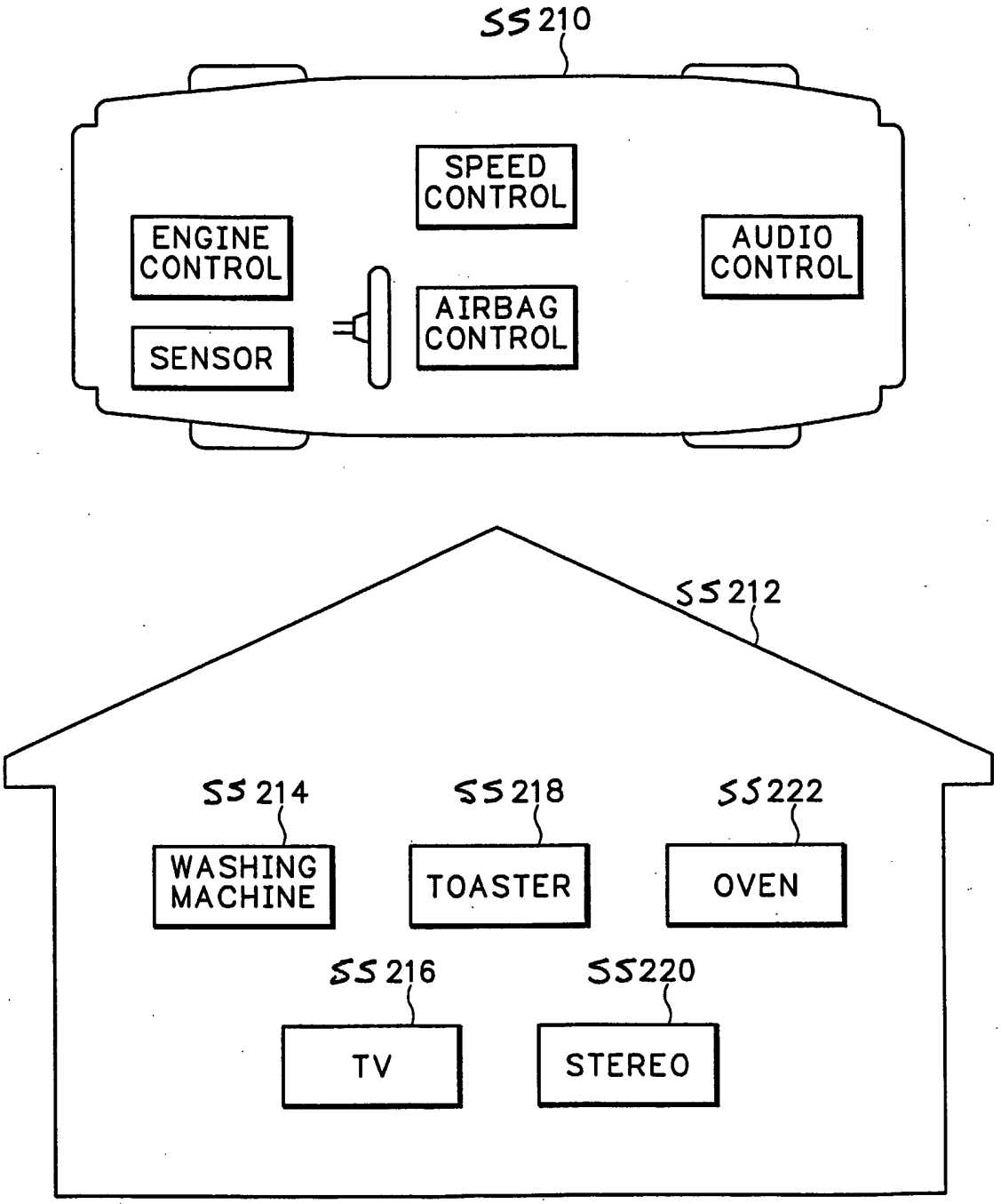

210
SPEED CONTROL
ENGINE CONTROL
AUDIO CONTROL
AIRBAG CONTROL
SENSOR
212
214
218
222
WASHING MACHINE
TOASTER
OVEN
216
220
TV
STEREO

FIG. 24