

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

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
14 August 2003 (14.08.2003)

PCT

(10) International Publication Number
WO 03/067444 A2

(51) International Patent Classification⁷: **G06F 13/00**

(21) International Application Number: PCT/US03/00884

(22) International Filing Date: 10 January 2003 (10.01.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/060,336 1 February 2002 (01.02.2002) US

(71) Applicant: **INTEL CORPORATION** [US/US]; 2200
Mission College Boulevard, Santa Clara, CA 95052 (US).

(72) Inventors: **BURTON, Robert**; 2120 East Victor Road,
Gilbert, AZ 85266 (US). **JOSHI, Aniruddha**; 781 North
Granada Drive, Chandler, AZ 85226 (US). **WANG, Jen-
nifer**; 9533 South Kenneth Place, Tempe, AZ 85284 (US).

(74) Agents: **HAILS, Jr., Robert, L.** et al.; Kenyon & Kenyon,
333 West San Carlos Street, Suite 600, San Jose, CA 95110
(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,
SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, 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, BG, CH, CY, CZ, DE, DK, EE,
ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, SE, SI,
SK, 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 "Guid-
ance Notes on Codes and Abbreviations" appearing at the begin-
ning of each regular issue of the PCT Gazette.*

(54) Title: METHOD AND SYSTEM FOR MONITORING DMA STATUS

(57) Abstract: Embodiments of the invention provide a status register for each channel of a DMA controller. The status register may be used to monitor and record events that occur during DMA transfers, including timeouts and aborts.



WO 03/067444 A2

METHOD AND SYSTEM FOR MONITORING DMA STATUS

Field of the Invention

5 [0001] The present invention relates to computer systems, and more particularly to a method and system for monitoring events that occur during the operation of a DMA (Direct Memory Access) controller of a computer system.

Background of the Invention

10 [0002] DMA controllers are used in known computer systems for offloading, from a CPU, the tasks of performing data transfers to and from memory. If the CPU itself were used to transfer, for example, each byte of a program stored on an external floppy disk into the system memory for execution, operations would be severely slowed and overall system performance severely degraded. A DMA
15 controller can instead be used to load the program, freeing the CPU for other operations.

[0003] Typically, the CPU gives the DMA controller a start address and a byte count for data to be transferred to or from the memory. The DMA controller then proceeds to perform the data transfer.

20 [0004] Among disadvantages of known systems, however, is that events that may occur during transfers of data by the DMA controller are not adequately monitored or signaled. In known systems, for example, only the start and the completion of a data transfer are signaled. The completion event may be signaled, for example, by generating an interrupt or by setting a status bit in a
25 register that is polled by the CPU.

[0005] However, other events that impact system availability may occur during DMA data transfers, and need to be tracked. One example of such an event is a data transfer being aborted. An abort signal may be generated by a peripheral device to which a DMA transfer has begun, to signal that the DMA
30 transfer has been stopped short of successful completion. The peripheral device might abort a DMA transfer, for example, because of a software or hardware failure. In conventional systems, such an abort signal may be treated as equivalent to a completion signal indicating a successful transfer of data. Thus,

information which could be useful in identifying failing devices or system failures may be lost.

[0006] Another example of such an event is a DMA transfer which is begun but never completed (either successfully or unsuccessfully). A DMA controller might, for instance, set a status bit indicating that data transfer to a peripheral device had begun, but the peripheral device might not be ready to receive the data, or might have failed during the data transfer. The DMA controller might, in such a situation, continually attempt to send data to the device, but unsuccessfully because the device is not ready or has failed. The status bit would continue to indicate that a data transfer was underway, while in fact no data transfer was occurring. This would waste DMA resources, since the DMA channel attempting to send the data could be used for other data transfers.

[0007] A mechanism that generates a "timeout" is known for handling such situations. A timeout is a signal, typically an interrupt, generated when an operation does not complete in a time allotted for it. Conventional DMA controllers typically do not generate timeouts for stalled or incomplete data transfers as described above.

[0008] Additionally, DMA transfers may suffer from "unfair" bus arbitration. That is, a peripheral device that needs to transfer data to or from the memory may be connected to a bus shared with other devices, but access to the bus for the DMA controller, for purposes of effecting the data transfer, may be inadequate or unduly delayed.

[0009] A method and system are needed that address the foregoing concerns.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] Fig. 1 shows one possible configuration of a computer system according to embodiments of the invention;

[0011] Fig. 2 shows one possible configuration of a status register according to embodiments of the invention; and

[0012] Fig. 3 shows a process flow according to embodiments of the invention.

5 **DETAILED DESCRIPTION**

[0013] According to embodiments of the invention, a status register configured to monitor DMA activity may be provided. The status register may record events that occur during data transfers by a DMA controller. Data transfers performed by a DMA controller may be referred to in the art as "transactions." The
10 status register may be configured to record events including aborts, transaction completions, and timeouts.

[0014] Fig. 1 illustrates one possible hardware platform for implementing embodiments of the invention. Fig. 1 shows a computer system 100 with an Intel
15 ® Hub Architecture. This known architecture may comprise a CPU 101 coupled to a MCH (Memory Control Hub) 103 coupled to a memory 102. The MCH 103 may further be coupled to an ICH (I/O Control Hub) 104 comprising a DMA controller 108. The ICH 104 may further be coupled to a PCI (Peripheral Component Interconnect) bus 107, a LPC (Low Pin Count) bus 109 and an IDE (Integrated Drive Electronics) bus 106.

20 [0015] The ICH 104 may receive requests from peripheral devices connected to the PCI, LPC and IDE buses, or other buses, not shown. Examples of peripheral devices include a disk drive 110, shown connected to the IDE bus 106, and "super I/O" 111, shown connected to the LPC bus 109. The super I/O block 111 could include, for example, a UART (Universal Asynchronous
25 Receiver/Transmitter). Among the requests received from peripheral devices by the DMA controller 108 may be DMA requests, or, more particularly, requests to the DMA controller 108 to perform data transfers from a requesting peripheral device to the memory 102, or from the memory 102 to a requesting peripheral device. For example, when a FIFO buffer of the UART is full, the UART may
30 issue a DMA request to have the contents of the FIFO buffer transferred to the memory 102.

[0016] The ICH 104 may include logic configured to record events associated with operations of the DMA controller to transfer data between the

peripheral devices and the memory 102 in accordance with DMA requests, as described in greater detail in the following. A DMA channel may be provided for each peripheral device connected to the PCI, LPC and IDE buses, or other bus. According to embodiments of the invention, a status register for each DMA

5 channel may be provided. Fig. 2 shows one possible configuration of a status register 200 that could be provided for each DMA channel. The status register 200 includes a channel identifier field 201, an abort indicator field 202, a transaction complete (TC) indicator field 203, a timeout indicator field 204, and a time field 205.

10 [0017] When the DMA controller 108 receives a DMA request, a start time corresponding to the start of the data transfer may be written to the time field 205. Concurrently, a timer (not shown) may be started. The timer may measure the duration of a period established for determining when a timeout event has occurred. More particularly, if the TC field 203 is not written to after a

15 predetermined period of time to indicate successful completion of the transaction, it may be determined that a timeout event has occurred.

[0018] If the transaction completes successfully, the TC field 203 may be updated to indicate successful completion of the transaction. On the other hand, if an abort or timeout event occurs, the abort field 203 or timeout field 204 may be

20 respectively updated to indicate these events.

[0019] When the TC field 203, abort field 202 or timeout field 204 are updated, an interrupt may be generated. In response to the interrupt, the CPU 101 may read the corresponding status register. If the abort field 202 or timeout field 204 have been updated, this information may be used by system

25 management software to determine an appropriate course of action. For example, an error message may be generated to notify a user that a particular device has failed, or that a particular device has failed a number of times that exceeds a predetermined threshold number. This information may be used to determine that the device should be reset or replaced, for example, or in

30 diagnosing recurring system problems. Alternatively, such information could be used to determine that the system should be re-booted.

[0020] Fig. 3 illustrates a process according to embodiments of the invention.

[0021] As shown in blocks 300 and 301, a request from peripheral devices connected to buses 106, 107 or 109 may be received by the DMA controller 108, and decoded to determine whether the request is a DMA request. The operations represented in blocks 300 and 301 may be repeated to continually check for the presence of a DMA request.

[0022] If the request is a DMA request, the DMA request may specify the DMA channel assigned to the requesting peripheral device. As shown in block 302, an interrupt may be generated, to cause the CPU to initialize the status register corresponding to the DMA channel specified in the DMA request.

Alternatively, the DMA controller may initialize the status register, without involving the CPU 101. The CPU 101 may have previously programmed the DMA controller 108 with the byte count, starting memory address, and direction of the data transfer requested. As noted above, a timer may also be started to measure a period for determining whether a timeout has occurred.

[0023] A transaction may then be initiated in response to the DMA request, as shown in block 303. As shown in blocks 304-309, the status of the transaction resulting from the DMA request may be monitored. More particularly, a sequence of tests as shown in blocks 304-306 may be continually applied to determine the status of the transaction associated with the request.

[0024] A test as shown in block 304 may be performed in order to determine whether the transaction has successfully completed. Such a determination could be made, for example, based on the byte count supplied by the CPU as described above. The DMA controller 108 keeps track of how many bytes have been transmitted or received during the transaction. Once the DMA controller 108 has transmitted or received the number of bytes specified in the byte count, an interrupt may be generated and the TC field of the corresponding status register may be appropriately updated to record the successful completion of the transaction, as shown in block 307.

[0025] If the test shown in block 304 does not indicate that a transaction complete has occurred, as test as shown in block 305 may be performed to determine whether a transaction timeout has occurred. Such a determination could be made as described above, wherein if a transaction complete is not recorded in the corresponding status register after a predetermined period of time,

it may be determined that a timeout event has occurred. If so, an interrupt may be generated and the timeout indicator field 204 of the corresponding status register may be appropriately updated to record the timeout, as shown in block 308.

5 [0026] If the test as shown in block 305 does not indicate that a transaction timeout has occurred, a test as shown in block 306 may be performed to determine whether a DMA abort has occurred. A DMA abort would typically be indicated by an abort signal generated by the peripheral device and placed on a bus decoded by the ICH 104. If an abort signal is decoded on the bus, an interrupt may be generated and the abort indicator field 204 of the corresponding
10 status register may be appropriately updated to record the abort, as shown in block 309.

[0027] As noted above, blocks 304-306 may continue to be executed until either a transaction complete, a transaction timeout, or a DMA abort are detected. Then, the process may return to blocks 300 and 301 to continually check for the
15 presence of a DMA request.

[0028] In view of the foregoing, it may be appreciated that a method and system have been disclosed for monitoring and recording events that occur during DMA data transfers. Information obtained by monitoring and recording events as described may assist in diagnosing system problems and thereby increasing
20 system reliability and availability.

[0029] Additionally, embodiments of the invention as discussed above address the problem of unfair bus arbitration. Notification to the DMA controller of conditions indicative of unfair bus arbitration could be carried out via the same technique as applied, for example, in the event of a timeout. While typically the
25 DMA controller would not be able to distinguish between unfair bus arbitration and an unresponsive device, there are other mechanisms available for determining whether a device that may be subject to unfair bus arbitration is functioning properly. System management software, for example, could be configured to distinguish a difference between a failed device and unfair bus arbitration.

30 [0030] Several embodiments of the present invention are specifically illustrated and described herein. However, it will be appreciated that modifications and variations of the present invention are covered by the above teachings and

within the purview of the appended claims without departing from the spirit and intended scope of the invention.

WHAT IS CLAIMED IS:

1. A method comprising:
 - (i) providing a status register for each channel of a DMA controller; and
 - (ii) recording, in said status register, events that occur during a data
- 5 transfer performed by said DMA controller, wherein said events include a timeout.
2. The method of claim 1, wherein said events further include an abort.
3. The method of claim 1, wherein (ii) comprises:
 - 10 receiving a request on a bus connected to a peripheral device;
 - determining whether said request is a DMA request; and
 - monitoring a status of a data transfer resulting from said DMA request.
4. The method of claim 3, further comprising:
 - 15 if a timeout or an abort occurs during said monitoring:
 - generating an interrupt; and
 - recording said timeout or abort in said status register.
5. The method of claim 1, further comprising:
 - 20 generating an error message when a number of aborts or timeouts
 - recorded in said status register exceeds a predetermined number.
6. A system comprising:
 - a memory;
 - 25 a DMA controller coupled to said memory;
 - a peripheral device;
 - a bus coupled to said peripheral device and said DMA controller; and
 - a status register to record events associated with operations of said DMA
 - controller to transfer data between said peripheral device and said memory;
 - 30 wherein said events include a timeout .
7. The system of claim 6, wherein said events further include an abort.

8. The system of claim 6, further comprising logic configured to:
receive a request on said bus;
determine whether said request is a DMA request; and
monitor a status of a data transfer resulting from said DMA request.

5

9. The system of claim 8, wherein said logic is further configured to, if a timeout or an abort occurs during said monitoring:
generate an interrupt; and
record said timeout or abort in said status register.

10

10. The system of claim 6, wherein said status register includes at least a channel identifier, an abort indicator, and a timeout indicator.

11. The system of claim 6, wherein said bus is one of a PCI bus, a LPC bus, and an IDE bus.

15

12. A method for increasing availability and reliability of a computer system, comprising:

(a) monitoring events associated with the operations of a DMA controller of said computer system, said events including timeouts; and
(b) recording the occurrence of said timeouts.

20

13. The method of claim 12, wherein (a) comprises:
determining whether a transaction performed by said DMA controller has completed successfully; and
if not, determining whether a timeout or an abort has occurred.

25

14. The method of claim 12, wherein (b) comprises:
writing to a status register associated with said DMA controller to indicate the occurrence of a timeout or an abort.

30

15. The method of claim 12, further comprising generating an error message to notify a user of said system of the occurrence of a timeout or an abort.

1/2

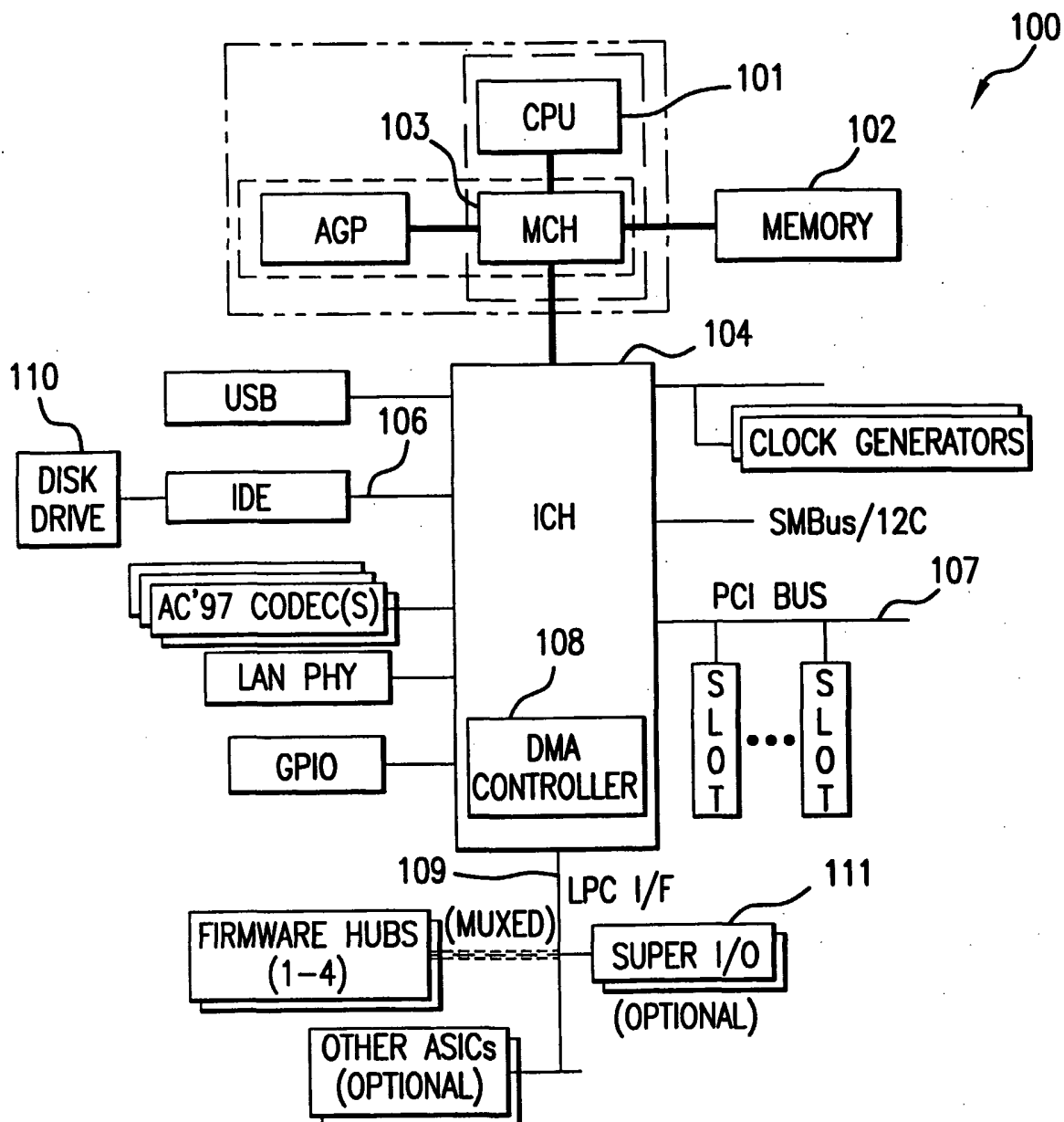


FIG. 1

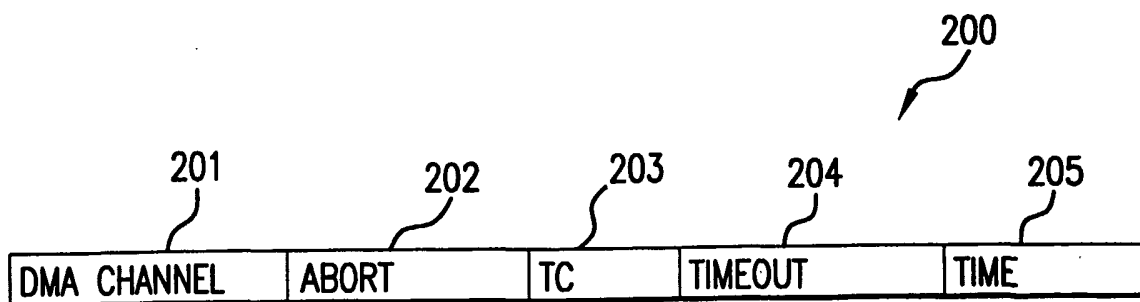


FIG. 2

2/2

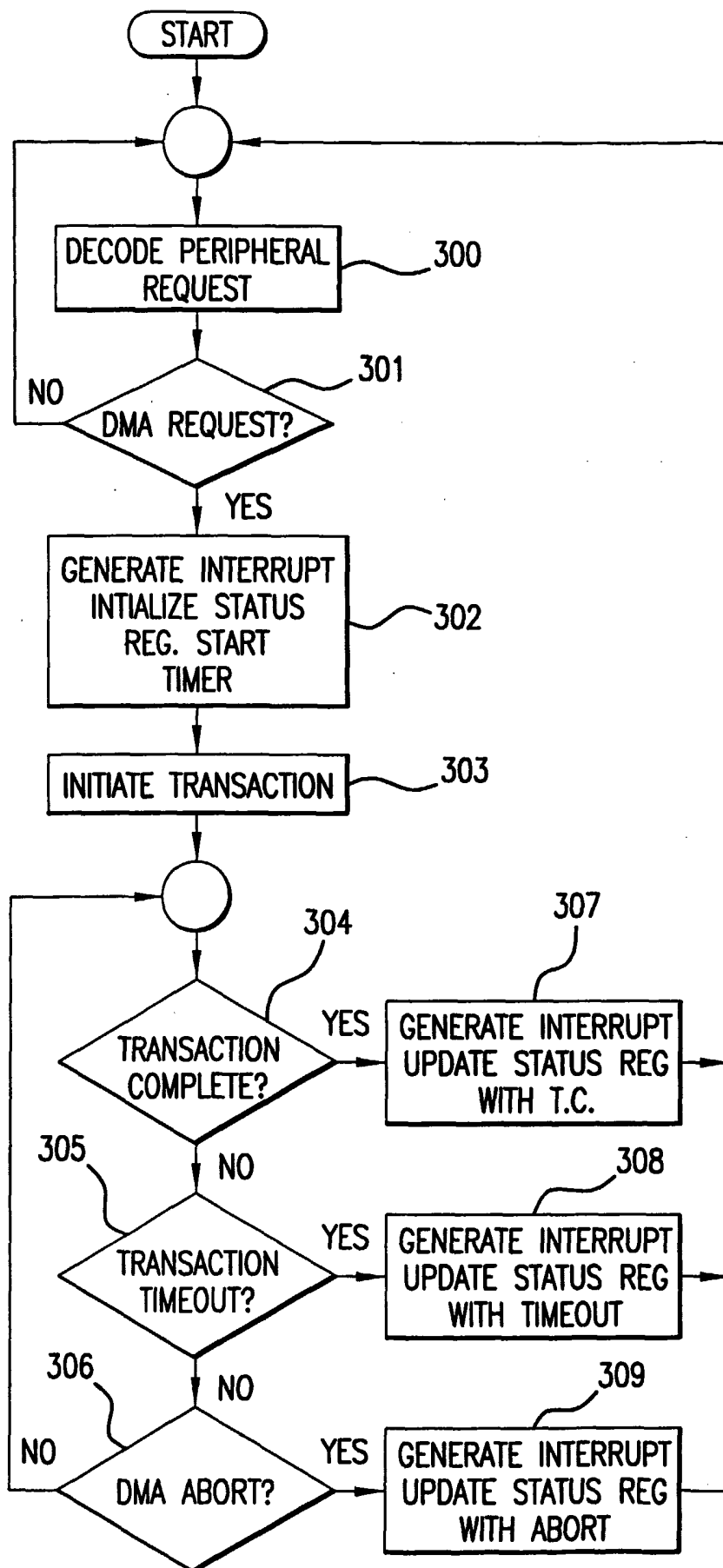


FIG.3